

Optimization for Supervised Machine Learning: Randomized Algorithms for Data and Parameters

Dissertation by

Filip Hanzely

In Partial Fulfillment of the Requirements

For the Degree of

Doctor of Philosophy

King Abdullah University of Science and Technology

Thuwal, Kingdom of Saudi Arabia

August, 2020

EXAMINATION COMMITTEE PAGE

The dissertation of Filip Hanzely is approved by the examination committee

Committee Chairperson: Peter Richtárik

Committee Members: Stephen Wright, Tong Zhang, Raul Fidel Tempone, Bernard Ghanem

© August, 2020

Filip Hanzely

All Rights Reserved

ABSTRACT

Optimization for Supervised Machine Learning: Randomized Algorithms for Data and Parameters

Filip Hanzely

Many key problems in machine learning and data science are routinely modeled as optimization problems and solved via optimization algorithms. With the increase of the volume of data and the size and complexity of the statistical models used to formulate these often ill-conditioned optimization tasks, there is a need for new efficient algorithms able to cope with these challenges.

In this thesis, we deal with each of these sources of difficulty in a different way. To efficiently address the big data issue, we develop new methods which in each iteration examine a small random subset of the training data only. To handle the big model issue, we develop methods which in each iteration update a random subset of the model parameters only. Finally, to deal with ill-conditioned problems, we devise methods that incorporate either higher-order information or Nesterov's acceleration/momentum. In all cases, randomness is viewed as a powerful algorithmic tool that we tune, both in theory and in experiments, to achieve the best results.

Our algorithms have their primary application in training supervised machine learning models via regularized empirical risk minimization, which is the dominant paradigm for training such models. However, due to their generality, our methods can be applied in many other fields, including but not limited to data science, engineering, scientific computing, and statistics.

ACKNOWLEDGEMENTS

I owe my deepest gratitude to my supervisor Peter Richtárik. Thank you very much for your guidance; it allowed me to get the best out of myself. Thanks a lot for the extraordinary support, career advice, and tons of encouragement. You showed me each aspect of being a complete researcher and always guided me in that direction.

Next, I would like to thank all members of our research group for countless stimulating discussions, namely: Konstantin Mishchenko, Samuel Horváth, Slavomír Hanzely, Robert Gower, Aritra Dutta, Nicolas Loizou, Alibek Sailanbayev, Jakub Konečný, Dominik Csiba, Elnur Gasanov, Eduard Gorbunov, Dmitry Kovalev, Adil Salim, Yazeed Basyoni, Mher Safaryan, El Houcine Bergou, Xun Qian, Zhize Li, and Egor Shulgin.

I am very grateful to all the great researchers I had a chance to collaborate with, especially Lin Xiao, Yurii Nesterov, Sebastian Stich, Jingwei Liang, and Nikita Doikov. I would also like to thank Michael Mahoney, Martin Jaggi, Alex D'Aspremont, Adrien Taylor, Praneeth Karimireddy, and Haihao Lu for multiple fruitful discussions. Further, I owe a big thanks to my internship hosts Rodolphe Jenatton and Sashank Reddi at Amazon and Google respectively as well as to other people I had a chance to interact with, namely Mathias Seeger, Srinadh Bhojanapalli, Cédric Archambeau and Sanjiv Kumar. I learned a lot from all of you!

I appreciate a lot all the support I received both from KAUST and from the Visual Computing Center at KAUST; I feel extremely lucky for all the opportunities I had. I am also very grateful to my defense committee, namely Stephen J Wright, Tong Zhang, Raúl F Tempone, and Bernard Ghanem.

I would like to thank all my friends that made my stay at KAUST pleasant. Last but not least, I am eminently grateful to my family for their love and support.

TABLE OF CONTENTS

Examination Committee Page	2
Copyright	3
Abstract	4
Acknowledgements	5
Table of Contents	6
List of Abbreviations	16
List of Figures	17
List of Tables	24
1 Introduction	26
1.1 Technical preliminaries and basic algorithms	27
1.1.1 Smoothness and convexity	27
1.1.2 Gradient descent	28
1.1.3 Nesterov's acceleration	29
1.1.4 Proximal operator and proximal gradient descent	30
1.1.5 Incorporating randomness	32
1.2 From finite sum to coordinate descent and back	33
1.2.1 From finite sum to coordinate descent	34
1.2.2 From coordinate descent to finite sum: three approaches	35
1.2.3 Towards better stochastic condition numbers	37
1.3 Relationship among the chapters	38
1.4 Outline and individual contributions	42
1.4.1 Accelerated coordinate descent with arbitrary sampling and best rates for minibatches (Chapter 2)	43
1.4.2 SEGA: Variance reduction via gradient sketching (Chapter 3)	44
1.4.3 99% of Worker-Master Communication in Distributed Optimization is Not Needed (Chapter 4)	45
1.4.4 One method to rule them all: Variance reduction for data, parameters and many new methods (Chapter 5)	45
1.4.5 A unified theory of SGD: Variance reduction, sampling, quantization and coordinate descent (Chapter 6)	46

1.4.6	Variance reduced coordinate descent with acceleration: New method with a surprising application to finite-sum problems (Chapter 7)	46
1.4.7	Federated learning of a mixture of global and local models (Chapter 8)	47
1.4.8	Stochastic subspace cubic Newton (Chapter 9)	47
1.4.9	Accelerated stochastic matrix inversion: General theory and speeding up BFGS rules for faster second-order optimization (Chapter 10)	47
1.4.10	Excluded papers	48

2 Accelerated Coordinate Descent with Arbitrary Sampling and Best Rates for Minibatches **49**

2.1	Arbitrary sampling and minibatching	50
2.2	Contributions	52
2.3	The ACD algorithm	54
2.4	Importance sampling for minibatches	55
2.4.1	Sampling 1: standard uniform minibatch sampling	57
2.4.2	Sampling 2: importance sampling for minibatches	57
2.4.3	Sampling 3: another importance sampling for minibatches	57
2.5	Experiments	58
2.5.1	Synthetic quadratics	59
2.5.2	Logistic regression	61
2.5.3	Support vector machines	64
2.6	Conclusion	65

3 SEGA: Variance Reduction via Gradient Sketching **66**

3.1	Gradient sketching	66
3.1.1	Related work	67
3.2	Contributions	68
3.3	The SEGA algorithm	68
3.3.1	SEGA as a variance-reduced method	69
3.3.2	SEGA versus coordinate descent	70
3.4	Convergence of SEGA for general sketches	71
3.4.1	Smoothness assumptions	71
3.4.2	Main result	71
3.5	Convergence of SEGA for coordinate sketches	73
3.5.1	Defining \mathcal{D} : samplings	74
3.5.2	Non-accelerated method	74
3.5.3	Accelerated method	75
3.6	Experiments	76
3.6.1	Comparison to projected gradient descent	76
3.6.2	Comparison to zeroth-order optimization methods	77
3.6.3	Subspace SEGA	78
3.6.4	Comparison to randomized coordinate descent	78
3.6.5	Evolution of iterates: Extra plots	79
3.7	Conclusion	79

4	99% of Worker-Master Communication in Distributed Optimization is Not Needed	83
4.1	From gradient descent to block coordinate descent and back	84
4.1.1	From gradient descent to independent block coordinate descent	85
4.2	Contributions	85
4.3	Practical implications and limitations	86
4.3.1	Main limitation	86
4.3.2	Practical implications	87
4.4	Independent block coordinate descent	88
4.4.1	The IBCD algorithm	88
4.4.2	Convergence of IBCD	88
4.4.3	Optimal block sizes	89
4.5	Variance reduction	89
4.5.1	Shared data ISAGA	90
4.5.2	Distributed ISAGA	91
4.6	SGD	92
4.7	Acceleration	94
4.8	Beyond interpolation without shared data and regularization	96
4.9	Experiments	97
4.9.1	Simple, well understood experiment	98
4.9.2	ISGD	100
4.9.3	IASGD	102
4.9.4	ISAGA	104
4.9.5	ISEGA	106
4.10	Conclusion	108
5	One Method to Rule Them All: Variance Reduction for Data, Parameters and Many New Methods	109
5.1	Contributions	111
5.2	Sketching	112
5.3	The GJS algorithm	113
5.4	Theory	114
5.5	Special cases	116
5.6	Experiments	118
5.6.1	SEGA and SVRCD with importance sampling	118
5.6.2	SVRCD: effect of ρ	118
5.6.3	ISAEGA	119
5.6.4	LSVRG with importance sampling	120
5.7	Conclusion	123
6	A Unified Theory of SGD: Variance Reduction, Sampling, Quantization and Coordinate Descent	126
6.1	The many faces of stochastic gradient descent	127
6.2	Contributions	129
6.3	Main result	130

6.3.1	Key assumption	130
6.3.2	Main theorem	131
6.4	The classic, the recent and the brand new	132
6.5	Experiments	135
6.5.1	SGD-MB: remaining experiments and exact problem setup.	136
6.5.2	Experiments on SGD-star	136
6.5.3	Experiments on N-SEGA	138
6.6	Conclusion	140
7	Variance Reduced Coordinate Descent with Acceleration: New Method With a Surprising Application to Finite-Sum Problems	141
7.1	Contributions	142
7.2	Preliminaries	143
7.3	Better rates for SEGA and SVRCD	143
7.4	Connection between SEGA (SVRCD) and SAGA (LSVRG)	145
7.4.1	Convergence rate of SAGA and LSVRG	146
7.4.2	SAGA is a special case of SEGA	147
7.5	The ASVRCD algorithm	148
7.6	Connection between ASVRCD and L-Katyusha	150
7.6.1	Convergence rate of L-Katyusha	150
7.6.2	L-Katyusha is a special case of ASVRCD	151
7.7	Experiments	151
7.7.1	The effect of acceleration and importance sampling	153
7.7.2	The effect of \mathbf{W}	153
7.8	Conclusion	153
8	Federated Learning of a Mixture of Global and Local Models	155
8.1	Federated learning	155
8.1.1	Some issues with current approaches to FL	156
8.2	Contributions	156
8.3	New formulation of FL	158
8.3.1	Technical preliminaries	159
8.3.2	Characterization of optimal solutions	160
8.4	The L2GD algorithm	161
8.4.1	Understanding communication	161
8.4.2	The dynamics of local GD and averaging steps	162
8.4.3	Convergence theory	163
8.4.4	Optimizing the rate and communication	163
8.5	The L2SGD+ algorithm	164
8.5.1	Setup	165
8.5.2	Theory	165
8.6	Experiments	166
8.6.1	Comparison of the methods	167
8.6.2	Effect of p	168
8.6.3	Effect of λ	169

8.7	Conclusion	170
9	Stochastic Subspace Cubic Newton Method	172
9.1	Subspace descent methods	172
9.2	Contributions	173
9.3	Preliminaries	174
9.4	The SSCN algorithm	175
9.4.1	Solving the subproblem	175
9.4.2	Special cases	176
9.5	Related literature	177
9.6	Global complexity bounds	178
9.6.1	Setup	178
9.6.2	Theory	179
9.7	Local convergence	181
9.8	Applications	183
9.8.1	Linear models	183
9.8.2	Dual of linear models	184
9.9	Experiments	184
9.9.1	Logistic regression	184
9.9.2	Log-sum-exp	186
9.10	Conclusion	188
10	Accelerated Stochastic Matrix Inversion: General Theory and Speeding up BFGS Rules for Faster Second-Order Optimization	192
10.1	Sketch-and-project for linear systems	193
10.2	Contributions	194
10.3	Accelerated stochastic algorithm for matrix inversion	195
10.3.1	The algorithm	196
10.3.2	Key assumptions and quantities	197
10.3.3	Convergence and change of the norm	197
10.3.4	Coordinate sketches with convenient probabilities	198
10.4	Accelerated stochastic BFGS update	198
10.4.1	The AMI algorithm	199
10.4.2	Vectorizing – a different insight	200
10.4.3	Accelerated BFGS as an optimization algorithm	200
10.5	Experiments	201
10.5.1	Accelerated matrix inversion	201
10.5.2	BFGS optimization method	214
10.6	Conclusion	216
11	Concluding Remarks	217
11.1	Summary	217
11.2	Future Research Work	219
	References	221

Appendices	243
A Table of Frequently Used Notation	244
B Appendix for Chapter 2	248
B.1 Proof of Theorem 2.3.2	248
B.1.1 Proof of inequality (2.14)	248
B.1.2 Descent lemma	248
B.1.3 Key technical inequality	248
B.1.4 Proof of the theorem	249
B.2 Better rates for minibatch CD (without acceleration)	251
B.2.1 Two uniform samplings and one new importance sampling	251
B.2.2 Comparing the samplings	253
B.3 Proofs for Section 2.4	255
B.3.1 Proof of Theorem 2.4.1	255
B.3.2 Proof of Lemma 2.4.2	256
B.3.3 Bound on $c(S_1, \mathbf{M})$	256
B.3.4 Proof of Theorem 2.4.3	257
C Appendix for Chapter 3	260
C.1 Proofs for Section 3.4	260
C.1.1 Proof of Theorem 3.4.2	261
C.1.2 Proof of Lemma C.1.3	262
C.1.3 Proof of Lemma C.1.4	263
C.2 Proofs for Section 3.5	263
C.2.1 Technical lemmas	263
C.2.2 Proof of Theorem 3.5.2	264
C.2.3 Proof of Corollary 3.5.3	265
C.2.4 Accelerated SEGA with arbitrary sampling	265
C.2.5 Proof of Lemma C.2.3	270
C.2.6 Proof of Lemma C.2.4	271
C.3 Subspace SEGA: a more aggressive approach	272
C.3.1 The algorithm	272
C.3.2 Lemmas	273
C.3.3 Main result	275
C.3.4 The conclusion of subspace SEGA	276
C.4 Simplified analysis of SEGA	276
C.4.1 Technical lemmas	277
C.4.2 Proof of Theorem C.4.1	278
D Appendix for Chapter 4	280
D.1 IBGD: Bernoulli alternative to IBCD	280
D.2 Asynchronous ISGD	281
D.3 Proofs for Section 4.4	282
D.3.1 Key techniques	282

D.3.2	Proof of Theorem 4.4.2	283
D.3.3	Proof of Theorem D.1.1	283
D.4	Missing parts from Sections 4.5 and 4.5.2	283
D.4.1	Useful lemmata	283
D.4.2	Proof of Theorem 4.5.3	286
D.4.3	Proof of Theorem 4.5.1	287
D.5	Proofs for Section 4.6	288
D.5.1	Useful lemmas	288
D.5.2	Proof of Theorem 4.6.3	291
D.5.3	Proof of Theorem 4.6.5	292
D.6	Missing parts from Section 4.7	294
D.6.1	Proof of Lemma 4.7.3	294
D.7	Proofs for Section 4.8	295
D.7.1	Useful lemmata	295
D.7.2	Proof of Theorem 4.8.1	297
D.8	Proofs for Section D.2	298
D.8.1	Useful lemmata	299
D.8.2	Proof of Theorem D.2.1	300
E	Appendix for Chapter 5	303
E.1	Summary of complexity results	303
E.2	Several lemmas	303
E.2.1	Existence lemma	303
E.2.2	Smoothness lemmas	304
E.2.3	Projection lemma	306
E.2.4	Decomposition lemma	307
E.3	Proof of Theorem 5.4.2	308
E.4	Special cases: SAGA-like methods	309
E.4.1	Basic variant of SAGA [37]	309
E.4.2	SAGA with arbitrary sampling	310
E.5	Special cases: SEGA-like methods	311
E.5.1	Basic variant of SEGA [77]	311
E.5.2	SEGA with arbitrary sampling	311
E.5.3	SVRCD with arbitrary sampling	313
E.6	Special cases: SGD-star	313
E.7	Special cases: loopless SVRG with arbitrary sampling (LSVRG)	314
E.8	Special cases: methods with Bernoulli \mathcal{U}	315
E.8.1	B2 (Bernoulli \mathcal{S})	315
E.8.2	LSVRG-inv (right \mathcal{S})	316
E.8.3	SVRCD-inv (left \mathcal{S})	317
E.9	Special cases: combination of left and right sketches	318
E.9.1	RL (right sampling \mathcal{S} , left unbiased sampling \mathcal{U})	318
E.9.2	LR (left sampling \mathcal{S} , right unbiased sampling \mathcal{U})	318
E.10	Special cases: joint left and right sketches	319
E.10.1	SAEGA	319

E.10.2	SVRCDG	320
E.10.3	ISAEGA (with distributed data)	321
E.11	Special cases: JacSketch	323
E.12	Special cases: proofs	324
E.12.1	SAGA methods: proofs	324
E.12.2	SEGA methods: proofs	326
E.12.3	Setup for Corollary E.6.1	327
E.12.4	Setup for Corollary E.7.1	327
E.12.5	Methods with Bernoulli \mathcal{U} : proofs	328
E.12.6	Combination of left and right sketches: proofs	329
E.12.7	Joint sketches: proofs	330
E.12.8	Setup for Corollary E.11.1	333
E.13	Convergence under strong growth condition	334
E.13.1	Technical proposition and lemma	334
E.13.2	Convergence proof	335
F	Appendix for Chapter 6	338
F.1	Special cases	338
F.1.1	Proximal SGD for stochastic optimization	338
F.1.2	SGD-SR	339
F.1.3	SGD-MB	340
F.1.4	SGD-star	343
F.1.5	SAGA	344
F.1.6	N-SAGA	346
F.1.7	SEGA	348
F.1.8	N-SEGA	349
F.1.9	SVRG	351
F.1.10	LSVRG	352
F.1.11	DIANA	353
F.1.12	Q-SGD-SR	356
F.1.13	VR-DIANA	357
F.1.14	JacSketch	359
F.1.15	Interpolation between methods	361
F.2	Proofs for Section 6.3	365
F.2.1	Basic facts and inequalities	365
F.2.2	Proof of Lemma 6.3.3	365
F.2.3	Proof of Theorem 6.3.4	366
G	Appendix for Chapter 7	367
G.1	Missing lemmas and proofs: SAGA/LSVRG is a special case of SEGA/SVRCD	367
G.1.1	Proof of Lemma 7.4.3	367
G.1.2	Proof of Lemma 7.4.4	368
G.2	Missing lemmas and proofs: ASVRCD	369
G.2.1	Technical lemmas	369
G.2.2	Proof of Theorem 7.5.1	372

G.2.3	Proof of Lemma 7.5.2	373
G.2.4	Proof of Lemma 7.5.3	374
G.3	Missing lemmas and proofs: L-Katyusha as a particular case of ASVRCD	375
G.3.1	Proof of Lemma 7.6.3	375
G.4	Tighter rates for GJS by exploiting prox and proof of Theorem 7.3.2	376
G.4.1	Towards the proof of Theorem G.4.1	377
G.4.2	Proof of Theorem 7.3.2	380
H	Appendix for Chapter 8	381
H.1	Remaining algorithms	381
H.1.1	Local GD with variance reduction	381
H.1.2	Efficient implementation of L2SGD+	383
H.1.3	Local SGD with variance reduction – general method	383
H.1.4	Local stochastic algorithms	387
H.2	Missing lemmas and proofs	389
H.2.1	Gradient and Hessian of Φ	389
H.2.2	Proof of Theorem 8.3.2	391
H.2.3	Proof of Theorem 8.3.3	392
H.2.4	Proof of Lemma 8.4.2	393
H.2.5	Proof of Theorem 8.4.3	393
H.2.6	Proof of Corollary 8.4.4	393
H.2.7	Proof of Corollary 8.5.3	394
H.2.8	Proof of Theorems 8.5.2, H.1.5, and H.1.6	394
I	Appendix for Chapter 9	398
I.1	Missing lemmas and proofs from Section 9.3	398
I.1.1	Explicit update	398
I.1.2	Proof of Lemma 9.3.3	398
I.1.3	Proof of Lemma 9.3.2	399
I.2	Proofs for Section 9.6	399
I.2.1	Proof of Lemma 9.6.2	399
I.2.2	Proof of Lemma 9.6.7	399
I.2.3	Proof of Theorem 9.6.8	401
I.2.4	Proof of Theorem 9.6.10	402
I.3	Proofs for Section 9.7	403
I.3.1	Several technical lemmas	403
I.3.2	Proof of Lemma 9.7.1	405
I.3.3	Proof of Theorem 9.7.2	406
J	Appendix for Chapter 10	408
J.1	Proofs for Section 10.3	408
J.1.1	Proof of Lemma 10.3.2	408
J.1.2	Technical lemmas to prove Theorem 10.3.3	409
J.1.3	Proof of Theorem 10.3.3	410
J.1.4	Changing norm	412

J.2	Proof of Corollary 10.3.4	413
J.3	Adding a stepsize	414
J.4	Allowing for different η	414
J.5	Proof of Theorem J.3.1	415
J.6	Proof of Theorem J.4.1	416
J.7	Proofs and further comments on Section 10.4	417
	J.7.1 Proof of Theorem 10.4.1	417
	J.7.2 Matrix inversion as linear system	419
J.8	Linear operators in Euclidean spaces	420
	J.8.1 Positive operators	421
	J.8.2 Pseudoinverse	422
K	Accepted Papers	423
L	Submitted Papers	424

LIST OF ABBREVIATIONS

Algorithm names

ACD	Accelerated CD
AMI	Accelerated Matrix Inversion
ASEGA	Accelerated SEGA
ASVRCD	Accelerated SVRCD
BFGS	Broyden-Fletcher-Goldfarb-Shanno
CD	Coordinate Descent
GD	Gradient Descent
GJS	Generalized Jacobian Sketching
IBCD	Independent Block Coordinate Descent
ISEGA	Independent SEGA
L2GD	Loopless LGD
LGD	Local GD
LSGD	Local SGD
SEGA	SkEtched Gradeint Algorithm
SGD	Stochastic GD
SSCN	Stochastic Subspace Cubic Newton
SVRCD	Stochastic Variance Reduced CD

Miscellaneous

ERM	Empirical Risk Minimization
ESO	Expected Separable Overapproximation
FL	Federated Learning
VR	Variance Reduction

LIST OF FIGURES

1.1	Graph depicting the relationships among the chapters of this thesis. Blue dashed arrow indicates motivation among chapters, while green dotted arrow indicates a significant insight that chapters shed on each other. As an example, let us explain the edges of Chapter 3 (SEGA): the development of SEGA was enabled by our results on CD (Chapter 2) and motivated us to develop the results contained in Chapters 4, 5 and 7. Further, Chapter 5 recovers/improves upon the convergence rate of SEGA, Chapter 6 enables a partial variance reduction in SEGA and lastly, Chapter 5 shows that SAGA is a special case of SEGA.	43
2.1	Coordinate descent. Comparison of accelerated, nonaccelerated algorithm with both importance and τ nice sampling for a various quadratic problems.	60
2.2	Coordinate descent. Comparison of speedup gained by both τ -nice sampling and importance sampling with and without acceleration on various quadratic problems.	61
2.3	Accelerated coordinate descent applied on the logistic regression problem, for various LibSVM datasets and minibatch sizes τ	62
2.4	ACD applied on the logistic regression problem, for various rescaled LibSVM datasets and minibatch sizes τ	63
2.5	Six variants of coordinate descent (AN, AU, NN, NU, AN2 and AU2) applied to a logistic regression problem, with minibatch sizes $\tau = 1, 8, 64$ and 512.	64
2.6	Accelerated coordinate descent applied on the dual of of SVM with squared hinge loss, for various LibSVM datasets.	65
3.1	Iterates of SEGA and CD	69
3.2	Convergence of SEGA and PGD on synthetic problems with $d = 500$. The indicator “Xd” in the label indicates the setting where the cost of solving linear system is Xd times higher comparing to the cost of evaluating a single directional derivative. Recall that a linear system is solved after each d oracle calls. Stepsizes $1/\lambda_{\max}(\mathbf{M})$ and $1/(d\lambda_{\max}(\mathbf{M}))$ were used for PGD and SEGA, respectively.	77
3.3	Comparison of SEGA and randomized direct search for various problems. Theory supported stepsizes were chosen for both methods. 500 dimensional problem.	78
3.4	Comparison of SEGA with sketches from a correct subspace versus coordinate sketches naiveSEGA. Step size chosen according to theory. 1000 dimensional problem.	79

3.5	Comparison of SEGA and ASEGA with corresponding coordinate descent methods for $\psi \equiv 0$	80
3.6	Evolution of iterates of SEGA, CD and biasSEGA (updates made via h^{k+1} instead of g^k).	81
3.7	Iterates of SEGA, CD and biasSEGA (updates made via h^{k+1} instead of g^k). Different starting point.	81
3.8	Iterates of projected SEGA, projected CD (which do not converge) and projected biasSEGA (updates made via h^{k+1} instead of g^k). The constraint set is represented by the shaded region.	81
4.1	Comparison of gradient descent, (standard) coordinate descent, (standard) coordinate descent with importance sampling and Algorithm 8 on artificial quadratic problem (4.15).	99
4.2	Behavior of Algorithm 8 for different τ on a simple artificial quadratic problem (4.15).	100
4.3	Comparison of SGD (gradient evaluated on a single datapoint) and Algorithm 11 with $n\tau = 1$. Constant $\alpha = \frac{1}{5L}$ was used for each algorithm. Label "batch_size" indicates how big minibatch was chosen for stochastic gradient of each worker's objective.	101
4.4	Behavior of Algorithm 11 while varying τ . Label "SGD" corresponds to the choice $n = 1, \tau = 1$. Stepsize $\alpha = \frac{1}{3L}$ was used in every case.	102
4.5	Comparison of Algorithm 12 for various (n, τ) such that $n\tau = 1$. Label "ASGD" corresponds to the choice $n = 1, \tau = 1$. Label "batch_size" indicates how big minibatch was chosen for stochastic gradient of each worker's objective. Parameter ρ was chosen by grid search.	103
4.6	Behavior of Algorithm 12 while varying τ . Label "ASGD" corresponds to the choice $n = 1, \tau = 1$. Parameter ρ was chosen by grid search.	104
4.7	Comparison of SAGA and Algorithm 9 for various values n and $\tau = n^{-1}$. Stepsize $\alpha = \frac{1}{L(3n^{-1} + \tau)}$ is chosen in each case.	105
4.8	Comparison of Algorithm 9 for different values of τ . Stepsize $\alpha = \frac{1}{L(3n^{-1} + \tau)}$ is chosen in each case. For this experiment, we choose smaller regularization; $\lambda = 0.000025$	106
4.9	Comparison of Algorithm 13 for various (n, τ) such that $n\tau = 1$ and GD. Stepsize $\frac{1}{L(1 + \frac{1}{n\tau})}$ was chosen for Algorithm 13 and $\frac{1}{2L}$ for GD.	107
4.10	Comparison of Algorithm 13 for different values of τ . Stepsize $\alpha = \frac{1}{L(1 + \frac{1}{n\tau})}$ is chosen in each case.	108
5.1	Comparison of SEGA-AS, SVRCD-AS, SEGA and proximal gradient on 4 quadratic problems given by Table 5.2. SEGA-AS, SVRCD-AS and SEGA compute single partial derivative each iteration (SVRCD computes all of them with probability ρ), SEGA-AS, SVRCD-AS with probabilities proportional to diagonal of \mathbf{M}	119

5.2	The effect of ρ on convergence rate of SVRCD on quadratic problems from Table 5.2. In every case, probabilities were chosen proportionally to the diagonal of \mathbf{M} and only a single partial derivative is evaluated in \mathcal{S}	120
5.3	ISAEGA applied on LIBSVM [23] datasets with $\lambda = 4 \cdot 10^{-5}$. Axis y stands for relative suboptimality, i.e. $\frac{f(x^k) - f(x^*)}{f(x^k) - f(x^0)}$	122
5.4	LSVRG applied on LIBSVM [23] datasets with $\lambda = 10^{-5}$. Axis y stands for relative suboptimality, i.e. $\frac{f(x^k) - f(x^*)}{f(x^k) - f(x^0)}$	124
5.5	LSVRG applied on LIBSVM [23] datasets. For a9a, $\lambda = 0$ and $\rho = \frac{1}{n}$ was chosen; for w8a, $\lambda = 10^{-8}$ and $\rho = \frac{3}{n}$ was chosen. Axis y stands for relative suboptimality, i.e. $\frac{f(x^k) - f(x^*)}{f(x^k) - f(x^0)}$	125
6.1	SGD-MB and independent SGD applied on LIBSVM [23] datasets with regularization parameter $\lambda = 10^{-5}$. Axis y stands for relative suboptimality, i.e. $\frac{f(x^k) - f(x^*)}{f(x^k) - f(x^0)}$. Title label “unif” corresponds to probabilities chosen by (i) while label “imp” corresponds to probabilities chosen by (ii). Lastly, legend label “r” corresponds to “replacement” with value “True” for SGD-MB and value “False” for independent SGD.	137
6.2	Comparison of SGD-star, SGD and SAGA on least squares problem. . . .	138
6.3	N-SEGA applied on constrained least squares problem with noised partial derivative oracle. Legend labels stand for the magnitude σ^2 of the oracle noise.	139
7.1	Comparison of both ASVRCD and SVRCD with importance and uniform sampling.	153
7.2	Comparison of ASVRCD and SVRCD for various \mathbf{W} . Label ‘r’ indicates the dimension of $\mathbf{Range}(\mathbf{W})$	154
8.1	Distance of solution $x(\lambda)$ of (8.2) to pure local solution $x(0)$ and global solution $x(\infty)$ as a function of λ . Logistic regression on a1a dataset. See Appendix for experimental setup.	164
8.2	Communication rounds to get $\frac{F(x^k) - F(x^*)}{F(x^0) - F(x^*)} \leq 10^{-5}$ as a function of p with $p^* \approx 0.09$ (for L2SGD+). Logistic regression on a1a dataset with $\lambda = 0.1$; details in the Appendix.	167
8.3	Variance reduced local SGD (Algorithm 20), shifted local SGD (Algorithm 64) and local SGD (Algorithm 63) applied on LibSVM problems for both homogenous split of data and Heterogenous split of the data. Stepsize for non-variance reduced method was chosen the same as for the analogous variance reduced method.	169
8.4	Effect of the aggregation probability p (legend of the plots) on the convergence rate of Algorithm 20. Choice $p = p^*$ corresponds to red dotted line with triangle marker. Parameter λ was chosen in each case as Table 8.1 indicates.	170



8.5	Effect of parameter λ (legend of the plot) on the convergence rate of Algorithm 20. The choice $\lambda = \lambda^*$ corresponds to brown dash-dotted line with diamond marker (the third one from the legend). Aggregation probability p was chosen in each case as Table 8.1 indicates.	171
9.1	Comparison of CD with uniform sampling, CD with importance sampling, accelerated CD with importance sampling and SSCN (Algorithm 21) with uniform sampling on LibSVM datasets.	185
9.2	Comparison of coordinate descent, accelerated coordinate descent and SSCN (all with uniform sampling) on LibSVM datasets. In each case we have normalized the data matrix to have identical norms of all columns. .	186
9.3	SSCN vs. SDNA on LibSVM datasets. All algorithms with uniform sampling.	187
9.4	SSCN and Coordinate Descent (CD) methods, minimizing Log-Sum-Exp function, $d = 500$	189
9.5	SSCN and Coordinate Descent (CD) methods, minimizing Log-Sum-Exp function, $d = 1000$	190
10.1	Accelerated matrix inversion on synthetic data. From left to right: (i) Eigenvalues of $A \in \mathbb{R}^{100 \times 100}$ are $1, 10^3, 10^3, \dots, 10^3$ and coordinate sketches with convenient probabilities are used. (ii) Eigenvalues of $A \in \mathbb{R}^{100 \times 100}$ are $1, 2, \dots, n$ and Gaussian sketches are used. Label “nsym” indicates non-enforcing symmetry and “-a” indicates acceleration. (iii) Epsilon dataset ($n = 2000$), coordinate sketches with uniform probabilities. (iv) SVHN dataset ($n = 3072$), coordinate sketches with convenient probabilities. Label “h” indicates that λ_{\min} was not precomputed, but θ was chosen as described in the text.	202
10.2	Accelerated matrix inversion on synthetic data. Parameter choice: $\eta = 1 + 10^{-1}, \beta = -n^{-1}, n = 100$. From left to right we have: Coordinate sketch with uniform (convenient) probabilities and Gaussian sketch respectively.	203
10.3	Accelerated matrix inversion on synthetic data. Parameter choice: $\eta = 1 + 10^{-3}, \beta = -n^{-1}, n = 100$. From left to right we have: Coordinate sketch with uniform (convenient) probabilities and Gaussian sketch respectively.	204
10.4	Accelerated matrix inversion on synthetic data. Parameter choice: $\eta = 1 + 10^{-5}, \beta = -n^{-1}, n = 100$. From left to right we have: Coordinate sketch with uniform (convenient) probabilities and Gaussian sketch, respectively.	204
10.5	Eigenvalues set to $1, 2, 3, \dots, n$. From left to right we have: Coordinate sketch with convenient probabilities, coordinate sketch with uniform probabilities and Gaussian sketch respectively.	205
10.6	Eigenvalues set to $1, 10, 10, \dots, 10$. From left to right we have: Coordinate sketch with convenient probabilities, coordinate sketch with uniform probabilities and Gaussian sketch respectively.	205
10.7	Accelerated matrix inversion on synthetic data. Eigenvalues set to $1, 100, 100, \dots, 100$. From left to right we have: Coordinate sketch with convenient probabilities, coordinate sketch with uniform probabilities and Gaussian sketch respectively.	206

10.8 Accelerated matrix inversion on synthetic data. Eigenvalues set to 1, 1000, 1000, ..., 1000. From left to right we have: Coordinate sketch with convenient probabilities, coordinate sketch with uniform probabilities and Gaussian sketch respectively.	206
10.9 Accelerated matrix inversion on synthetic data. Eigenvalues set to 10000, 1, 1, ..., 1. From left to right we have: Coordinate sketch with convenient probabilities, coordinate sketch with uniform probabilities and Gaussian sketch respectively.	206
10.10 Accelerated matrix inversion on synthetic data. Eigenvalues set to 1, 2, ..., n . From left to right we have: Coordinate sketch with convenient probabilities, coordinate sketch with uniform probabilities and Gaussian sketch respectively.	207
10.11 Accelerated matrix inversion on synthetic data. Eigenvalues set to 1, 10, 10, ..., 10. Coordinate sketch with convenient probabilities, coordinate sketch with uniform probabilities and Gaussian sketch respectively. . . .	207
10.12 Accelerated matrix inversion on synthetic data. Eigenvalues set to 1, 100, 100, ..., 100. From left to right we have: Coordinate sketch with convenient probabilities, coordinate sketch with uniform probabilities and Gaussian sketch respectively.	207
10.13 Accelerated matrix inversion on synthetic data. Eigenvalues set to 1, 1000, 1000, ..., 1000. From left to right we have: Coordinate sketch with convenient probabilities, coordinate sketch with uniform probabilities and Gaussian sketch respectively.	208
10.14 Accelerated matrix inversion on synthetic data. Eigenvalues set to 10000, 1, 1, ..., 1. From left to right we have: Coordinate sketch with convenient probabilities, coordinate sketch with uniform probabilities and Gaussian sketch respectively.	208
10.15 Accelerated matrix inversion on real data. Dataset aloi: $n = 128$. From left to right we have: Coordinate sketch with convenient probabilities, coordinate sketch with uniform probabilities and Gaussian sketch respectively.	209
10.16 Accelerated matrix inversion on real data. Dataset w1a: $n = 300$. From left to right we have: Coordinate sketch with convenient probabilities, coordinate sketch with uniform probabilities and Gaussian sketch respectively.	209
10.17 Accelerated matrix inversion on real data. Dataset w2a: $n = 300$. From left to right we have: Coordinate sketch with convenient probabilities, coordinate sketch with uniform probabilities and Gaussian sketch respectively.	209
10.18 Accelerated matrix inversion on real data. Dataset mushrooms: $n = 112$. From left to right we have: Coordinate sketch with convenient probabilities, coordinate sketch with uniform probabilities and Gaussian sketch respectively.	210

10.19	Accelerated matrix inversion on real data. Dataset protein: $n = 357$. From left to right we have: Coordinate sketch with convenient probabilities, coordinate sketch with uniform probabilities and Gaussian sketch respectively.	210
10.20	Accelerated matrix inversion on real data. Dataset phishing: $n = 68$. From left to right we have: Coordinate sketch with convenient probabilities, coordinate sketch with uniform probabilities and Gaussian sketch respectively.	210
10.21	Accelerated matrix inversion on real data. Dataset madelon: $n = 500$. From left to right we have: Coordinate sketch with convenient probabilities, coordinate sketch with uniform probabilities and Gaussian sketch respectively.	211
10.22	Accelerated matrix inversion on real data. Dataset epsilon: $n = 2000$. From left to right we have: Coordinate sketch with convenient probabilities, coordinate sketch with uniform probabilities and Gaussian sketch respectively.	211
10.23	Accelerated matrix inversion on real data. Dataset svhn: $n = 3072$. From left to right we have: Coordinate sketch with convenient probabilities, coordinate sketch with uniform probabilities and Gaussian sketch respectively.	211
10.24	Accelerated matrix inversion on real data. Dataset gisette: $n = 5000$. From left to right we have: Coordinate sketch with convenient probabilities, coordinate sketch with uniform probabilities and Gaussian sketch respectively.	212
10.25	Accelerated matrix inversion on synthetic data. Sensitivity to acceleration parameters. Eigenvalues of A are set to $1, 2, \dots, n$. From left to right we have: Coordinate sketches with convenient probabilities, coordinate sketches with uniform probabilities and Gaussian sketches. Choice of parameters as per (10.16) in the middle of plots. Each instance was run for 5 seconds.	213
10.26	Accelerated matrix inversion on synthetic data. Sensitivity to acceleration parameters. Eigenvalues of A are set to $1, 10, 10, \dots, 10$. From left to right we have: Coordinate sketches with convenient probabilities, coordinate sketches with uniform probabilities and Gaussian sketches. Choice of parameters as per (10.16) in the middle of plots. Each instance was run for 2 seconds.	213
10.27	Accelerated matrix inversion on synthetic data. Sensitivity to acceleration parameters. Eigenvalues of A are set to $1, 1000, 1000, \dots, 1000$. From left to right we have: Coordinate sketches with convenient probabilities, coordinate sketches with uniform probabilities and Gaussian sketches. Choice of parameters as per (10.16) in the middle of plots. Each instance was run for 10 seconds.	214
10.28	Algorithm 24 (BFGS with accelerated matrix inversion quasi-Newton update) vs standard BFGS. Left column: time, right column: iteration. From top to bottom: phishing, mushrooms, australian and splice dataset.	215

10.29	Accelerated BFGS applied on real data. Left to right: <code>madelon</code> , <code>covtype</code> , <code>a9a</code>	215
I.1	Proof of $(1 + \frac{1}{c}) \frac{1}{2(1+c)^2} - \omega_*(\frac{1}{1+c}) \geq 0$ for all $c > 0$	405

LIST OF TABLES

1.1	Summary of representative algorithms proposed in each chapter and topics covered in each chapter. Columns (chapter topics): VR = variance reduced method, Accel = Nesterov's acceleration, Subsp = subspace descent, Prox = proximal setup, Distrib = distributed setup. Further clarifications: * ACD allows for subspaces spanned by standard basis vectors only; † these methods consider a general subspace oracle, but perform full dimensional updates; ‡ SSCN requires the regularizer ψ to be separable.	42
1.2	List of all algorithms stated in this work. Marker  indicates that the algorithm is new (i.e., proposed in this work) while marker  indicates that the algorithm is known.	43
2.1	Complexity results for non-accelerated (CD) and accelerated (ACD) coordinate descent methods for μ -strongly convex functions and arbitrary sampling S . The last row corresponds to the setup with arbitrary proper sampling S (i.e., a random subset of $[d]$ with the property that $p_i \stackrel{\text{def}}{=} \mathbb{P}(i \in S) > 0$). We let $\tau \stackrel{\text{def}}{=} \mathbf{E}[S]$ be the expected mini-batch size. We assume that f is \mathbf{M} -smooth (see (2.6)). The positive constants v_1, v_2, \dots, v_d are the ESO parameters (depending on f and S), defined in (2.7). The first row arises as a special of the third row in the non-minibatch (i.e., $\tau = 1$) case. Here we have $v_i = L_i \stackrel{\text{def}}{=} \mathbf{M}_{ii}$. The second row is a special case of the first row for the optimal choice of the probabilities p_1, p_2, \dots, p_d	52
2.2	New complexity results for ACD with minibatch size $\tau = \mathbb{E}[S^k]$ and various samplings (we suppress $\log(1/\epsilon)$ factors in all expressions). Constants: μ = strong convexity constant of f , $L = \lambda_{\max}(\mathbf{M})$, $\beta = (\tau - 1)/(d - 1)$, $1 \leq \gamma \leq \sqrt{d}$, and $\omega \leq \mathcal{O}(\sqrt{\tau})$ (ω can be as small as $\mathcal{O}(\tau/d)$).	56
2.3	Problem types for testing ACD.	59
3.1	Complexity results for coordinate descent (CD) and our sketched gradient method (SEGA), specialized to coordinate sketching, for \mathbf{M} -smooth and μ -strongly convex functions.	73
3.2	Spectrum of \mathbf{M}	77
4.1	Summary of all algorithms proposed in the chapter.	86
5.1	Selected special cases of GJS (Algorithm 14) arising by choosing operators S and \mathcal{U} in particular ways. R is a random subset of $[n]$, L is a random subset of $[d]$, $p_i = \mathbb{P}(i \in L)$, $p_j = \mathbb{P}(j \in R)$	116

5.2	Four types of quadratic problems. We choose $u \sim N(0, \mathbf{I}_d)$, and γ to be such that $\ \gamma \mathbf{M}^{-1} u\ = \frac{3}{2}$. Notation $c^{[d]}$ stands for a vector (c, c^2, \dots, c^d) .	118
5.3	Table of LibSVM data used for our experiments.	121
6.1	List of specific existing (in some cases generalized) and new methods which fit our general analysis framework. VR = variance reduced method, AS = arbitrary sampling, Quant = supports gradient quantization, RCD = randomized coordinate descent type method. ^a Special case of SVRG with 1 outer loop only; ^b Special case of DIANA with 1 node and quantization of exact gradient.	133
6.2	The parameters for which the methods from Table 6.1 (special cases of (6.5)) satisfy Assumption 6.3.1. The meaning of the expressions appearing in the table, as well as their justification is defined in detail in the Appendix (Section F.1).	134
6.3	Four types of least squares.	138
7.1	Choice of \mathbf{M} . O_{dd} is set of all odd positive integers smaller than $d + 1$, while matrix \mathbf{U} was set as random orthonormal matrix (generated by QR decomposition from a matrix with independent standard normal entries).	152
8.1	Setup for the experiments.	167
A.1	Summary of frequently used notation.	244
A.2	Summary of frequently used notation specific to Chapter 3.	245
A.3	Summary of frequently used notation specific to Chapter 4.	245
A.4	Summary of frequently used notation specific to Chapter 5.	246
A.5	Summary of frequently used notation specific to Chapter 9.	247
E.1	Iteration complexity of selected special cases of GJS (Algorithm 14). Whenever m appears in a result, we assume that $\mathbf{M}_j = m\mathbf{I}_d$ for all j (i.e., f_j is m -smooth). Whenever m_i appears in a result, we assume that f is \mathbf{M} -smooth with $\mathbf{M} = \text{Diag}(m_1, \dots, m_d)$. Whenever m_i^j appears in a result, we assume that $\mathbf{M}_j = \text{Diag}(m_1^j, \dots, m_d^j)$. Quantities p_i for $i \in [d]$, p_j for $j \in [n]$, ρ and δ are probabilities defining the algorithms.	304

Chapter 1

Introduction

Over the past several decades, optimization has become a key tool in the toolbox of modern technology, enabling a multitude of areas of engineering, computer science, physics, economics, finance, chemistry, computational biology, as well as many other fields of human endeavor.

In this thesis, we predominantly focus on continuous optimization problems arising in the training of supervised machine learning models¹. Informally, the training of such models can be described as the search for the parameters characterizing the model that best fits the observed data. In particular, the dominant paradigm for solving supervised machine learning problems is to cast them as *regularized empirical risk minimization* (ERM) problems, often also called *finite-sum* optimization problems, which take the form

$$\min_{x \in \mathbb{R}^d} \left\{ F(x) \stackrel{\text{def}}{=} \underbrace{\frac{1}{n} \sum_{i=1}^n f_i(x)}_{\stackrel{\text{def}}{=} f(x)} + \psi(x) \right\}. \quad (1.1)$$

In the above problem, the vector $x \in \mathbb{R}^d$ represents the parameters describing the model we wish to train (e.g., support vector machine, logistic regression or a neural network), the function f_i measures the misfit of model x with respect to the i th data point, and function $\psi : \mathbb{R}^d \rightarrow \mathbb{R} \cup \{+\infty\}$ is a regularizer whose role is to incorporate prior information or impart desirable properties onto the model. The objective function F measures the (regularized) empirical loss of model x .

The training of machine learning models carries a multitude of challenges, with the two most pronounced being the size of the training dataset (i.e., big n) and the size of the model (i.e., big d). *Big data* and *big model* scenarios render standard deterministic optimization methods, such as gradient descent and Newton's method, inefficient at solving (1.1). In the past decade, this led to a “Cambrian explosion” of new iterative algorithms utilizing *randomness* in various ingenious ways aimed at addressing the big data and big model problems. In order to identify a model of suitable (optimization or generalization) properties, these new randomized methods typically rely on significantly cheaper iterations than their deterministic counterparts at the cost of requiring many more iterations. However, the benefits of such an approach often vastly outweigh the

¹Our results are applicable beyond supervised machine learning (training of regression/classification models), as we shall see. Supervised machine learning is, however, the primary application we have in mind.

costs, both in theory and in practice, which makes them the methods of choice in the big data or big model regime. The per-iteration savings are due to the inclusion of suitable randomization strategies such as *subsampling the data*, i.e., working with a small subset of the functions f_i in each iteration only, or *subsampling the parameters*, i.e., updating a small subset of the parameters in each iteration only.

Informally speaking, the main goal of this thesis is to develop, under appropriate assumptions on the properties of the regularized empirical loss function F , through its constituents $\{f_i\}_{i=1}^n$ and ψ , *new state-of-the-art randomized optimization algorithms* for solving the ERM problem (1.1), both in theory (by establishing improved convergence and complexity results) and in practice (by extensive experimental testing on synthetic and real data). While the structure of F varies slightly among the individual chapters of this thesis, we mostly assume that f is differentiable and convex, while ψ is convex, possibly non-smooth, but assumed to be *proximable*².

1.1 Technical preliminaries and basic algorithms

In this section, we introduce typical assumptions that we impose on the functions $\{f_i\}$ and ψ appearing in (1.1) throughout the individual chapters, as well as introduce standard tricks and results in optimization which we build upon in this work. We describe gradient descent—the cornerstone of first order optimization—followed by three standard tricks from the literature that gradient descent can be furnished with: Nesterov’s acceleration [149], proximal operator [9] and randomness [179].

We shall first equip \mathbb{R}^d with an inner product and a norm. The standard Euclidean inner product of vectors $x, y \in \mathbb{R}^d$ is $\langle x, y \rangle \stackrel{\text{def}}{=} \sum_{i=1}^d x_i \cdot y_i$ and the (induced) Euclidean norm is $\|x\| \stackrel{\text{def}}{=} \langle x, x \rangle^{1/2}$. For the reader’s convenience, we present a table of frequently used notation in Appendix A.

1.1.1 Smoothness and convexity

We now introduce two key concepts which will be used in various places throughout this text: *convexity* and *smoothness*. We will often assume that the objective F (or some part of F) is convex and smooth. The exact assumptions used differ from chapter to chapter, and are described therein. Let us first start with (strong) convexity.

Definition 1.1.1 (Strong convexity and convexity). *Let $\mu \geq 0$. Function $h : \mathbb{R}^d \rightarrow \mathbb{R}$ is μ -strongly convex if for all $x, y \in \mathbb{R}^d$, and all $t \in [0, 1]$:*

$$h(tx + (1-t)y) \leq th(y) + (1-t)h(x) - \frac{\mu t(1-t)}{2} \|x - y\|^2.$$

In the special case where $\mu = 0$, we say that h is convex.

The following standard result states that for a sufficiently smooth function h , strong convexity provides a global quadratic (or linear in the $\mu = 0$ case) lower bound on h and a uniform lower bound on the eigenvalues of its Hessian.

²The well-known notion of “proximability” is formally introduced in Section 1.1.4.

Proposition 1.1.2 (Nesterov [154]). *Let $h : \mathbb{R}^d \rightarrow \mathbb{R}$ be differentiable. Then, h is μ -strongly convex if and only if for all $x, y \in \mathbb{R}^d$:*

$$h(x) \geq h(y) + \langle \nabla h(y), x - y \rangle + \frac{\mu}{2} \|x - y\|^2.$$

If h is further twice differentiable, it is μ -strongly convex if and only if for all $x \in \mathbb{R}^d$ we have $\nabla^2 h(x) \succeq \mu \mathbf{I}$, where $\mathbf{I} \in \mathbb{R}^{d \times d}$ is the identity matrix.

Next, we introduce a typical *smoothness* assumption we make throughout the thesis.

Definition 1.1.3 (L -smoothness). *Differentiable function $h : \mathbb{R}^d \rightarrow \mathbb{R}$ is L -smooth if it has L -Lipschitz gradient, namely for all $x, y \in \mathbb{R}^d$:*

$$\|\nabla h(x) - \nabla h(y)\| \leq L\|x - y\|.$$

Analogously to strong convexity, smoothness provides us with both an upper bound on the function value as well as with an upper bound on the Hessian at each point in the domain.

Proposition 1.1.4 (Nesterov [154]). *A differentiable function $h : \mathbb{R}^d \rightarrow \mathbb{R}$ is L -smooth if and only if for all $x, y \in \mathbb{R}^d$:*

$$h(y) + \langle \nabla h(y), x - y \rangle - \frac{L}{2} \|x - y\|^2 \leq h(x) \leq h(y) + \langle \nabla h(y), x - y \rangle + \frac{L}{2} \|x - y\|^2. \quad (1.2)$$

If h is further twice differentiable, it is L -smooth if and only if for all $x \in \mathbb{R}^d$ we have $\nabla^2 h(x) \preceq L\mathbf{I}$, where \preceq designates the Löwner ordering of matrices.

We are now ready to present the backbone of the world of first-order optimization algorithms—gradient descent—along with a few basic and well known extensions.

1.1.2 Gradient descent

For the sake of expositional simplicity, consider optimization problem (1.1) in its simplest form: $\psi \equiv 0$ and $n = 1$. That is, we consider the unregularized case and ignore the finite-sum structure of f . In this case, $F = f$.

Note that if f is L -smooth, the second inequality in (1.2) provides us with a global convex quadratic upper bound on f using zero and first-order information about f at arbitrary point y :

$$f(x) \leq f(y) + \langle \nabla f(y), x - y \rangle + \frac{L}{2} \|x - y\|^2. \quad (1.3)$$

Minimizing this upper bound in the variable x gives

$$x = y - \frac{1}{L} \nabla f(y).$$

Doing this iteratively, we arrive at the famous *gradient descent* method (Algorithm 1), which is a trivial baseline we build on throughout this thesis.

Algorithm 1 Gradient descent (GD)

```

1: Input: Starting point  $x^0 \in \mathbb{R}^d$ , smoothness constant  $L > 0$ 
2: for  $k = 0, 1, 2, \dots$  do
3:    $x^{k+1} = x^k - \frac{1}{L} \nabla f(x^k)$ 
4: end for

```

Convergence properties of gradient descent are described in Proposition 1.1.5. This standard result posits sublinear convergence for the class of smooth and convex functions and linear convergence for the class of smooth and strongly convex functions.

Proposition 1.1.5 (Nesterov [154]). *Let $f^* \stackrel{\text{def}}{=} \min_{x \in \mathbb{R}^d} f(x)$ and $x^* \stackrel{\text{def}}{=} \arg \min_{x \in \mathbb{R}^d} F(x) = \arg \min_{x \in \mathbb{R}^d} f(x)$. Suppose that the sequence of iterates $\{x^k\}_{k=0}^\infty$ is generated by Algorithm 1. If f is L -smooth and convex, then*

$$f(x^k) - f^* \leq \frac{2L \|x^0 - x^*\|^2}{k + 4}.$$

If we additionally assume that f is μ -strongly convex³, then

$$f(x^k) - f^* \leq \left(\frac{L - \mu}{L + \mu} \right)^{2k} \frac{L \|x^0 - x^*\|^2}{2}.$$

Next, we introduce a handful of tricks that can be incorporated on top of gradient descent: *Nesterov's acceleration*, *proximal operator*, and *stochasticity*. As we shall see, these tricks are mutually “orthogonal”, which means that, generally speaking, they can be built on top of each other for a more pronounced additive benefit.

1.1.3 Nesterov's acceleration

Notice that gradient descent is a *greedy* method. Indeed, the next iterate is constructed to find a point with the smallest guaranteed function value given the information we have about f : zero and first-order information about f at the current iterate, and the smoothness parameter L . As a byproduct, gradient descent forgets all the past information gathered throughout the optimization process. It turns out that in this case, greediness as an algorithmic design tool is suboptimal since appropriate use of history can yield to a significant improvement in iteration complexity. Nesterov's *accelerated gradient descent* method (stated as Algorithm 2) is an algorithm that achieves this.

The following proposition describes the convergence rate of Nesterov's accelerated gradient descent method.

Proposition 1.1.6 (Nesterov [154, 149]). *Suppose that sequence $\{x^k\}_{k=0}^\infty$ was generated*

³This implies that, necessarily, $\mu \leq L$.

Algorithm 2 Nesterov's accelerated gradient descent (AGD)

```

1: Input: Starting point  $x^0 = y^0 \in \mathbb{R}^d$ , smoothness constant  $L > 0$ , strong convexity
    $\mu \geq 0$ 
2: for  $k = 0, 1, 2, \dots$  do
3:    $x^{k+1} = y^k - \frac{1}{L} \nabla F(y^k)$ 
4:   if  $\mu = 0$  then
5:      $y^{k+1} = x^{k+1} + \frac{k}{k+3} (x^{k+1} - x^k)$ 
6:   else
7:      $y^{k+1} = x^{k+1} + \frac{\sqrt{L} - \sqrt{\mu}}{\sqrt{L} + \sqrt{\mu}} (x^{k+1} - x^k)$ 
8:   end if
9: end for

```

by Algorithm 2. If f is L -smooth and μ -strongly convex⁴, then

$$f(x^k) - f^* \leq \min \left\{ \left(1 - \sqrt{\frac{\mu}{L}}\right)^k, \frac{4}{(2+k)^2} \right\} \left(f(x^0) - f^* + \frac{L}{2} \|x^0 - x^*\|^2 \right).$$

Up to a constant factor, the method only requires a square root of the number of iterations needed by gradient descent.

It is important to mention that Algorithm 2 is *optimal* in terms of oracle complexity for both smooth convex and smooth strongly convex problems as it (up to a constant) matches the corresponding lower bound [154].

1.1.4 Proximal operator and proximal gradient descent

In their simplest form, neither gradient descent nor Nesterov's accelerated gradient descent are applicable in the presence of a non-smooth regularizer ψ . In this section, we offer a brief overview of the proximal gradient descent method, which is capable of solving (1.1) for any convex closed⁵ provided that ψ is *proximable*, which means that the *proximal operator* of ψ , defined as

$$\text{prox}_{\alpha\psi}(x) \stackrel{\text{def}}{=} \arg \min_{y \in \mathbb{R}^d} \left\{ \psi(y) + \frac{1}{2\alpha} \|y - x\|^2 \right\}, \quad (1.4)$$

where $\alpha > 0$, is easily computable (e.g., in closed form).

Example 1. In the following two examples we give formulas for the proximal operators of two commonly used regularizers.

- In some applications, ψ is used to represent a hard constraint on model x . In particular, let $Q \subseteq \mathbb{R}^d$ be any nonempty closed convex set. It is easy to see that the optimization problem

$$\min_{x \in Q} f(x)$$

⁴We allow for $\mu = 0$.

⁵We say that a convex function $h : \mathbb{R}^d \rightarrow \mathbb{R} \cup \{+\infty\}$ is closed if for any $c \in \mathbb{R}$ the sublevel set $\{x; | h(x) \leq c\}$ is a closed set.

can be equivalently written in the form (1.1) by setting ψ to be the “indicator” function of Q :

$$\min_{x \in \mathbb{R}^d} f(x) + \psi(x), \quad \psi(x) \stackrel{\text{def}}{=} \begin{cases} 0 & \text{if } x \in Q \\ \infty & \text{if } x \notin Q \end{cases}.$$

Consequently, the proximal operator of ψ becomes the projection operator onto Q , i.e.,

$$\forall \alpha > 0 : \quad \text{prox}_{\alpha\psi}(x) = \arg \min_{y \in Q} \|x - y\|^2.$$

- In applications where one prefers a sparse solution x^* , one can set ψ to be the sparsity-inducing ℓ_1 norm: $\psi(x) = \|x\|_1 \stackrel{\text{def}}{=} \sum_{i=1}^d |x_i|$. In a such case, the proximal operator of ψ is equivalent to applying elementwise soft-thresholding; i.e.,

$$\forall \alpha \geq 0 : \quad (\text{prox}_{\alpha\psi}(x))_i = \begin{cases} 0 & \text{if } |x_i| \leq \alpha \\ \text{sign}(x_i)(|x_i| - \alpha) & \text{if } |x_i| > \alpha \end{cases}$$

for all $i \in \{1, 2, \dots, d\}$.

Clearly, in both these examples, it is not possible to apply gradient descent to minimize $F = f + \psi$ since F is not differentiable due to the presence of ψ . While it is possible to replace gradient with subgradient⁶—resulting in the *subgradient method*—such an approach suffers from inferior convergence guarantees [154, 69].

Alternatively, we might take advantage of the proximability of ψ and incorporate the proximal operator into the optimization procedure. The most natural approach is to alternate the gradient step with the proximal step, which results in the *proximal gradient descent* method (PGD) [9, 8]. If the regularizers considered in the above example are used, the method is often alternatively known under the name *projected gradient descent* and ISTA, respectively.

Algorithm 3 Proximal gradient descent (PGD)

- 1: **Input:** Starting point $x^0 \in \mathbb{R}^d$, smoothness constant $L > 0$
 - 2: **for** $k = 0, 1, 2, \dots$ **do**
 - 3: $x^{k+1} = \text{prox}_{\frac{1}{L}\psi} \left(x^k - \frac{1}{L} \nabla f(x^k) \right)$
 - 4: **end for**
-

The following proposition describes the convergence rate of proximal gradient descent (Algorithm 3). The method is, up to a small constant factor, as fast as gradient descent. Consequently, incorporating the regularizer ψ into the optimization method does not hurt the convergence rate. In some cases, the presence of ψ might make optimization easier; we will elaborate on this soon.

⁶Informally speaking, subgradient is a generalization of the gradient for convex, possibly non-differentiable functions.

Proposition 1.1.7 (Beck [8]). *Let $F^* \stackrel{\text{def}}{=} \min_{x \in \mathbb{R}^d} F(x)$ and $x^* \stackrel{\text{def}}{=} \arg \min_{x \in \mathbb{R}^d} F(x)$. Suppose that the sequence $\{x^k\}_{k=0}^\infty$ is generated by Algorithm 3. If f is L -smooth and convex, we have*

$$F(x^k) - F^* \leq \frac{L \|x^0 - x^*\|^2}{2k}.$$

If further f is μ -strongly convex (with $\mu > 0$), we have

$$\|x^k - x^*\|^2 \leq \left(1 - \frac{\mu}{L}\right)^k \|x^0 - x^*\|^2.$$

1.1.5 Incorporating randomness

All of the optimization algorithms introduced so far are agnostic to the finite-sum structure of f . Consequently, if f_i measures the misfit of the current model x at the i th datapoint, both GD and AGD are passing through the entire dataset every iteration. The larger the number of datapoints n is, the more expensive it is to perform an iteration of these methods, which makes them impractical.

How can one effectively deal with big n then? The most natural approach is simply to replace the expensive computation of the (full) gradient of f ,

$$\nabla f(x) = \frac{1}{n} \sum_{i=1}^n \nabla f_i(x),$$

via a cheap stochastic approximation thereof, resulting in the celebrated *stochastic gradient descent* (SGD) method [179], which we state as Algorithm 4.

For simplicity of exposition, we have once again adopted the assumption that $\psi \equiv 0$. However, as we shall show later, the same result holds in the regularized case by incorporating the proximal operator into the algorithm.

Algorithm 4 Stochastic gradient descent (SGD)

- 1: **Input:** Starting point $x^0 \in \mathbb{R}^d$, stepsize $\alpha > 0$
 - 2: **for** $k = 0, 1, 2, \dots$ **do**
 - 3: Sample $i \in \{1, 2, \dots, n\}$ uniformly at random
 - 4: $x^{k+1} = x^k - \alpha \nabla f_i(x^k)$
 - 5: **end for**
-

Convergence rate of SGD is presented in Proposition 1.1.8. In particular, under smoothness and strong convexity assumptions, SGD enjoys a fast, linear rate to a specific neighborhood of the optimum.

Proposition 1.1.8 (Nguyen et al [159], Gower et al [60]). *Suppose that function f_i is L -smooth and convex for all i , while function f is μ -strongly convex with $\mu > 0$. Then, for any $\alpha \leq \frac{1}{2L}$ we have*

$$\mathbb{E} [\|x^k - x^*\|^2] \leq (1 - \alpha\mu)^k \|x^0 - x^*\|^2 + \frac{2\alpha \sum_{i=1}^n \|\nabla f_i(x^*)\|^2}{n\mu}.$$

Note that if the gradients $\nabla f_i(x^*)$ are all zero, which typically happens for over-parameterized models, the above result posits a linear convergence rate to the optimal solution x^* . In general, the right hand side in the complexity guarantee can be made arbitrarily small by choosing the stepsize α sufficiently small and k sufficiently large. Alternatively, this can be achieved by choosing a suitable decreasing stepsize schedule. However, such adjustments will lead to a worse convergence rate: we get a $\tilde{O}(1/k)$ rate towards the true optimum. It is possible to preserve the linear rate even if the gradients at the optimum are not zero, but for this to happen, one needs to adjust SGD to employ one of the many *variance-reduction* techniques proposed in the literature.

1.2 From finite sum to coordinate descent and back

Let us consider a very specific form of the objective (1.1): assume that for all i , function f_i corresponds to a loss of a linear model,⁷ i.e., $f_i(x) = \phi_i(\langle a_i, x \rangle)$ for some convex $\phi : \mathbb{R} \rightarrow \mathbb{R}$ and $a_i \in \mathbb{R}^d$, while $\psi \equiv 0$. The considered objective thus becomes:

$$\min_{x \in \mathbb{R}^d} \left\{ F(x) = f(x) = \frac{1}{n} \sum_{i=1}^n \phi_i(\langle a_i, x \rangle) \right\}.$$

Until very recently, such models were predominantly optimized using standard deterministic methods such as gradient descent, accelerated gradient/FISTA or Newton's method. As already mentioned, these classical methods require an evaluation of $\nabla f(x^k)$ at every iteration.⁸ Consequently, the deterministic methods have to evaluate the dot product $\langle a_i, x \rangle$ for all $i \in \{1, 2, \dots, n\}$ in each step, and thus are either very expensive and may even be infeasible in the big data setting (i.e., when n, d are large).⁹ The demand for solving such big data problems resulted in the development of algorithms working with a small random subset of the training data in each iteration only.

The most natural approach is to use SGD (i.e., subsample the finite sum) as described in Section 1.1.5. The main idea of SGD is to in each iteration pick a random index j , $1 \leq j \leq n$, and move the current iterate x in the direction of the stochastic gradient

$$\nabla f_j(x) = a_j \phi'_j(\langle a_j, x \rangle). \quad (1.5)$$

While the cost of performing a simple SGD iteration is often $\mathcal{O}(d)$ only, SGD is slow in terms of how many iterations are required to get to an ϵ -neighborhood to the optimum (where ϵ is relatively small). In particular, the stochastic gradient estimator (1.5) has a (non-zero) variance at the optimum, causing SGD to be gradually slower over time. Consequently, SGD either converges linearly to a neighborhood of the solution only (Proposition 1.1.8), or converges sublinearly to the true optimum using a decreasing stepsize policy.

Fortunately, the issue of sublinear convergence of SGD has been resolved using a more

⁷Our results go beyond linear models. The assumption is made here in order to provide a simple motivation for coordinate descent methods.

⁸Newton's method requires the computation of $\nabla^2 f(x^k)$ on top of that.

⁹In particular, the cost of performing a single iteration is $\mathcal{O}(nd)$ for (accelerated) gradient descent and $\mathcal{O}(nd^2 + d^3)$ for Newton's method.

sophisticated stochastic gradient estimator whose variance progressively diminishes as $x^k \rightarrow x^*$. Methods based on such sophisticated estimators are commonly known as *variance reduced algorithms*; the most famous among them are SAG [182], SAGA [37], SDCA [191], SVRG [88], S2GD [101], Finito [38], MISO [133], QUARTZ [169] and SARAH [160]. SAGA and SVRG achieve the variance reduction property by incorporating *control variates* [82] into the stochastic gradient – we will exploit this idea multiple times throughout this text.

1.2.1 From finite sum to coordinate descent

An orthogonal approach to subsampling the finite sum is to subsample the domain (parameter space) and use (Randomized) Coordinate Descent (CD) [152]. In its most basic form, CD samples a random index i (where $1 \leq i \leq d$) and updates the i th coordinate of the current iterate x in the direction of $\nabla_i f(x)e_i$, where $\nabla_i f(x)$ is the i th partial derivative of f at x and e_i is i th standard basis vector. Unlike SGD, CD does not suffer from the intrinsic variance at the optimum. At the same time, a single iteration of CD can¹⁰ be implemented in time $\mathcal{O}(n)$ and consequently, CD is a serious competition to variance reduced SGD algorithms. To decide which approach is superior to solve a given problem is rather complex [30]. However, the general rule of thumb suggests to use CD if $d > n$ and variance reduced SGD if $n > d$.

The above described, most straightforward version of CD, is still fairly inefficient. Firstly, it is suboptimal in terms of iteration complexity;¹¹ one shall combine it with Nesterov’s acceleration as per [166, 7, 157]. Secondly, currently used hardware often allows evaluating a subset of partial derivatives in parallel almost as fast as a single partial derivative. This leads to the need to develop a tight theory of CD methods under arbitrary sampling of the subsets¹² to allow the user to tune CD for his/her own specific hardware [166]. However, those two CD adjustments were never combined before and this is where the story of this thesis starts. In particular, in a part of Chapter 2, we propose an accelerated CD method with arbitrary sampling (ACD).

One of the main disadvantages of CD methods over SGD algorithms is that they do not allow for a proximal regularizer ψ that is non-separable.¹³ In particular, non-separable ψ prevents attainment of a linear convergence rate for CD as the corresponding stochastic gradient estimator suffers from the inherent (non-zero) variance at the optimum, which very much resembles the story of SGD. Since the mechanism of variance reduction has already successfully “fixed” the issue for SGD, one might ask whether it is possible to incorporate an analogous trick into CD methods. Fortunately, we were successful: in Chapter 3, we propose a new randomized algorithm—SEGA—which accesses only a block of partial derivatives of f each iteration and still converges linearly to the solution despite the presence of a *non-separable* regularizer ψ . This is the first variance-reduced CD method in the literature.

¹⁰The trick lies in the memorization of the dot products $\langle a_i, x \rangle$, see [152] for details.

¹¹Total number of iteration to reach ε -solution.

¹²I.e., we wish to give as tight rate as possible for any given probability distribution over all subsets of $\{1, 2, \dots, d\}$ and corresponding sampling strategy for CD.

¹³We say that a function h is separable if it can be written as $h(x) = \sum_{i=1}^d h_i(x_i)$.

1.2.2 From coordinate descent to finite sum: three approaches

The development of SEGA provided us with many insights and ideas for future research. It brought us back to the finite-sum minimization in three somewhat independent ways, which we describe next.

Distributed optimization and random sparsification

In many applications, the scale of the problem we are solving is so large that the dataset does not fit into the memory of a single machine. Consequently, multiple machines need to be employed to both store the data and train the model. In this thesis, we consider a specific, centralized case of distributed optimization/learning, where the machines are not allowed to communicate directly among themselves, but instead are allowed to communicate with a central server/master, also known as *parameter server*.

Note that the optimization problem (1.1) provides convenient notation for the mentioned scenario: function f_i might represent a loss of the model on data owned by i th machine. In such a case, the value of n corresponds to the number of machines/workers instead of the size of the dataset.

Distributed optimization brings up several new challenges that are not present in standard optimization. Specifically, the communication between the workers and the parameter server/master takes a non-trivial time, often much more than the computation itself. There are several different ways to reduce communication complexity of gradient-type methods, one of which is *gradient sparsification*. Specifically, in order to communicate some non-sparse gradient $\nabla f_i(x) \in \mathbb{R}^d$, one should send d real numbers (often this is $32d$ or $64d$ bits). In contrast, to communicate a randomly sparsified gradient $\nabla_{j_i} f_i(x) e_{j_i}$, where $1 \leq j_i \leq d$ is selected uniformly at random, we only need to send a single real number along with its position, which is at least $d/2$ times cheaper in practice.

The major drawback of random sparsification is that the estimator $g(x)$ of $\nabla f(x)$ constructed as a naive aggregation of sparsified gradients from the workers

$$g(x) \stackrel{\text{def}}{=} \frac{1}{n} \sum_{i=1}^n \nabla_{j_i} f_i(x) e_{j_i}$$

is very noisy, and its variance does not diminish as the method progresses through its iterations. Indeed, $g(x)$ has a non-zero variance at the optimum. In Chapter 4, we incorporate control variates (similarly to SEGA) on top of the sparsified gradient, which enables us to eliminate the adverse effect of the variance at the optimum on the convergence rate. Consequently, we show that our method can reduce worker→server communication by as much as the factor of n without hurting the convergence rate by more than a small constant. To illustrate the scale of this effect, consider a setup with 100 workers. In this case, we prove that only 1% of the usual worker→server communication is needed to preserve the fast convergence rate.

Unification of algorithms

Since the variance reduced methods in three different setups (classical finite sum, Chapter 3, and Chapter 4) share certain intrinsic similarities, one may wonder whether it is possible to unify them in a single algorithm, admitting a single analysis, so that one would not have to keep developing novel variance reduced algorithms along with their analyses from scratch. In Chapter 5, we propose a general method—GJS (Generalized Jacobian Sketching)—which constructs a gradient estimator given that a randomized linear transformation (a *sketch*) of the Jacobian matrix

$$\mathbf{G}(x) \stackrel{\text{def}}{=} [\nabla f_1(x), \nabla f_2(x), \dots, \nabla f_n(x)] \in \mathbb{R}^{d \times n}$$

is observed in each iteration. The sketch is allowed to follow an arbitrary fixed distribution, and in special cases includes right matrix multiplication (in such a case we can recover SAGA or SVRG), and left matrix multiplication (in such a case we can recover SEGA). many more sketches are possible, which gives rise to novel method not considered in literature before. This work is the first unification of stochastic optimization algorithms which subsample the finite sum, such as SAGA, and algorithms which subsample the parameters, such as SEGA. Our theory gives the currently best-known convergence rate in each special case, and also allows for the development of importance sampling rates that exploit the smoothness structure of the objective.

We did not stop here, the story of this thesis unfolds further.

Our findings made us realize that we can go one step further in terms of generality. In particular, the analysis of variance reduced SGD algorithms and non-variance reduced SGD shares a number of similar steps that can be abstracted to a *unified analysis framework*, which is what we do in Chapter 6. We provide a convergence rate for SGD given that the unbiased stochastic gradient g^k at iteration k satisfies the novel general parametric bound

$$\mathbb{E} \left[\|g^k - \nabla f(x^*)\|^2 \mid x^k, \sigma_k^2 \right] \leq 2A(f(x^k) - f(x^*)) + B\sigma_k^2 + D_1,$$

where $A, B, D_1 \in \mathbb{R}$ are some nonnegative constants, while the sequence of nonnegative random variables numbers σ_k satisfies

$$\mathbb{E} [\sigma_{k+1}^2 \mid x^k, \sigma_k^2] \leq (1 - \rho)\sigma_k^2 + 2CD_f(x^k, x^*) + D_2$$

for some nonnegative constants $\rho \leq 1, C, D_2$. Remarkably, the above inequalities enable us to analyze SGD, variance reduced methods for both finite sum and subspace gradients (i.e., SAGA and SEGA), quantized methods [136], and to develop and analyze several new algorithms of intriguing properties. Specifically, we introduce quantized methods with arbitrary sampling, partially variance reduced algorithms and an efficient, with-replacement importance sampling for minibatch SGD.

Both of the above-mentioned frameworks have many different applications besides recovering well-known algorithms. In particular, we have noticed an application in *federated learning*, which we describe in Chapter 8: Local SGD method (LSGD) with imperfect aggregation can be seen as (non-uniform) SGD applied to a carefully constructed 2-sum

objective that we introduce. The corresponding variance reduced algorithm we propose (a special case of GJS) achieves a linear rate which does not rely on the assumption of data homogeneity, and is favorable to classical variants of local SGD in terms of the convergence speed and communication complexity. Besides the importance of the newly proposed objective from the modeling perspective, our results suggest that the celebrated LSGD method should better be seen as minimizing our objective than the classical finite sum, which explains the difficulties in the standard analysis of LSGD, and reveals that the method implicitly aims to find *personalized* models.

Product space objective

Having previously discovered variance reduced variants of CD methods, and their subspace generalizations, we realized that there is a *new deep connection* between these methods and modern variance reduced methods for finite sum minimization. Specifically, we found that subspace VR algorithms are *more general* than finite sum VR algorithms: *applying subspace VR methods (SEGA) to minimize a particular product space (i.e., in the domain of \mathbb{R}^{nd}) objective is equivalent to applying SAGA to minimize arbitrary finite sum objective*. In order to obtain the best-known convergence rate of SAGA from SEGA, we had to tighten SEGA theory to take advantage of the structure of the non-smooth function ψ . As a by-product, we have improved upon the rate of GJS as well. More details are provided in Chapter 7.

1.2.3 Towards better stochastic condition numbers

The iteration complexity of each proposed algorithm in this work is determined by the so-called *stochastic condition number*, which is itself a function of the objective smoothness, strong convexity, and randomness of the algorithm. Thus a natural question arises: what is the best possible stochastic condition number, assuming that the source of stochasticity is fixed (we want to have the freedom to develop arbitrary stochastic algorithm)? Intuitively speaking, the stochastic condition number is non-decreasing in the smoothness and non-increasing in the strong convexity parameters, and is minimized if these parameters are equal. Such a setting corresponds to a quadratic objective, where both the smoothness and strong convexity are measured with respect to the same Euclidean norm, given via the Hessian of the objective.

Therefore, minimizing a general convex objective should not be simpler than minimizing the corresponding quadratic. We can now ask the reverse question: is there an algorithm which can minimize a non-quadratic convex objective with the same rate as if the function was in fact quadratic, with its Hessian being the Hessian of the non-quadratic function at the optimum? In Chapter 9, we provide an affirmative answer: we develop a second-order¹⁴ subspace descent method—SSCN (Stochastic Subspace Cubic Newton)—capable of achieving so. In particular, the local convergence rate of SSCN matches the rate of stochastic subspace descent applied to the problem of minimizing the quadratic

¹⁴I.e., the method is allowed to access second derivatives of the objective.

function

$$x \mapsto \frac{1}{2}(x - x^*)^\top \nabla^2 f(x^*)(x - x^*),$$

where x^* is the minimizer of f .

However, SSCN does not achieve the optimal stochastic convergence rate as it does not incorporate Nesterov's momentum or another acceleration mechanism. In Chapter 10, we introduce an accelerated sketch-and-project¹⁵ method with a superior rate to its non-accelerated counterpart developed by Gower and Richtárik [61]. In particular, besides direct applications we elaborate on in the text, the fast rate from Chapter 10 may also serve as an ambitious goal for the local rates of stochastic higher-order methods.

1.3 Relationship among the chapters

Section 1.2 describes how the chapters of this thesis were developed historically, outlining the chain of thought that led from one project to another.¹⁶ In this section, we elaborate on some non-historical connections among the chapters.

High-level picture: A step towards the optimization utopia

In the utopian optimization universe, a complexity¹⁷ would be known for any algorithm applied to solve any optimization problem. Such a knowledge would enable the practitioners to always apply an ideal algorithm given the problem to be solved and the complexity of the interest. This thesis presents a multiple steps towards the optimization utopia:

- **We fill the missing gaps** in the current literature in terms of the tightening best-known theory of well-established algorithms (Chapters 5, 7, 8), generalizing/extending the well-established algorithms (Chapters 2, 5, 6) and proposing a brand-new methods (Chapters 3, 4, 5, 6, 7, 8, 9, 10).
- **We establish novel and often surprising connections** between various algorithms, providing a better understanding of the optimization field (Chapters 5, 6, 7, 8).
- **We unify and generalize** both the known and the newly introduced algorithms, allowing to tailor the randomized optimization strategy for a broad range of different applications (Chapters 5, 6).

Next, we describe specific topics that the thesis chapters focus on.

¹⁵Sketch-and-project is a general stochastic method to minimize quadratic objective that recovers subspace descent in a special case.

¹⁶With one exception – Chapter 10 was developed before everything else.

¹⁷A complexity in a broader sense, for example the number of gradient evaluations, number of communication rounds, number of flops, or any other value of the interest.

Self-variance reduced methods, sublinear rates and control variates

The algorithms proposed in this thesis can be categorized based on their relation to control variates into three different classes:

- **Fast stochastic algorithms that do not require the aid of control variates.** This category includes ACD (Chapter 2), SSCN (Chapter 9), accelerated sketch-and-project (Chapter 10), and over-parameterized SGD (i.e., SGD applied to a finite-sum problem where $\nabla f_i(x^*) = 0$ for all i ; see Chapter 6 for the general method and rate or Chapter 4 for an application to distributed optimization).
- **Stochastic algorithms that do not use control variates despite the fact that control variates would improve the rate.** Such methods converge sublinearly (or converge linearly to a certain neighborhood of the optimal solution) due to the inherent variance of the gradient estimator at the optimum. This category includes local SGD (Chapter 8), some variants of sparsified parallel algorithms (Chapter 4), and a number of other SGD variants that can be analyzed using the framework of Chapter 6.
- **Linearly converging stochastic algorithms aided by control variates.** Those include SEGA (Chapter 3), sparsified VR algorithms ISEGA, ISAGA (Chapter 4), as well as local SGD with variance reduction (Chapter 8). All of these algorithms can also be obtained as a special case of the GJS framework (Chapter 5) – GJS tightens the rate of SEGA and extends both ISEGA and ISAGA (and allows for their combination). The rate of GJS is further improved in Chapter 7, which allows for exploiting the specific structure of the regularizer ψ .

Randomization over the data or parameters

As already mentioned, there are two different ways in which randomization can enter an optimization procedure – either subsampling the domain (parameters) or subsampling the finite sum (data).

- **Subsampling the space.** Generally speaking, methods in this category in each iteration compute the gradient over a randomly chosen subspace only. This corresponds to a subset of partial derivatives in the special case when the subspace is spanned by a subset of the standard unit basis vectors. While some algorithms update the current iterate along the selected random subspace only (ACD from Chapter 2 and SSCN from Chapter 9), the others perform a full dimensional update due to the presence of control variates (SEGA from Chapter 3, ISEGA from Chapter 4 or SVRCD from Chapters 5, 7). We shall also mention that the methods aided by the control variates are usually somewhat slower than the methods moving along the subspace only.
- **Subsampling the data.** Various chapters of this thesis propose or improve upon known methods that subsample the finite sum (1.1). As a special case of the GJS framework (Chapter 5), we were able to introduce Loopless SVRG (LSVRG) [83, 106]

with arbitrary sampling and proximal step (thus making it significantly faster). Next, we introduce a linearly convergent variance reduced local SGD method in Chapter 8; which is by an order of magnitude faster than other variants of local SGD in the literature. Lastly, the unified SGD analysis we provide in Chapter 6 allowed us to both analyze a new, with-replacement minibatch SGD method with importance sampling which is cheaper to implement than the without-replacement variant, and improve upon several quantized SGD algorithms (for example, we propose the first quantized SGD method with arbitrary sampling).

- **Subsampling both the domain and the space at the same time.** Two chapters of this work consider random linear measurements of the Jacobian as an oracle model: GJS (Chapter 5) is a variance reduced algorithm for minimizing a general finite-sum objective, while accelerated sketch-and-project (Chapter 10) is an algorithm for minimizing quadratics. Our oracle model allows for sampling from both the space and the finite sum at the same time. In a special case, this reduces to the gradient sparsification approach we propose in Chapter 4, and thus recovers the ISEGA, ISAGA or ISAEGA algorithms we which proposed previously. Needless to say, the unified SGD analysis from Chapter 6 captures this level of generality as well.

To conclude this section, we shall mention that domain-subsampling algorithms are often capable of performing finite sum subsampling, either through the product space objective, which we introduce in Chapter 7, or via the duality trick from [191].

Distributed optimization

Two chapters of this work consider predominantly distributed optimization, where the bottleneck of the optimization system is communication. Chapter 4 and Chapter 8 present two orthogonal approaches in two different distributed setups. Specifically, Chapter 4 introduces a new method based on random sparsification of the gradient, which *provably reduces the worker→server communication by order of the number of the workers at essentially no cost*.¹⁸ On the other hand, Chapter 8 focuses on the *federated learning* paradigm. In it, we introduce a novel personalization-encouraging objective which we argue is more natural to be optimized by local gradient methods, and for the first time prove communication complexity benefits of local gradient decent methods. We shall note that all variance reduced algorithms introduced in these chapters are a special case of GJS (Chapter 5), and at the same time, Chapter 5 extends the results of Chapter 4 allowing for both subsampling the local objective and gradient sparsification while keeping linear rate. Further, all convergence rates of Chapters 4, 8, as well as the rates of other quantized algorithms for distributed optimization [136, 85] can be obtained as a special case of the framework of Chapter 6.

¹⁸In some distributed computation systems, communication from the workers to the server, is 10-20 times more expensive than the communication from the server to workers [136].

Importance sampling for minibatches

While minibatch variants of CD methods are very popular in practice, until now, there was no importance sampling for CD that outperforms the standard uniform minibatch sampling in terms of worst-case guarantees. In Chapter 2 we design new importance sampling for minibatch CD and minibatch ACD which significantly outperforms previous state-of-the-art minibatch ACD in practice. Surprisingly, the sampling strategy applies to stochastic minibatch methods that subsample the finite sum objective – it can improve upon SGD, SAGA, SVRG and others.¹⁹ Further, Chapter 6 presents a with-replacement variant of SGD, where the importance minibatch sampling is particularly cheap to implement.

Proximal methods

Most of the algorithms proposed in this work support arbitrary proximable regularizer ψ which is proper, closed, convex, and possibly non-smooth. This includes Chapters 3, 5, 6, 7 and 8 and a part of Chapter 4 (further generalized in Chapter 5). Next, Chapter 9 requires ψ to be separable as it proposes a subspace descent method without control variates.

While the standard analysis of proximal methods provides a rate identical to the corresponding non-proximal variants, in Chapter 7 we show that the presence of ψ with a specific structure might significantly simplify the problem and thus enable faster optimization. As a consequence of this observation, we show that fast rates of variance reduced algorithms that subsample the finite sum can be obtained from variance reduced methods that subsample the space. This establishes a new and deep link between two strands of optimization methods.

Accelerated algorithms

Many of the algorithms proposed in this work incorporate some form of Nesterov’s acceleration [149]. In some chapters, the acceleration is the or one of the key contributions (i.e., ACD in Chapter 2, ASVRCD in Chapter 7, accelerated sketch-and-project in Chapter 10), while some other chapters merely demonstrate that acceleration can be incorporated into the loop (i.e., ASEG in Chapter 3 or IASGD in Chapter 4).

Second order methods

While this thesis focuses predominantly on first-order optimization, Chapters 9 and 10 study second-order algorithms as well. Specifically, Chapter 9 introduces the Stochastic Subspace Cubic Newton method (SSCN) – a new globally convergent second-order subspace descent method. On the other hand, Chapter 10 introduces accelerated sketch-and-project method for solving linear systems in Euclidean spaces, which can be seen as a first-order and second-order method at the same time due to the quadratic nature of the objective.

¹⁹It applies to all special cases covered by Chapters 5 and 6; see the corresponding appendices.

Chapter	Ref	Alg	VR	Accel	Subsp	Prox	Distrib	Note
2	[78]	ACD	✗	✓	✓*	✗	✗	Minibatch sampling
3	[77]	SEGA	✓	✓	✓ [†]	✓	✗	Non-separable regularizer
4	[137]	ISEGA	✓	✓	✓ [†]	✓	✓	Reduced communication
5	[79]	GJS	✓	✗	✓ [†]	✓	✓	General VR framework
6	[55]	SGD	✓	✗	✓ [†]	✓	✓	General SGD analysis
7	[76]	ASVRCD	✓	✓	✓ [†]	✓	✗	Subspace \geq finite sum
8	[80]	LGD	✓	✗	✗	✓	✓	Local SGD with VR
9	[74]	SSCN	✗	✗	✓	✓ [‡]	✗	2 nd order + cubic regularizer
10	[58]	AMI	✗	✓	✓	✗	✗	Quadratic objective

Table 1.1: Summary of representative algorithms proposed in each chapter and topics covered in each chapter. Columns (chapter topics): VR = variance reduced method, Accel = Nesterov’s acceleration, Subsp = subspace descent, Prox = proximal setup, Distrib = distributed setup. Further clarifications: * ACD allows for subspaces spanned by standard basis vectors only; [†] these methods consider a general subspace oracle, but perform full dimensional updates; [‡] SSCN requires the regularizer ψ to be separable.

Summary of the links among the chapters

To conclude this section, we summarize both what the chapters are about, as well outline several links among them.

First, Table 1.1 presents a representative algorithm for each chapter of this thesis, as well as the covered topics. Next, Table 1.2 highlights which algorithms presented in this thesis are novel and which are not. Lastly, Figure 1.1 summarizes the essential connections among the chapters of this thesis that were outlined above.

1.4 Outline and individual contributions

Each chapter of this work consists of a single paper; some of them are already published while the others are at various stages of the submission process. Let us now give a brief overview of the contents of each chapter individually.

#	New	#	New	#	New	#	New	#	New	#	New	#	New
1	✗	11	✓	21	✓	31	✓	41	✓	51	✓	61	✓
2	✗	12	✓	22	✓	32	✓	42	✓	52	✗	62	✓
3	✗	13	✗	23	✓	33	✓	43	✓	53	✗	63	✓
4	✗	14	✓	24	✓	34	✓	44	✗	54	✗	64	✓
5	✓	15	✓	25	✓	35	✓	45	✗	55	✗		
6	✓	16	✓	26	✓	36	✓	46	✓	56	✓		
7	✓	17	✓	27	✗	37	✓	47	✓	57	✗		
8	✓	18	✓	28	✗	38	✓	48	✗	58	✗		
9	✓	19	✓	29	✓	39	✓	49	✓	59	✓		
10	✓	20	✓	30	✓	40	✓	50	✓	60	✓		

Table 1.2: List of all algorithms stated in this work. Marker ✓ indicates that the algorithm is new (i.e., proposed in this work) while marker ✗ indicates that the algorithm is known.

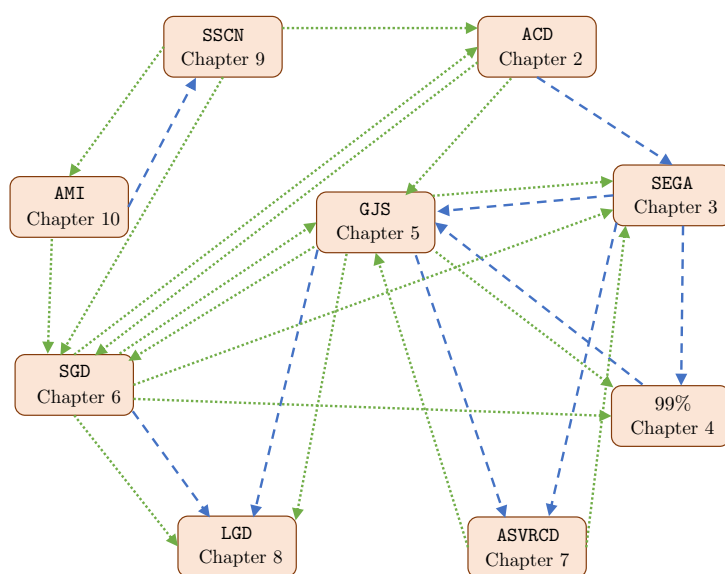


Figure 1.1: Graph depicting the relationships among the chapters of this thesis. Blue dashed arrow indicates motivation among chapters, while green dotted arrow indicates a significant insight that chapters shed on each other. As an example, let us explain the edges of Chapter 3 (SEGA): the development of SEGA was enabled by our results on CD (Chapter 2) and motivated us to develop the results contained in Chapters 4, 5 and 7. Further, Chapter 5 recovers/improves upon the convergence rate of SEGA, Chapter 6 enables a partial variance reduction in SEGA and lastly, Chapter 5 shows that SAGA is a special case of SEGA.

1.4.1 Accelerated coordinate descent with arbitrary sampling and best rates for minibatches (Chapter 2)

Accelerated coordinate descent is a widely popular optimization algorithm due to its efficiency in large-dimensional problems. It achieves state-of-the-art complexity on an impor-

tant class of empirical risk minimization problems. In this work, we design and analyze an accelerated coordinate descent (ACD) method, which in each iteration updates a random subset of coordinates according to an arbitrary but fixed probability law, which is a parameter of the method. While minibatch variants of ACD are more popular and relevant in practice, there is no importance sampling for ACD that outperforms the standard uniform minibatch sampling. Through insights enabled by our general analysis, we design new importance sampling for minibatch ACD, which significantly outperforms previous state-of-the-art minibatch ACD in practice. We prove a rate that is at most $\mathcal{O}(\sqrt{\tau})$ times worse than the rate of minibatch ACD with uniform sampling, but can be $\mathcal{O}(d/\tau)$ times better, where τ is the minibatch size. Since in modern supervised learning training systems, it is standard practice to choose $\tau \ll d$, and often $\tau = \mathcal{O}(1)$, our method can lead to dramatic speedups. We obtain similar results for minibatch non-accelerated CD as well, achieving improvements on previous best rates. Further, the importance sampling for non-accelerated CD can be incorporated into stochastic algorithms that decompose finite sums such as SGD, SAGA, and others.

The chapter is based on the paper:

[78] Filip Hanzely and Peter Richtárik. Accelerated coordinate descent with arbitrary sampling and best rates for minibatches. In *Proceedings of Machine Learning Research*, pages 304–312. PMLR, 16–18 Apr 2019.

1.4.2 SEGA: Variance reduction via gradient sketching (Chapter 3)

In Chapter 3, we propose a randomized first-order optimization method—SEGA (SkEtched GrAdient)—which progressively throughout its iterations builds a variance-reduced estimate of the gradient from random linear measurements (sketches) of the gradient obtained from an oracle. In each iteration, SEGA updates the current estimate of the gradient through a sketch-and-project operation using the information provided by the latest sketch, and this is subsequently used to compute an unbiased estimate of the true gradient through a random relaxation procedure. This unbiased estimate is then used to perform a gradient step. Unlike standard subspace descent methods, such as coordinate descent, SEGA can be used for optimization problems with a *non-separable* proximal term. We provide a general convergence analysis and prove linear convergence for strongly convex objectives. In the special case of coordinate sketches, SEGA can be enhanced with various techniques such as *importance sampling*, *minibatching*, and *acceleration*, and its rate is up to a small constant factor identical to the best-known rate of coordinate descent from Chapter 2.

The chapter is based on the paper:

[77] Filip Hanzely, Konstantin Mishchenko, and Peter Richtárik. SEGA: Variance reduction via gradient sketching. In *Advances in Neural Information Processing Systems*, pages 2083–2094, 2018.

1.4.3 99% of Worker-Master Communication in Distributed Optimization is Not Needed (Chapter 4)

We improve upon algorithms that fit the following template: a local gradient estimate is computed independently by each worker, then communicated to a master, which subsequently performs averaging. The average is broadcast back to the workers, which uses it to perform a gradient-type step to update the local version of the model. We observe that the above template is fundamentally inefficient in that too much data is unnecessarily communicated from the workers to the server, which slows down the overall system. We propose a fix based on a new update-sparsification method we develop in this work, which we suggest be used on top of existing methods. Namely, we develop a new variant of parallel block coordinate descent based on independent sparsification of the local gradient estimates before communication. We demonstrate that with only m/n blocks sent by each of n workers, where m is the total number of parameter blocks, the theoretical iteration complexity of the underlying distributed methods is essentially unaffected. As an illustration, this means that when $n = 100$ parallel workers are used, the communication of 99% blocks is redundant, and hence a waste of time. Our theoretical claims are supported through extensive numerical experiments that demonstrate an almost perfect match with our theory on a number of synthetic and real datasets.

The chapter is based on the paper:

[137] Konstantin Mishchenko, Filip Hanzely, and Peter Richtárik. 99% of worker-master communication in distributed optimization is not needed. In *36th Conference on Uncertainty in Artificial Intelligence, (UAI 2020)*. AUAI, 2020.

1.4.4 One method to rule them all: Variance reduction for data, parameters and many new methods (Chapter 5)

Next, in Chapter 3, we propose a remarkably general variance-reduced method suitable for solving regularized empirical risk minimization problems with either a large number of training examples, or a large model dimension, or both. In special cases, our method reduces to several known and previously thought to be unrelated methods, such as SAGA [37], LSVRG [83, 106], JacSketch [65], SEGA [77] and ISEGA [137], and their arbitrary sampling and proximal generalizations. However, we also highlight a large number of new specific algorithms with interesting properties. We provide a single theorem establishing linear convergence of the method under smoothness and quasi strong convexity assumptions. With this theorem, we recover best-known and sometimes improved rates for known methods arising in special cases. As a by-product, we provide the first unified method and theory for stochastic gradient and stochastic coordinate descent type methods.

The chapter is based on the paper:

[79] Filip Hanzely and Peter Richtárik. One method to rule them all: Variance reduction for data, parameters and many new methods. *arXiv preprint arXiv:1905.11266*, 2019.

1.4.5 A unified theory of SGD: Variance reduction, sampling, quantization and coordinate descent (Chapter 6)

We introduce a unified analysis of a large family of variants of proximal stochastic gradient descent (SGD), which so far have required different intuitions, convergence analyses, have different applications, and which have been developed separately in various communities. We show that our framework includes methods with and without the following tricks, and their combinations: variance reduction, importance sampling, mini-batch sampling, quantization, and coordinate sub-sampling. As a by-product, we obtain the first unified theory of SGD and randomized coordinate descent (CD) methods, the first unified theory of variance reduced and non-variance-reduced SGD methods, and the first unified theory of quantized and non-quantized methods. A key to our approach is a parametric assumption on the iterates and stochastic gradients. In a single theorem, we establish a linear convergence result under this assumption and strong-quasi convexity of the loss function. Whenever we recover an existing method as a special case, our theorem gives the best-known complexity result. Our approach can be used to motivate the development of new useful methods and offers pre-proved convergence guarantees. To illustrate the strength of our approach, we develop five new variants of SGD, and through numerical experiments, demonstrate some of their properties.

The chapter is based on the paper:

[55] Eduard Gorbunov, Filip Hanzely, and Peter Richtárik. A unified theory of SGD: Variance reduction, sampling, quantization and coordinate descent. In *The 23rd International Conference on Artificial Intelligence and Statistics*, 2020.

1.4.6 Variance reduced coordinate descent with acceleration: New method with a surprising application to finite-sum problems (Chapter 7)

Further, in Chapter 7, we propose ASVRCD: an accelerated version of stochastic variance reduced coordinate descent. As other variance reduced coordinate descent methods such as SEGA or SVRCD, our method can deal with problems that include a non-separable and non-smooth regularizer, while accessing a random block of partial derivatives in each iteration only. However, ASVRCD incorporates Nesterov’s momentum, which offers favorable iteration complexity guarantees over both SEGA and SVRCD. As a by-product of our theory, we show that a variant of Katyusha [4] is a specific case of ASVRCD, recovering the optimal oracle complexity for the finite sum objective.

The chapter is based on the paper:

[76] Filip Hanzely, Dmitry Kovalev, and Peter Richtárik. Variance reduced coordinate descent with acceleration: New method with a surprising application to finite-sum problems. In *International Conference on Machine Learning*, 2020.

1.4.7 Federated learning of a mixture of global and local models (Chapter 8)

We propose a new optimization formulation for training federated learning models. The standard formulation has the form of an empirical risk minimization problem constructed to find a single global model trained from the private data stored across all participating devices. In contrast, our formulation seeks an explicit trade-off between this traditional global model and the local models, which can be learned by each device from its own private data without any communication. Further, we develop several efficient variants of SGD (with and without partial participation and with and without variance reduction) for solving the new formulation and prove communication complexity guarantees. Notably, our methods are similar but not identical to federated averaging / local SGD, thus shedding some light on the essence of the elusive method. In particular, our methods do not perform full averaging steps and instead merely take steps towards averaging. We argue for the benefits of this new paradigm for federated learning.

The chapter is based on the paper:

[80] Filip Hanzely and Peter Richtárik. Federated learning of a mixture of global and local models. *arXiv preprint arXiv:2002.05516*, 2020.

1.4.8 Stochastic subspace cubic Newton (Chapter 9)

In Chapter 9, we propose a new randomized second-order optimization algorithm—Stochastic Subspace Cubic Newton (SSCN)—for minimizing a high dimensional convex function f . Our method can be seen both as a *stochastic* extension of the cubically-regularized Newton method [156], and a *second-order* enhancement of stochastic subspace descent [109]. We prove that as we vary the minibatch size, the global convergence rate of SSCN interpolates between the rate of stochastic coordinate descent (CD) and the rate of cubic regularized Newton, thus giving new insights into the connection between first and second-order methods. Remarkably, the local convergence rate of SSCN matches the rate of stochastic subspace descent applied to the problem of minimizing the quadratic function $x \mapsto \frac{1}{2}(x - x^*)^\top \nabla^2 f(x^*)(x - x^*)$, where x^* is the minimizer of f , and hence depends on the properties of f at the optimum only. Our numerical experiments show that SSCN outperforms non-accelerated first-order CD algorithms while being competitive to their accelerated variants.

The chapter is based on the paper:

[74] Filip Hanzely, Nikita Doikov, Peter Richtárik, and Yurii Nesterov. Stochastic subspace cubic Newton method. In *International Conference on Machine Learning*, 2020.

1.4.9 Accelerated stochastic matrix inversion: General theory and speeding up BFGS rules for faster second-order optimization (Chapter 10)

In Chapter 10, we present the first accelerated randomized algorithm for solving linear systems in Euclidean spaces. One essential problem of this type is the matrix inversion

problem. In particular, our algorithm can be specialized to invert positive definite matrices in such a way that all iterates (approximate solutions) generated by the algorithm are positive definite matrices themselves. This opens the way for many applications in the field of optimization and machine learning. As an application of our general theory, we develop the *first accelerated (deterministic and stochastic) quasi-Newton updates*. Our updates lead to provably more aggressive approximations of the inverse Hessian and lead to speedups over classical non-accelerated rules in numerical experiments. Experiments with empirical risk minimization show that our rules can accelerate the training of machine learning models.

The chapter is based on the paper:

[58] Robert M Gower, Filip Hanzely, Peter Richtárik, and Sebastian U Stich. Accelerated stochastic matrix inversion: general theory and speeding up bfgs rules for faster second-order optimization. In *Advances in Neural Information Processing Systems*, pages 1619–1629, 2018.

1.4.10 Excluded papers

I had a chance to co-author four more papers during my studies, which are not included in this work: one about an accelerated mirror descent method for relatively smooth optimization [81], two about robust principal component analysis [45, 44] and the last one about optimal algorithms for personalized federated learning [75].

Chapter 2

Accelerated Coordinate Descent with Arbitrary Sampling and Best Rates for Minibatches

In this chapter we consider a particular instance of the general optimization problem (1.1) with $\psi \equiv 0$ and f not necessarily having a finite-sum structure, i.e.,

$$\min_{x \in \mathbb{R}^d} f(x). \quad (2.1)$$

Specifically, we assume that f is a smooth and strongly convex function, and the main difficulty comes from the dimension d being very large (e.g., millions or billions). In this regime, *coordinate descent* (CD) variants of gradient methods are the state of the art.

The simplest variant of CD in each iteration updates a single variable of x by taking a one dimensional gradient step along the direction of the i th unit basis vector $e_i \in \mathbb{R}^d$, which leads to the update rule

$$x^{k+1} = x^k - \alpha_i \nabla_i f(x^k) e_i, \quad (2.2)$$

where $\nabla_i f(x^k) \stackrel{\text{def}}{=} e_i^\top \nabla f(x^k)$ is the i th partial derivative and α_i is a suitably chosen stepsize. The classical smoothness assumption used in the analysis of CD methods [152] is to require the existence of constants $L_i > 0$ such that

$$f(x + te_i) \leq f(x) + t \nabla_i f(x) + \frac{L_i}{2} t^2 \quad (2.3)$$

holds for all $x \in \mathbb{R}^d$, $t \in \mathbb{R}$ and $i \in [d] \stackrel{\text{def}}{=} \{1, 2, \dots, d\}$. In this setting, one can choose the stepsizes to be $\alpha_i = 1/L_i$.

There are several rules studied in the literature for choosing the coordinate i in iteration k , including cyclic rules [130, 206, 183, 215, 72], Gauss-Southwell or other greedy rules [161, 220, 201], random (stationary) rules [152, 173, 177, 192, 122, 49] and adaptive random rules [29, 202]. In this work we focus on stationary random rules, which are popular by practitioners and well understood in theory.

Updating one coordinate at a time. The simplest randomized CD method of the form (2.2) chooses coordinate i in each iteration uniformly at random. If f is μ -strongly convex, then this method converges in $(d \max_i L_i / \mu) \log(1/\epsilon)$ iterations in expectation. If index i is chosen with probability $p_i \propto L_i$, then the iteration complexity improves to $(\sum_i L_i / \mu) \log(1/\epsilon)$. The latter result is always better than the former, and can be up to d times better. These results were established in a seminal paper by Nesterov [152]. The

analysis was later generalized to arbitrary probabilities $p_i > 0$ by Richtárik and Takáč [173], who obtained the complexity

$$\left(\max_i \frac{L_i}{p_i \mu} \right) \log \frac{1}{\epsilon}. \quad (2.4)$$

Clearly, (2.4) includes the previous two results as special cases. Note that the importance sampling probabilities given by $p_i \propto L_i$ minimizes the complexity bound (2.4) and are therefore in this sense optimal.

Minibatching: updating more coordinates at a time. In many situations it is advantageous to update a small *subset* (*minibatch*) of coordinates in each iteration, which leads to the *minibatch CD method* which has the form

$$x_i^{k+1} = \begin{cases} x_i^k - \alpha_i \nabla_i f(x^k) & i \in S^k, \\ x_i^k & i \notin S^k. \end{cases} \quad (2.5)$$

For instance, it is often equally easy to fetch information about a small batch of coordinates S^k from memory at the same or comparable time as it is to fetch information about a single coordinate. If this memory access time is the bottleneck as opposed to computing the actual updates to coordinates $i \in S^k$, then it is more efficient to update all coordinates belonging to the minibatch S^k . Alternatively, in situations where parallel processing is available, one is able to compute the updates to a small batch of coordinates simultaneously, leading to speedups in wall clock time. With this application in mind, minibatch CD methods are also often called *parallel* CD methods [177].

2.1 Arbitrary sampling and minibatching

Arbitrary sampling. The method (2.5) was analyzed in [177] for *uniform samplings* S^k , i.e., assuming that $\mathbb{P}(i \in S^k) = \mathbb{P}(j \in S^k)$ for all i, j . However, the ultimate generalization is captured by the notion of *arbitrary sampling* [175]. A *sampling* refers to a set-valued random mapping S with values being the subsets of $[d]$. The word *arbitrary* refers to the fact that no additional assumptions on the sampling, such as uniformity, are made. This result generalizes the results mentioned above.

M-smoothness. For minibatch CD methods it is useful to assume a more general notion of smoothness parameterized by a positive semidefinite matrix $\mathbf{M} \in \mathbb{R}^{d \times d}$. We say that f is \mathbf{M} -smooth if

$$f(x + h) \leq f(x) + \nabla f(x)^\top h + \frac{1}{2} h^\top \mathbf{M} h \quad (2.6)$$

for all $x, h \in \mathbb{R}^d$. The standard L -smoothness condition is obtained in the special case when $\mathbf{M} = L\mathbf{I}$, where \mathbf{I} is the identity matrix in \mathbb{R}^d . Note that if f is \mathbf{M} -smooth, then (2.3) holds for $L_i = \mathbf{M}_{ii}$. Conversely, it is known that if (2.3) holds, then (2.6) holds for $\mathbf{M} = d\mathbf{Diag}(L_1, L_2, \dots, L_d)$ [152]. If h has at most ω nonzero entries, then this result can be strengthened and (2.6) holds with $\mathbf{M} = \omega\mathbf{Diag}(L_1, L_2, \dots, L_d)$ [177, Theorem 8]. In many situations, \mathbf{M} -smoothness is a very natural assumption. For instance, in

the context of empirical risk minimization (ERM), which is a key problem in supervised machine learning, f is of the form $f(x) = \frac{1}{n} \sum_{i=1}^n \phi_i(\mathbf{A}_i x) + \frac{\mu}{2} \|x\|^2$, where $\mathbf{A}_i \in \mathbb{R}^{m \times d}$ are data matrices, $\phi_i : \mathbb{R}^m \rightarrow \mathbb{R}$ are loss functions and $\mu \geq 0$ is a regularization constant. If ϕ_i is convex and γ_i -smooth for all i , then f is μ -strongly convex and \mathbf{M} -smooth with $\mathbf{M} = (\frac{1}{n} \sum_i \gamma_i \mathbf{A}_i^\top \mathbf{A}_i) + \mu \mathbf{I}$ [167]. In these situations it is useful to design CD algorithms making full use of the information contained in the data as captured in the smoothness matrix \mathbf{M} .

Given a sampling S and \mathbf{M} -smooth function f , let $v = (v_1, \dots, v_d)$ be positive constants satisfying the ESO (expected separable overapproximation) inequality

$$\mathbf{P} \circ \mathbf{M} \preceq \mathbf{Diag}(p_1 v_1, \dots, p_d v_d), \quad (2.7)$$

where \mathbf{P} is the *probability matrix* associated with sampling S , defined by $\mathbf{P}_{ij} \stackrel{\text{def}}{=} \mathbb{P}(i \in S \ \& \ j \in S)$, $p_i \stackrel{\text{def}}{=} \mathbf{P}_{ii} = \mathbb{P}(i \in S)$ and \circ denotes the Hadamard (i.e., elementwise) product of matrices. From now on we define the *probability vector* as $p \stackrel{\text{def}}{=} (p_1, \dots, p_d) \in \mathbb{R}^d$ and let $v = (v_1, \dots, v_d) \in \mathbb{R}^d$ be the vector of ESO parameters. With this notation, (2.7) can be equivalently written as $\mathbf{P} \circ \mathbf{M} \preceq \mathbf{Diag}(p \circ v)$. We say that S is *proper* if $p_i > 0$ for all i .

It can be show by combining the results of [175] and [167] that under the above assumptions, the minibatch CD method (2.5) with stepsizes $\alpha_i = 1/v_i$ enjoys the iteration complexity

$$\left(\max_i \frac{v_i}{p_i \mu} \right) \log \frac{1}{\epsilon}. \quad (2.8)$$

Since in situations when $|S^k| = 1$ with probability 1 once can choose $v_i = L_i$, the complexity result (2.8) generalizes (2.4). Inequality (2.7) is standard in minibatch coordinate descent literature. It was studied extensively in [167], and has been used to analyze parallel CD methods [177, 175, 49], distributed CD methods [174, 48], accelerated CD methods [49, 48, 166, 21], and dual methods [169, 21].

Importance sampling for minibatches. It is easy to see, for instance, that if we do not restrict the class of samplings over which we optimize, then the trivial *full sampling* $S^k = [d]$ with probability 1 is optimal. For this sampling, \mathbf{P} is the matrix of all ones, $p_i = 1$ for all i , and (2.7) holds for $v_i = L \stackrel{\text{def}}{=} \lambda_{\max}(\mathbf{M})$ for all i . The minibatch CD method (2.5) reduces to gradient descent, and the complexity estimate (2.8) becomes $(L/\mu) \log(1/\epsilon)$, which is the standard rate of gradient descent. However, typically we are interested in finding the best sampling from the class of samplings which use a minibatch of size τ , where $\tau \ll d$. While we have seen that the importance sampling $p_i = L_i / \sum_j L_j$ is optimal for $\tau = 1$, in the minibatch case $\tau > 1$ the problem of determining a sampling which minimizes the bound (2.8) is much more difficult. For instance, [175] consider a certain parametric family of samplings where the problem of finding the best sampling from this family reduces to a linear program.

Surprisingly, and in contrast to the situation in the $\tau = 1$ case where an optimal sampling is known and is in general non-uniform, there is no minibatch sampling that is

	CD	ACD
$\tau = 1, p_i > 0$	$\left(\max_i \frac{L_i}{p_i \mu} \right) \log \frac{1}{\epsilon}$ [173]	$\sqrt{\max_i \frac{L_i}{p_i^2 \mu} \log \frac{1}{\epsilon}}$ (this work)
$\tau = 1, \text{ best } p_i$	$\frac{\sum_i L_i}{\mu} \log \frac{1}{\epsilon}; \quad p_i \propto L_i$ [152]	$\frac{\sum_i \sqrt{L_i}}{\sqrt{\mu}} \log \frac{1}{\epsilon}; \quad p_i \propto \sqrt{L_i}$ [7]
arbitrary sampling S	$\left(\max_i \frac{v_i}{p_i \mu} \right) \log \frac{1}{\epsilon}$ [175]	$\sqrt{\max_i \frac{v_i}{p_i^2 \mu} \log \frac{1}{\epsilon}}$ (this work)

Table 2.1: Complexity results for non-accelerated (CD) and accelerated (ACD) coordinate descent methods for μ -strongly convex functions and arbitrary sampling S . The last row corresponds to the setup with arbitrary proper sampling S (i.e., a random subset of $[d]$ with the property that $p_i \stackrel{\text{def}}{=} \mathbb{P}(i \in S) > 0$). We let $\tau \stackrel{\text{def}}{=} \mathbf{E}[|S|]$ be the expected mini-batch size. We assume that f is \mathbf{M} -smooth (see (2.6)). The positive constants v_1, v_2, \dots, v_d are the ESO parameters (depending on f and S), defined in (2.7). The first row arises as a special of the third row in the non-minibatch (i.e., $\tau = 1$) case. Here we have $v_i = L_i \stackrel{\text{def}}{=} \mathbf{M}_{ii}$. The second row is a special case of the first row for the optimal choice of the probabilities p_1, p_2, \dots, p_d .

guaranteed to outperform τ -nice sampling. We say that S is τ -nice if it samples uniformly from among all subsets of $[d]$ of cardinality τ . The probability matrix of this sampling is given by $\mathbf{P} = \frac{\tau}{d} ((1 - \beta)\mathbf{I} + \beta ee^\top)$, where $\beta = \frac{\tau-1}{d-1}$ (assume $d > 1$) and $e \in \mathbb{R}^d$ is the vector of all ones, and $p_i = \frac{\tau}{d}$ [167]. It follows that the ESO inequality (2.7) holds for $v_i = (1 - \beta)\mathbf{M}_{ii} + \beta L$. By plugging into (2.8), we get the iteration complexity

$$\frac{d}{\tau} \left(\frac{(1 - \beta) \max_i \mathbf{M}_{ii} + \beta L}{\mu} \right) \log \frac{1}{\epsilon}. \quad (2.9)$$

This rate interpolates between the rate of CD with uniform probabilities (for $\tau = 1$) and the rate of gradient descent (for $\tau = d$).

2.2 Contributions

For *accelerated coordinate descent (ACD)* without minibatching (i.e., when $\tau = 1$), the currently best known iteration complexity result, due to [7], is

$$\mathcal{O} \left(\frac{\sum_i \sqrt{L_i}}{\sqrt{\mu}} \log \frac{1}{\epsilon} \right). \quad (2.10)$$

The probabilities used in the algorithm are proportional to the square roots of the coordinate-wise Lipschitz constants: $p_i \propto \sqrt{L_i}$. This is the first CD method with a complexity guarantee which does not explicitly depend on the dimension n , and is an improvement on

the now-classical result of [152] giving the complexity

$$\mathcal{O}\left(\sqrt{\frac{d \sum_i L_i}{\mu}} \log \frac{1}{\epsilon}\right).$$

The rate (2.10) is always better than this, and can be up to \sqrt{d} times better if the distribution of L_i is extremely non-uniform. Unlike in the non-accelerated case described in the previous section, there is no complexity result for ACD with general probabilities such as (2.4), or with an arbitrary sampling such as (2.8). In fact, an ACD method was not even designed in such settings, despite a significant recent development in accelerated coordinate descent methods [152, 116, 122, 166, 7].

To summarize, our key contributions are:

- **ACD with arbitrary sampling.** We design an ACD method which is able to operate with an *arbitrary sampling* of subsets of coordinates. We describe our method in Section 2.3.
- **Iteration complexity.** We prove (see Theorem 2.3.2) that the iteration complexity of ACD is

$$\mathcal{O}\left(\sqrt{\max_i \frac{v_i}{p_i^2 \mu}} \log \frac{1}{\epsilon}\right), \quad (2.11)$$

where v_i are ESO parameters given by (2.7) and $p_i > 0$ is the probability that coordinate i belongs to the sampled set S^k : $p_i \stackrel{\text{def}}{=} \mathbb{P}(i \in S^k)$. The result of Allen-Zhu et al. (2.10) (NUACDM) can be recovered as a special case of (2.11) by focusing on samplings defined by $S^k = \{i\}$ with probability $p_i \propto \sqrt{L_i}$ (recall that in this case $v_i = L_i$). When $S^k = [d]$ with probability 1, then our method reduces to accelerated gradient descent (Algorithm 2, AGD), and since $p_i = 1$ and $v_i = L$ (the Lipschitz constant of ∇f) for all i , (2.11) reduces to the standard complexity of AGD: $\mathcal{O}(\sqrt{L/\mu} \log(1/\epsilon))$.

- **Weighted strong convexity.** We prove a slightly more general result than (2.11) in which we allow the strong convexity of f to be measured in a weighted Euclidean norm with weights v_i/p_i^2 . In situations when f is naturally strongly convex with respect to a weighted norm, this more general result will typically lead to a better complexity result than (2.11), which is fine-tuned for standard strong convexity. There are applications when f is naturally a strongly convex with respect to some weighted norm [7].
- **Minibatch methods.** We design several *new* importance samplings for minibatches, calculate the associated complexity results, and show through experiments that they significantly outperform the standard uniform samplings used in practice and constitute the state of the art. Our importance sampling leads to rates which are provably within a small factor from the best known rates, but can lead to an improvement by a factor of $\mathcal{O}(d)$. We are the first to establish such a result, both for CD (Appendix B.2) and ACD (Section 2.4). Further, the importance sampling

we design for CD can be applied beyond coordinate descent algorithms: Chapters 5 and 6 discuss an application in stochastic algorithms that subsample the finite sum.

The key complexity results obtained in this chapter are summarized and compared to prior results in Table 2.1.

2.3 The ACD algorithm

The accelerated coordinate descent method (ACD) we propose is formalized as Algorithm 5. If we removed (4) and (7) from the method, and replaced y^{k+1} in (6) by x^{k+1} , we would recover the CD method. Acceleration is obtained by the inclusion of the extrapolation steps (4) and (7). As mentioned before, we will analyze our method under a more general strong convexity assumption.

Assumption 2.3.1. *Function f is μ_w -strongly convex with respect to the $\|\cdot\|_w$ norm. That is,*

$$f(x+h) \geq f(x) + \langle \nabla f(x), h \rangle + \frac{\mu_w}{2} \|h\|_w^2, \quad (2.12)$$

for all $x, h \in \mathbb{R}^d$, where $\mu_w > 0$.

Note that if f is μ -strongly convex in the standard sense (i.e., for $w = (1, \dots, 1)$), then f is μ_w -strongly convex for any $w > 0$ with $\mu_w = \min_i \frac{\mu}{w_i}$. Considering a general μ_w -strong convexity allows us to get a tighter convergence rate in some cases [7].

Algorithm 5 ACD (Accelerated coordinate descent with arbitrary sampling)

- 1: **Parameters:** i.i.d. proper samplings $S^k \sim \mathcal{D}$; $v, w \in \mathbb{R}_{++}^d$; $\mu_w > 0$; stepsize parameters $\eta, \theta > 0$.
 - 2: Initial iterate $y^0 = z^0 \in \mathbb{R}^d$
 - 3: **for** $k = 0, 1, 2, \dots$ **do**
 - 4: $x^{k+1} = (1 - \theta)y^k + \theta z^k$
 - 5: Get $S^k \sim \mathcal{D}$
 - 6: $y^{k+1} = x^{k+1} - \sum_{i \in S^k} \frac{1}{v_i} \nabla_i f(x^{k+1}) e_i$
 - 7: $z^{k+1} = \frac{1}{1 + \eta \mu_w} \left(z^k + \eta \mu_w x^{k+1} - \sum_{i \in S^k} \frac{\eta}{p_i w_i} \nabla_i f(x^{k+1}) e_i \right)$
 - 8: **end for**
-

Using the tricks developed in [116, 49, 122], Algorithm 5 can be implemented so that only $|S^k|$ coordinates are updated in each iteration. We are now ready derive a convergence rate of ACD.

Theorem 2.3.2 (Convergence of ACD). *Let S^k be i.i.d. proper (but otherwise arbitrary) samplings. Let \mathbf{P} be the associated probability matrix and $p_i \stackrel{\text{def}}{=} \mathbb{P}(i \in S^k)$. Assume f is \mathbf{M} -smooth (see (2.6)) and let v be ESO parameters satisfying (2.7). Further, assume that f is μ_w -strong convex (with $\mu_w > 0$) for*

$$w_i \stackrel{\text{def}}{=} \frac{v_i}{p_i^2}, \quad i = 1, 2, \dots, d, \quad (2.13)$$

with respect to the weighted Euclidean norm $\|\cdot\|_w$ (i.e., we enforce Assumption 2.3.1). Then

$$\mu_w \leq \frac{\mathbf{M}_{ii} p_i^2}{v_i} \leq p_i^2 \leq 1, \quad i = 1, 2, \dots, d. \quad (2.14)$$

In particular, if f is μ -strongly convex with respect to the standard Euclidean norm, then we can choose

$$\mu_w = \min_i \frac{p_i^2 \mu}{v_i}. \quad (2.15)$$

Finally, if we choose

$$\theta \stackrel{\text{def}}{=} \frac{\sqrt{\mu_w^2 + 4\mu_w} - \mu_w}{2} = \frac{2\mu_w}{\sqrt{\mu_w^2 + 4\mu_w} + \mu_w} \geq 0.618\sqrt{\mu_w}$$

and $\eta \stackrel{\text{def}}{=} \frac{1}{\theta}$, then the random iterates of ACD satisfy

$$\mathbb{E}[P^k] \leq (1 - \theta)^k P^0, \quad (2.16)$$

where $P^k \stackrel{\text{def}}{=} \frac{1}{\theta^2} (f(y^k) - f(x^*)) + \frac{1}{2(1-\theta)} \|z^k - x^*\|_w^2$ and x^* is the optimal solution of (2.1).

Noting that $1/0.618 \leq 1.619$, as an immediate consequence of (2.16) and (2.16) we get bound

$$k \geq \frac{1.619}{\sqrt{\mu_w}} \log \frac{1}{\epsilon} \Rightarrow \mathbb{E}[P^k] \leq \epsilon P^0. \quad (2.17)$$

If f is μ -strongly convex, then by plugging (2.15) into (2.17) we obtain the iteration complexity bound

$$1.619 \cdot \sqrt{\max_i \frac{v_i}{p_i^2 \mu}} \log \frac{1}{\epsilon}. \quad (2.18)$$

Complexity (2.18) is our key result (also mentioned in (2.11) and Table 2.1).

2.4 Importance sampling for minibatches

Let $\tau \stackrel{\text{def}}{=} \mathbb{E}[|S^k|]$ be the expected minibatch size. The next theorem provides an insightful lower bound for the complexity of ACD we established, one independent of p and v .

Theorem 2.4.1 (Limits of minibatch performance). *Let the assumptions of Theorem 2.3.2 be satisfied and let f be μ -strongly convex. Then the dominant term in the rate (2.18) of ACD admits the lower bound*

$$\sqrt{\max_i \frac{v_i}{p_i^2 \mu}} \geq \frac{\sum_i \sqrt{\mathbf{M}_{ii}}}{\tau \sqrt{\mu}}. \quad (2.19)$$

Note that for $\tau = 1$ we have $\mathbf{M}_{ii} = v_i = L_i$, and the lower bound is achieved by using the importance sampling $p_i \propto \sqrt{L_i}$. Hence, this bound gives a limit on how much speedup, compared to the best known complexity in the $\tau = 1$ case, we can hope for as

Lower bound	$S_1 : p_i = \frac{\tau}{d}$	$S_2 : \frac{p_i^2}{\mathbf{M}_{ii}} \propto 1$	$S_3 : \frac{p_i^2}{\mathbf{M}_{ii}} \propto 1 - p_i$
$\frac{\sum_i \sqrt{\mathbf{M}_{ii}}}{\tau \sqrt{\mu}}$	$\frac{d\sqrt{(1-\beta) \max_i \mathbf{M}_{ii} + \beta L}}{\tau \sqrt{\mu}}$	$\frac{\gamma \sum_i \sqrt{\mathbf{M}_{ii}}}{\tau \sqrt{\mu}}$	$\omega \frac{d\sqrt{(1-\beta) \max_i \mathbf{M}_{ii} + \beta L}}{\tau \sqrt{\mu}}$
(2.19)	= uniform ACD for $\tau = 1$ = AGD for $\tau = d$	$\leq \sqrt{d} \times \text{lower bound}$ $\bullet \tau \leq \frac{\sum_j \sqrt{\mathbf{M}_{jj}}}{\max_i \mathbf{M}_{ii}}$	\bullet fastest in practice \bullet any τ allowed

Table 2.2: New complexity results for ACD with minibatch size $\tau = \mathbb{E}[|S^k|]$ and various samplings (we suppress $\log(1/\epsilon)$ factors in all expressions). Constants: $\mu =$ strong convexity constant of f , $L = \lambda_{\max}(\mathbf{M})$, $\beta = (\tau - 1)/(d - 1)$, $1 \leq \gamma \leq \sqrt{d}$, and $\omega \leq \mathcal{O}(\sqrt{\tau})$ (ω can be as small as $\mathcal{O}(\tau/d)$).

we increase τ . The bound says we can not hope for better than linear speedup in the minibatch size. An analogous result (obtained by removing all the squares and square roots in (2.19)) was established in [175] for CD.

In what follows, it will be useful to write the complexity result (2.18) in a new form by considering a specific choice of the ESO vector v .

Lemma 2.4.2. *Choose any proper sampling S and let \mathbf{P} be its probability matrix and p its probability vector. Let $c(S, \mathbf{M}) \stackrel{\text{def}}{=} \lambda_{\max}(\mathbf{P}' \circ \mathbf{M}')$, where $\mathbf{P}' \stackrel{\text{def}}{=} \mathbf{D}^{-1/2} \mathbf{P} \mathbf{D}^{-1/2}$, $\mathbf{M}' \stackrel{\text{def}}{=} \mathbf{D}^{-1} \mathbf{M} \mathbf{D}^{-1}$ and $\mathbf{D} \stackrel{\text{def}}{=} \text{Diag}(p)$. Then the vector v defined by $v_i \stackrel{\text{def}}{=} c(S, \mathbf{M}) p_i^2$ satisfies the ESO inequality (2.7) and the total complexity (2.18) becomes*

$$1.619 \cdot \frac{\sqrt{c(S, \mathbf{M})}}{\sqrt{\mu}} \log \frac{1}{\epsilon}. \quad (2.20)$$

Let $\text{Tr}(\cdot)$ be a trace function. Since $\frac{1}{d} \text{Tr}(\mathbf{P}' \circ \mathbf{M}') \leq c(S, \mathbf{M}) \leq \text{Tr}(\mathbf{P}' \circ \mathbf{M}')$ and

$$\text{Tr}(\mathbf{P}' \circ \mathbf{M}') = \sum_i \mathbf{P}'_{ii} \mathbf{M}'_{ii} = \sum_i \mathbf{M}'_{ii} = \sum_i \mathbf{M}_{ii} / p_i^2,$$

we get the bounds:

$$\sqrt{\frac{1}{d} \sum_i \frac{\mathbf{M}_{ii}}{p_i^2 \mu}} \log \frac{1}{\epsilon} \leq \sqrt{\frac{c(S, \mathbf{M})}{\mu}} \log \frac{1}{\epsilon} \leq \sqrt{\sum_i \frac{\mathbf{M}_{ii}}{p_i^2 \mu}} \log \frac{1}{\epsilon}. \quad (2.21)$$

2.4.1 Sampling 1: standard uniform minibatch sampling

Let S_1 be the τ -nice sampling. It can be shown (see Lemma B.3.3) that $c(S_1, \mathbf{M}) \leq \frac{d^2}{\tau^2}((1 - \beta) \max_i \mathbf{M}_{ii} + \beta L)$, and hence the iteration complexity (2.18) becomes

$$\mathcal{O} \left(\frac{d}{\tau} \sqrt{\frac{(1 - \beta) \max_i \mathbf{M}_{ii} + \beta L}{\mu}} \log \frac{1}{\epsilon} \right). \quad (2.22)$$

This result interpolates between ACD with uniform probabilities (for $\tau = 1$) and accelerated gradient descent (for $\tau = d$). Note that the rate (2.22) is a strict improvement on the CD rate (2.9).

2.4.2 Sampling 2: importance sampling for minibatches

Consider now the sampling S_2 which includes every $i \in [d]$ in S_2 , independently, with probability $p_i = \tau \frac{\sqrt{\mathbf{M}_{ii}}}{\sum_j \sqrt{\mathbf{M}_{jj}}}$. This sampling was not considered in the literature before.

Note that $\mathbb{E}[|S_2|] = \sum_i p_i = \tau$. For this sampling, bounds (2.21) become:

$$\frac{\sum_i \sqrt{\mathbf{M}_{ii}}}{\tau \sqrt{\mu}} \log \frac{1}{\epsilon} \leq \sqrt{\frac{c(S, \mathbf{M})}{\mu}} \log \frac{1}{\epsilon} \leq \frac{\sqrt{d} \sum_i \sqrt{\mathbf{M}_{ii}}}{\tau \sqrt{\mu}} \log \frac{1}{\epsilon}. \quad (2.23)$$

Clearly, with this sampling we obtain an ACD method with complexity within a \sqrt{d} factor from the lower bound established in Theorem 2.4.1. For $\tau = 1$ we have $\mathbf{P}' = \mathbf{I}$ and hence

$$\begin{aligned} c(S, \mathbf{M}) &= \lambda_{\max}(\mathbf{I} \circ \mathbf{M}') = \lambda_{\max}(\mathbf{Diag}(\mathbf{M}')) \\ &= \max_i \mathbf{M}_{ii} / p_i^2 = \left(\sum_j \sqrt{\mathbf{M}_{jj}} \right)^2. \end{aligned}$$

Thus, the rate of ACD achieves the lower bound in (2.23) (see also (2.10)) and we recover the best current rate of ACD in the $\tau = 1$ case, established by Allen-Zhu et al. [7]. However, the sampling has an important limitation: it can be used for $\tau \leq \sum_j \sqrt{\mathbf{M}_{jj}} / \max_i \mathbf{M}_{ii}$ only as otherwise the probabilities p_i exceed 1.

2.4.3 Sampling 3: another importance sampling for minibatches

Now consider sampling S_3 which includes each coordinate i within S_3 independently, with probability p_i satisfying the relation $p_i^2 / \mathbf{M}_{ii} \propto 1 - p_i$. This is equivalent to setting

$$p_i \stackrel{\text{def}}{=} \frac{2\mathbf{M}_{ii}}{\sqrt{\mathbf{M}_{ii}^2 + 2\mathbf{M}_{ii}\delta} + \mathbf{M}_{ii}}, \quad (2.24)$$

where δ is a scalar for which $\sum_i p_i = \tau$. This sampling was not considered in the literature before. Probability vector p was chosen as (2.24) for two reasons: i) $p_i \leq 1$ for all i ,

and therefore the sampling can be used for all τ in contrast to S_1 , and ii) we can prove Theorem 2.4.3.

Let $c_1 \stackrel{\text{def}}{=} c(S_1, \mathbf{M})$ and $c_3 \stackrel{\text{def}}{=} c(S_3, \mathbf{M})$. In light of (2.20), Theorem 2.4.3 compares S_1 and S_3 and says that ACD with S_3 has at most $\mathcal{O}(\sqrt{\tau})$ times worse rate compared to ACD with S_1 , but has the capacity to be $\mathcal{O}(d/\tau)$ times better. We prove in Appendix B.2 a similar theorem for CD. We stress that, despite some advances in the development of importance samplings for minibatch methods [175, 31], S_1 was until now the state-of-the-art in theory for CD. We are the first to give a provably better rate in the sense of Theorem B.2.3. The numerical experiments show that S_3 consistently outperforms S_1 , and often dramatically so.

Theorem 2.4.3. *The leading complexity terms c_1 and c_3 of ACD (Algorithm (2.5)) with samplings S_1 , and S_3 , respectively, defined in Lemma 2.4.2, compare as follows:*

$$c_3 \leq 2 \frac{(2d - \tau)(d\tau + d - \tau)}{(d - \tau)^2} c_1 = \mathcal{O}(\tau) c_1. \quad (2.25)$$

Moreover, there exists \mathbf{M} where $c_3 \leq \mathcal{O}\left(\frac{\tau^2}{d^2}\right) c_1$.

In real world applications, minibatch size τ is limited by hardware and in typical situations, one has $\tau \ll d$, oftentimes $\tau = \mathcal{O}(1)$. The importance of Theorem 2.4.3 is best understood from this perspective.

2.5 Experiments

We perform extensive numerical experiments to justify that minibatch ACD with importance sampling works well in practice.

We first present some synthetic examples in Section 2.5.1 in order to have better understanding of both acceleration and importance sampling, and to see how it performs on what type of data. We also study how minibatch size influences the convergence rate.

Then, in Section 2.5.2, we work with logistic regression problem on LibSVM [23] data. For small datasets, we choose the parameters of ACD as theory suggests and for large ones, we estimate them, as we describe in the main body of the chapter. Lastly, we tackle dual of SVM problem with squared hinge loss, which we present in Section 2.5.3.¹

In most of plots we compare of both accelerated and non-accelerated CD with all samplings S_1, S_2, S_3 introduced in Sections 2.4.1, 2.4.2 and 2.4.3 respectively. We refer to ACD with sampling S_3 as AN (Accelerated Nonuniform), ACD with sampling S_1 as AU, ACD with sampling S_2 as AN2, CD with sampling S_3 as NN, CD with sampling S_1 as NU and CD with sampling S_2 as NN2. As for Sampling 2, it might happen that probabilities

¹Coordinate descent methods which allow for separable proximal operator were proven to be efficient to solve ERM problem, when applied on dual [189, 191, 192, 223]. Although we do not develop proximal methods in this chapter, we empirically demonstrate that ACD allows for this extension as well. As a specific problem to solve, we choose dual of SVM with hinge loss. The results and a detailed description of the experiment are presented in Section 2.5.3, and are indeed in favour of ACD with importance sampling. Therefore, ACD is not only suitable for *big dimensional* problems, it can handle the *big data* setting as well.

Type	M
1	$\mathbf{A}^\top \mathbf{A} + \mathbf{I}$ for $\mathbf{A}^{\frac{d}{2} \times d}$; have independent entries from $N(0, 1)$
2	$\mathbf{A}^\top \mathbf{A} + \mathbf{I}$ for $\mathbf{A}^{2d \times d}$; have independent entries from $N(0, 1)$
3	$\text{Diag}(1, 2, \dots, d)$
4	$\mathbf{A} + \mathbf{I}$, $\mathbf{A}_{d,d} = d$, $\mathbf{A}_{1:(d-1),1:(d-1)} = 1$, $\mathbf{A}_{1:(d-1),d} = \mathbf{A}_{d,1:(d-1)} = 0$
5	$\mathbf{A}^\top \mathbf{D} \mathbf{A} + \mathbf{I}$ for $\mathbf{A}^{\frac{d}{2} \times d}$; have independent entries from $N(0, 1)$, $\mathbf{D} = \frac{1}{\sqrt{d}} \text{Diag}(1, 2, \dots, d)$

Table 2.3: Problem types for testing ACD.

become larger than one if τ is large (see Section 2.4.2), we set those probabilities to 1 while keeping the rest as it is.

All the experimental results clearly show that acceleration, importance sampling and minibatching have a significant impact on practical performance of CD methods. Moreover, the difference in the performance of samplings S_2 and S_3 is negligible, and therefore we recommend using S_3 , as it is not limited by the bound on expected minibatch size τ .

2.5.1 Synthetic quadratics

As we mentioned, the goal of this section is to provide a better understanding of both acceleration and importance sampling. For this purpose we consider as simple setting as possible – minimizing quadratic

$$f(x) = \frac{1}{2} x^\top \mathbf{M} x - b^\top x, \quad (2.26)$$

where $b \sim N(0, I)$ and \mathbf{M} is chosen as one of the 5 types, as Table 2.3 suggests.

In the first example we perform (Figure 2.1), we compare the performance of both accelerated and non-accelerated algorithm with both nonuniform and τ nice sampling on problems as per Table 2.3. In all experiments, we set $d = 1000$ and we plot a various choices of τ .

Comparison of methods on synthetic data

Figure 2.1 presents the numerical performance of ACD for various types of synthetic problems given by (2.26) and Table 2.3. It suggests what our theory shows: accelerated algorithm is always faster than its non-accelerated counterpart, and on top of that, performance of τ -nice sampling (S_1) can be negligibly faster than importance sampling (S_2, S_3), but is usually significantly slower. A significance of the importance sampling is mainly demonstrated on problem type 4, which roughly coincides with Examples 12 and 13. Figure 2.1 presents Sampling 2 only for the cases when the bound on τ from Section 2.4.2 is satisfied.

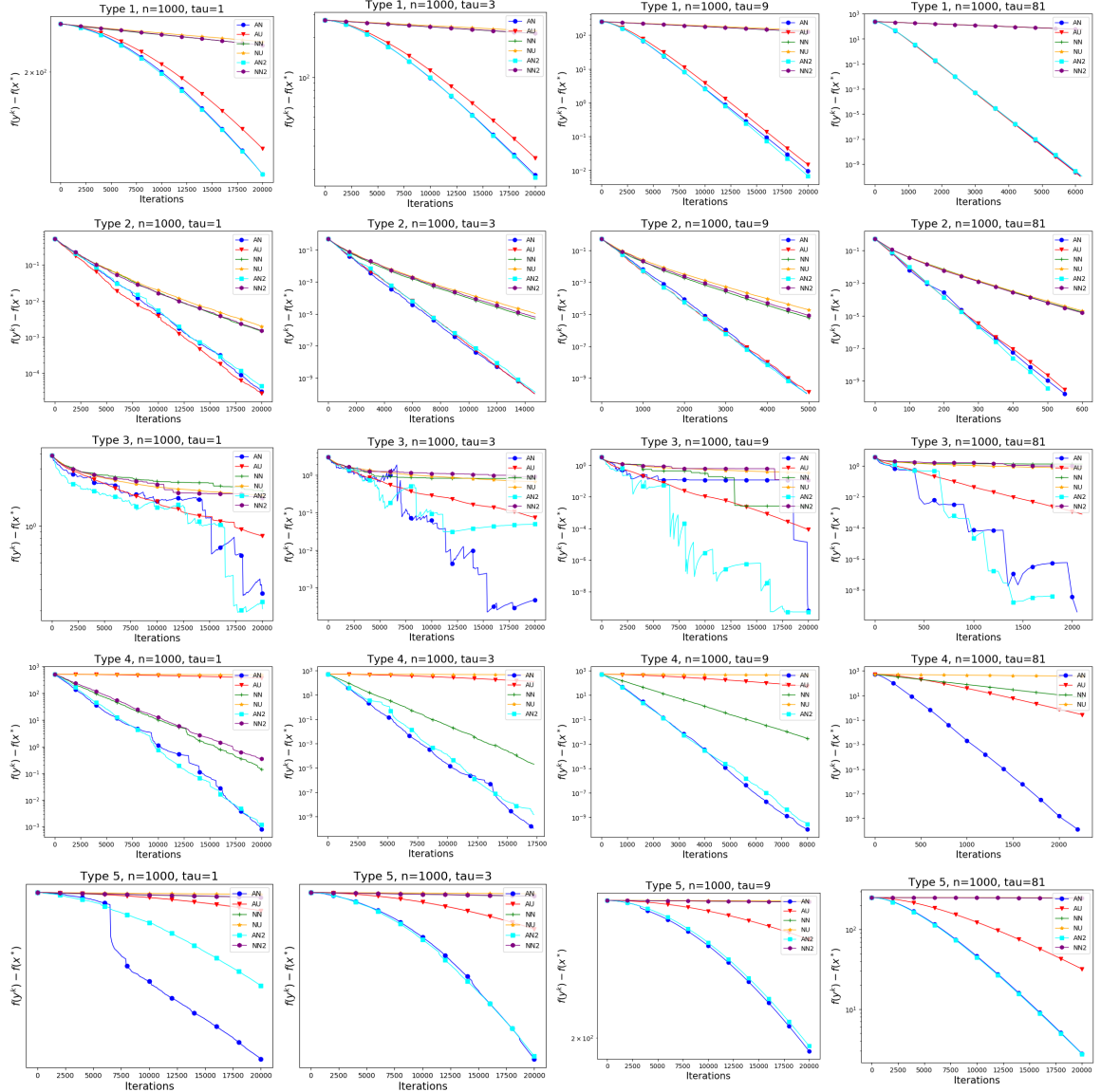


Figure 2.1: Coordinate descent. Comparison of accelerated, nonaccelerated algorithm with both importance and τ nice sampling for a various quadratic problems.

Speedup in τ

The next experiment shows an empirical speedup for the coordinate descent algorithms for a various types of problems. For simplicity, we do not include Sampling 2. Figure 2.2 provides the results. Oftentimes, the empirical speedup (in terms of the number of iteration) in τ is close to linear, which demonstrates the power and significance of minibatching.

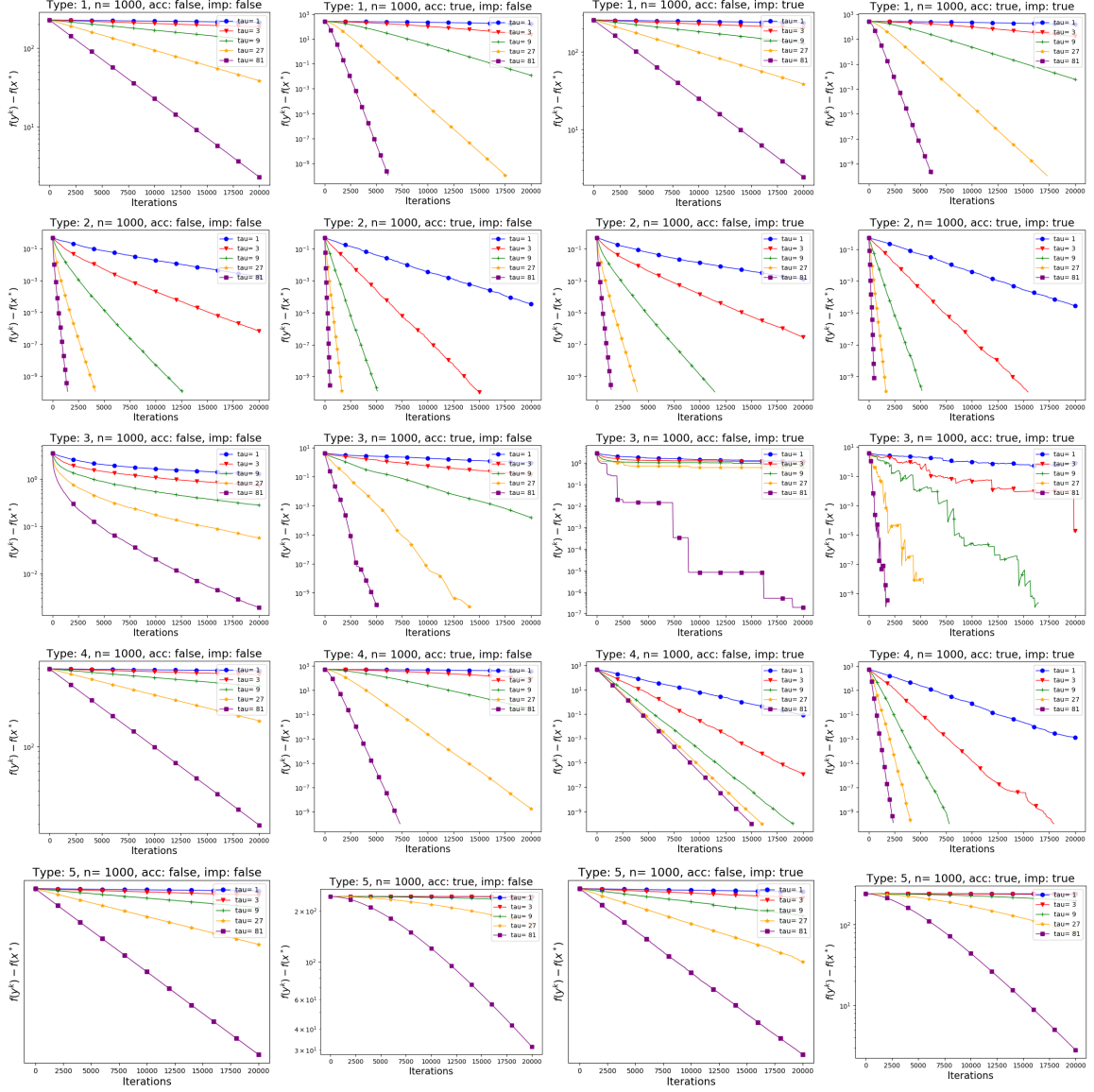


Figure 2.2: Coordinate descent. Comparison of speedup gained by both τ -nice sampling and importance sampling with and without acceleration on various quadratic problems.

2.5.2 Logistic regression

In this section we apply ACD on the regularized logistic regression problem, i.e.

$$f(x) = \frac{1}{n} \sum_{i=1}^n \log(1 + \exp(\mathbf{A}_{i,:} x \cdot b)) + \frac{\lambda}{2} \|x\|^2,$$

for $b \in \{-1, 1\}$ and data matrix \mathbf{A} comes from LibSVM. In each experiment in this section, we have chosen regularization parameter λ to be the average diagonal element of the smoothness matrix. We first apply the methods with the optimal parameters as our theory suggests on smaller datasets. On larger ones (Section 2.5.2), we set them in

a cheaper way, which is not guaranteed to work by theory we provide.

In our first experiment, we apply ACD on LibSVM data directly for various minibatch sizes τ . Figure 2.3 shows the results. As expected, ACD is always better to CD, and importance sampling is always better to uniform one.

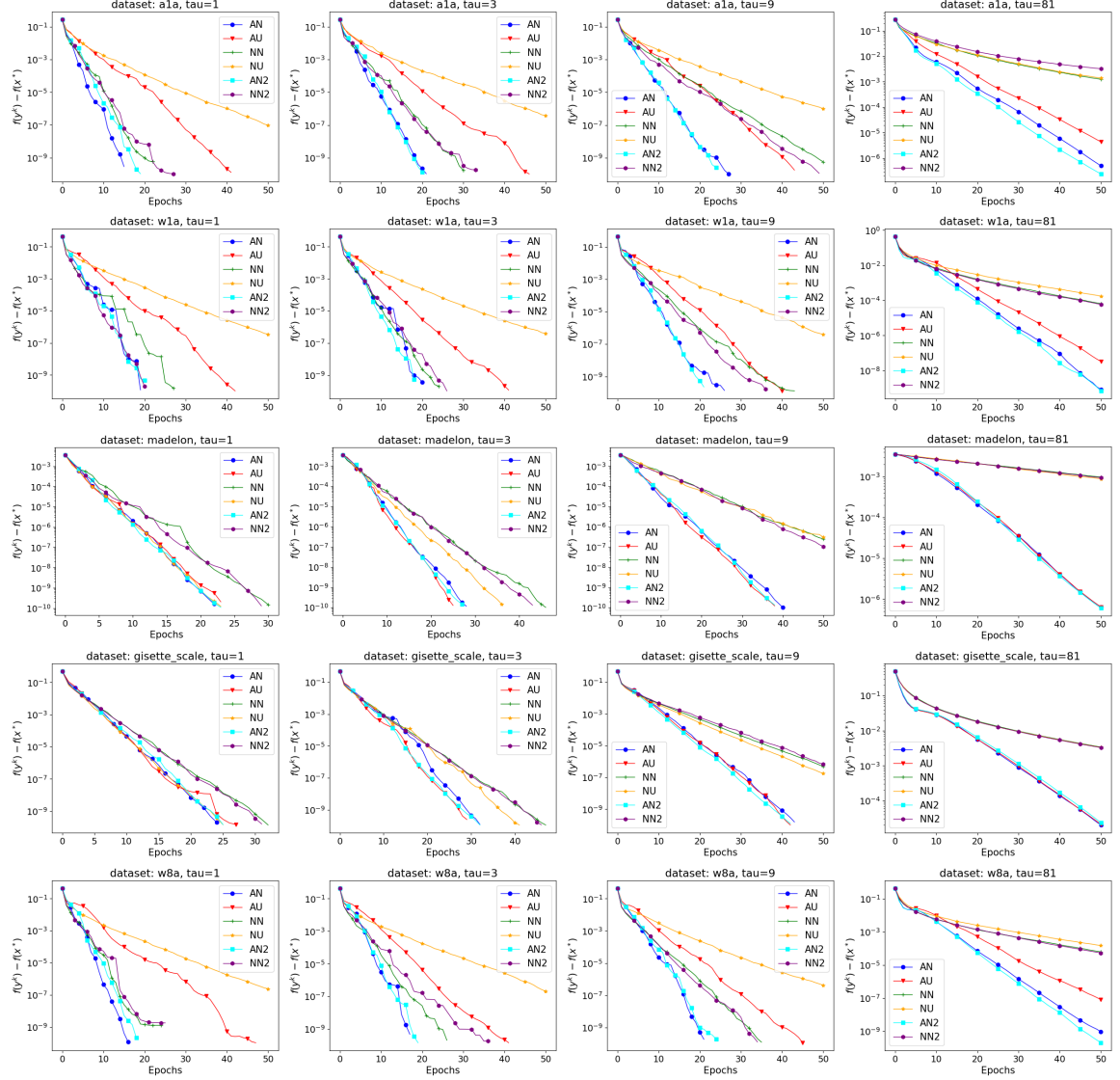


Figure 2.3: Accelerated coordinate descent applied on the logistic regression problem, for various LibSVM datasets and minibatch sizes τ

Note that, for some datasets and especially bigger minibatch sizes, the effect of importance sampling is sometimes negligible. To demonstrate the power of importance sampling, in the next experiment, we first corrupt the data – we multiply each row and column of the data matrix \mathbf{A} by random number from uniform distribution over $[0, 1]$. The results can be seen in Figure 2.4. As expected, the effect of importance sampling becomes more significant.

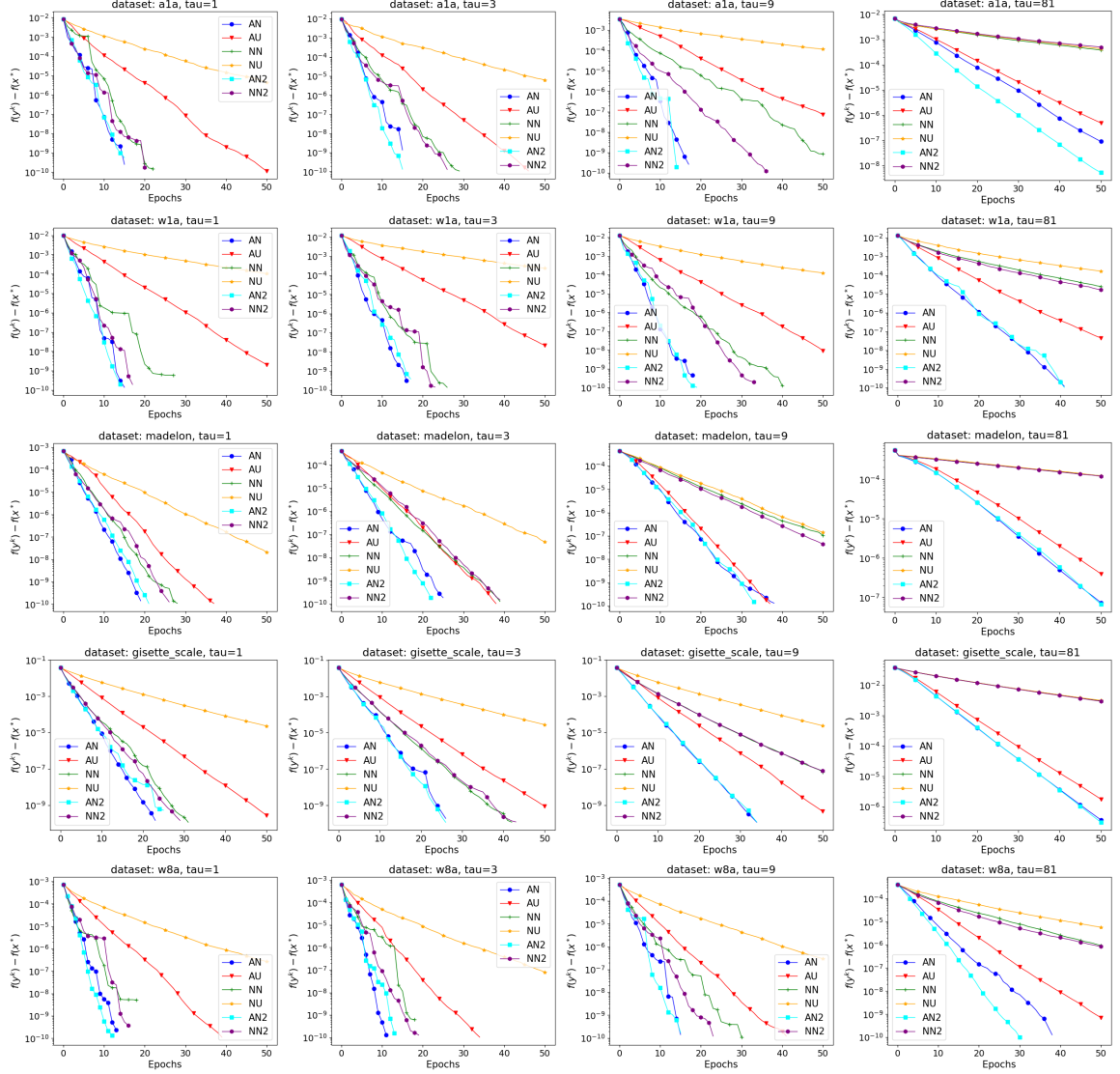


Figure 2.4: ACD applied on the logistic regression problem, for various rescaled LibSVM datasets and minibatch sizes τ .

Practical method on larger dataset

In Figure 2.5, we report on a logistic regression problem with a few selected LibSVM [23] datasets. For larger datasets, pre-computing both strong convexity parameter μ and v may be expensive (however, recall that for v we need to tune only one scalar). Therefore, we choose ESO parameters v from Lemma 2.4.2, while estimating the smoothness matrix as $10\times$ its diagonal. An estimate of the strong convexity μ for acceleration was chosen to be the minimal diagonal element of the smoothness matrix. We provide a formal formulation of the logistic regression problem, along with more experiments applied to further datasets in Appendix 2.5.2, where we choose v and μ in full accord with the theory.

We have chosen regularization parameter λ to be the average diagonal element of the smoothness matrix and estimated v, μ as described in Section 2.5.

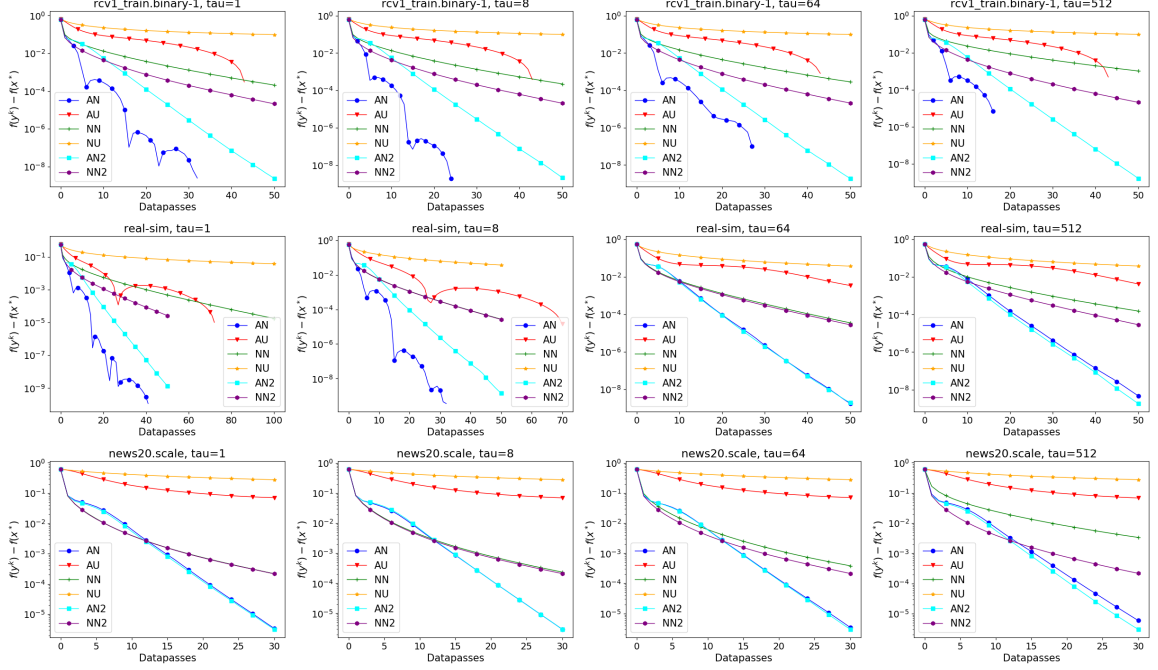


Figure 2.5: Six variants of coordinate descent (AN, AU, NN, NU, AN2 and AU2) applied to a logistic regression problem, with minibatch sizes $\tau = 1, 8, 64$ and 512 .

2.5.3 Support vector machines

In this section we apply ACD on the dual of SVM problem with squared hinge loss, i.e.,

$$f(x) = \frac{1}{\lambda d^2} \sum_{j=1}^n \left(\sum_{i=1}^d b_i \mathbf{A}_{ji} x_i \right)^2 - \frac{1}{d} \sum_{i=1}^d x_i + \frac{1}{4d} \sum_{i=1}^d x_i^2 + \mathcal{I}_{[0, \infty]}(x),$$

where $\mathcal{I}_{[0, \infty]}$ stands for indicator function of set $[0, \infty]$, i.e. $\mathcal{I}_{[0, \infty]}(x) = 0$ if $x \in \mathbb{R}_+^d$, otherwise $\mathcal{I}_{[0, \infty]}(x) = \infty$. As for the data, we have rescaled each row and each column of the data matrix coming from LibSVM by random scalar generated from uniform distribution over $[0, 1]$. We have chosen regularization parameter λ to be maximal diagonal element of the smoothness matrix divided by 10 in each experiment below. We deal with nonsmooth indicator function using proximal operator, which happens to be a projection in this case. We choose ESO parameters v from Lemma 2.4.2, while estimating the smoothness matrix as \sqrt{d} -times multiple of its diagonal. An estimate of the strong convexity μ for acceleration was chosen to be minimal diagonal element of the smoothness matrix, therefore we adapt a similar approach as in Section 2.5.2.

Recall that we did not provide a theory for the proximal steps. However, we make the experiment to demonstrate that ACD can solve big data problems on top of large dimensional problems. Although the results are presented in the main body, we restate them here again (Figure 2.6) for the sake of readability.

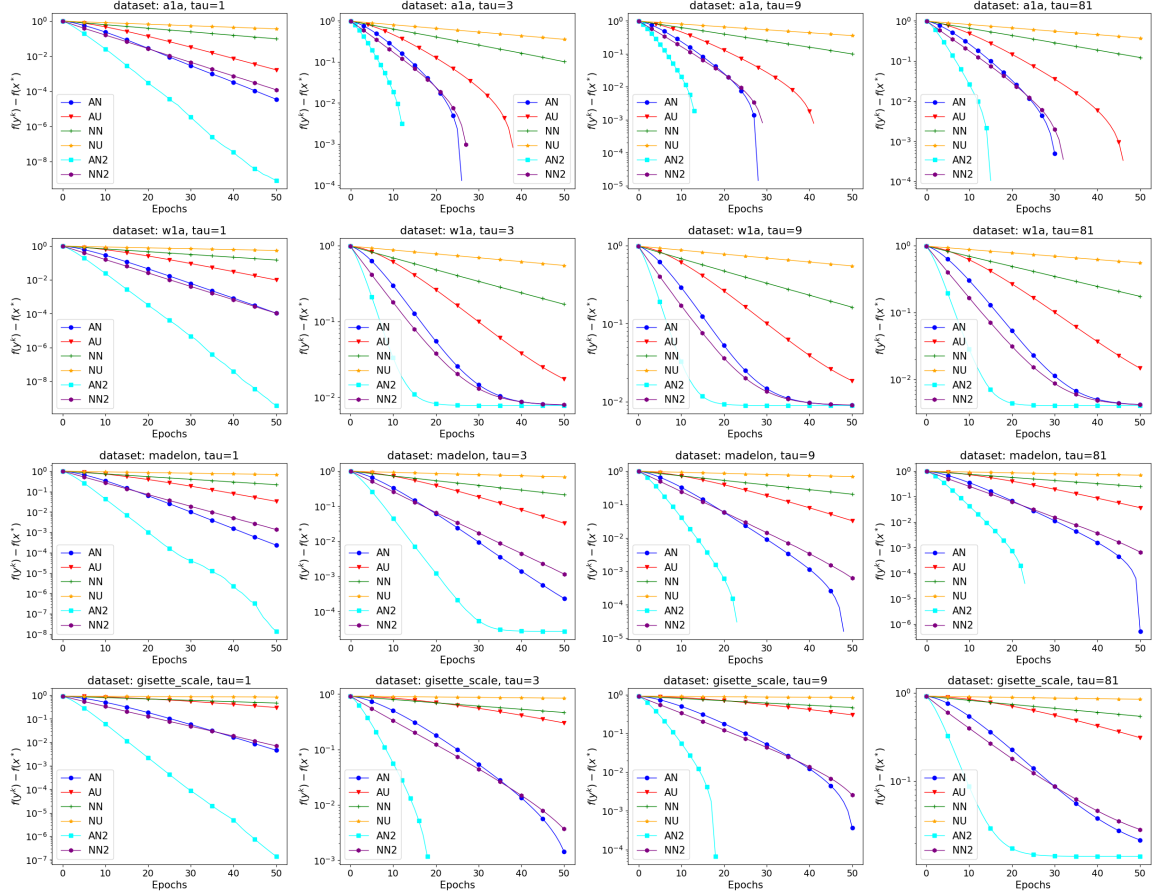


Figure 2.6: Accelerated coordinate descent applied on the dual of SVM with squared hinge loss, for various LibSVM datasets.

2.6 Conclusion

In this chapter we have presented an minibatch version of accelerated coordinate descent and provided best rates for arbitrary sampling. We have introduced the importance sampling for minibatches, which can be arbitrarily better to uniform sampling, but can be at most constant times worse to uniform sampling. This is the first result of the kind for minibatch coordinate descent samplings.

As mentioned throughout the chapter, setting of Algorithm 5 has a limitation – it does not allow a minimization with non-separable regularizer using proximal operator. In particular, objective with non-separable proximal regularizer is not expected to have zero gradient at optimum; and therefore coordinate descent methods can not be expected to converge, unless a decreasing step size is used which leads to significantly slower method. The next chapter solves the issue using variance reduction technique called SEGA.

Chapter 3

SEGA: Variance Reduction via Gradient Sketching

In this chapter, we again consider a specific instance of the optimization problem (1.1). In particular, f is not necessarily assumed to have a finite-sum structure. However, we allow the presence of a closed convex regularizer $\psi : \mathbb{R}^d \rightarrow \mathbb{R} \cup \{+\infty\}$, of which a proximal operator (1.4) is available. In summary, we aim to solve the following optimization task:

$$\min_{x \in \mathbb{R}^d} \left\{ F(x) \stackrel{\text{def}}{=} f(x) + \psi(x) \right\}. \quad (3.1)$$

3.1 Gradient sketching

The main goal of this chapter is to design provably fast proximal gradient-type methods for solving (3.1) *without assuming that the true gradient of f is available*. Instead, we assume that an oracle provides a *random linear transformation (i.e., a sketch) of the gradient*, which is the information available to drive the iterative process. In particular, given a fixed distribution \mathcal{D} over matrices $\mathbf{S} \in \mathbb{R}^{d \times \tau}$ ($b \geq 1$ can but does not need to be fixed), and a query point $x \in \mathbb{R}^d$, our oracle provides us the random linear transformation of the gradient given by

$$\zeta(\mathbf{S}, x) \stackrel{\text{def}}{=} \mathbf{S}^\top \nabla f(x) \in \mathbb{R}^\tau, \quad \mathbf{S} \sim \mathcal{D}. \quad (3.2)$$

Information of this type is available/used in a variety of scenarios. For instance, randomized coordinate descent (CD) methods use oracle (3.2) with \mathcal{D} corresponding to a distribution over standard basis vectors. Minibatch/parallel variants of CD methods utilize oracle (3.2) with \mathcal{D} corresponding to a distribution over random column submatrices of the identity matrix. If one is prepared to use difference of function values to approximate directional derivatives, then one can apply our oracle model to zeroth-order optimization [27]. Indeed, the directional derivative of f in a random direction $\mathbf{S} = s \in \mathbb{R}^{d \times 1}$ can be approximated by $\zeta(s, x) \approx \frac{1}{\epsilon}(f(x + \epsilon s) - f(x))$, where $\epsilon > 0$ is sufficiently small.

Example 2 (Sketches). We now illustrate this concept using two examples.

- (i) **Coordinate sketch.** Let \mathcal{D} be the uniform distribution over standard unit basis vectors e_1, e_2, \dots, e_d of \mathbb{R}^d . Then $\zeta(e_i, x) = e_i^\top \nabla f(x)$, i.e., the i^{th} *partial derivative* of f at x .
- (ii) **Gaussian sketch.** Let \mathcal{D} be the standard Gaussian distribution in \mathbb{R}^d . Then for $s \sim \mathcal{D}$ we have $\zeta(s, x) = s^\top \nabla f(x)$, i.e., the *directional derivative* of f at x in direction s .

We describe SEGA in Section 3.3. Convergence results for general sketches are described in Section 3.4. Refined results for coordinate sketches are presented in Section 3.5, where we also describe and analyze an accelerated variant of SEGA. Experimental results can be found in Section 3.6. We also include here experiments with a *subspace* variant of SEGA, which is described and analyzed in Appendix C.3. Conclusions are drawn and potential extensions outlined in Section 3.7. A simplified analysis of SEGA in the case of coordinate sketches and for $\psi \equiv 0$ is developed in Appendix C.4 (under standard assumptions as in the main body).

We introduce notation when and where needed. For convenience, we provide a table of frequently used notation in Appendix A.

3.1.1 Related work

In the last decade, stochastic gradient-type methods for solving problem (3.1) have received unprecedented attention by theoreticians and practitioners alike. Specific examples of such methods are stochastic gradient descent (SGD) [179], variance-reduced variants of SGD such as SAG [182], SAGA [37], SVRG [88], and their accelerated counterparts [121, 4]. While these methods are specifically designed for objectives formulated as an expectation or a finite sum, we do not assume such a structure. Moreover, these methods utilize a fundamentally different stochastic gradient information: they have access to an unbiased estimator of the gradient. In contrast, we do not assume that (3.2) is an unbiased estimator of $\nabla f(x)$. In fact, $\zeta(\mathbf{S}, x) \in \mathbb{R}^r$ and $\nabla f(x) \in \mathbb{R}^d$ do not even necessarily belong to the same space. Therefore, our algorithms and results should be seen as complementary to the above line of research.

While the gradient sketch $\zeta(\mathbf{S}, x)$ does not immediately lead to an unbiased estimator of the gradient, SEGA uses the information provided in the sketch to *construct* an unbiased estimator of the gradient via a *sketch-and-project* process. Sketch-and-project iterations were introduced in [61] in the context of linear feasibility problems. A dual view uncovering a direct relationship with stochastic subspace ascent methods was developed in [62]. The latest and most in-depth treatment of sketch-and-project for linear feasibility is based on the idea of stochastic reformulations [176]. Sketch-and-project can be combined with Polyak [129, 128] and Nesterov momentum [58], extended to convex feasibility problems [145], matrix inversion [64, 63, 58], and empirical risk minimization [57, 65]. Connections to gossip algorithms for average consensus were made in [127, 126].

The line of work most closely related to our setup is that on randomized coordinate/-subspace descent methods [152, 62]. Indeed, the information available to these methods is compatible with our oracle for specific distributions \mathcal{D} . However, the main disadvantage of these methods is that they are not able to handle non-separable regularizers ψ . In contrast, the algorithm we propose—SEGA—works for any regularizer ψ . In particular, SEGA can handle non-separable constraints even with coordinate sketches, which is out of range of current coordinate descent methods. Hence, our work could be understood as extending the reach of coordinate and subspace descent methods from separable to arbitrary regularizers, which allows for a plethora of new applications. Our method is able to work with an arbitrary regularizer due to its ability to *build an unbiased variance-reduced estimate of the gradient* of f throughout the iterative process from the random linear

measurements thereof provided by the oracle. Moreover, and unlike coordinate descent, SEGA allows for general sketches from essentially any distribution \mathcal{D} .

Another stream of work on designing gradient-type methods without assuming perfect access to the gradient is represented by the *inexact gradient descent* methods [34, 40, 185]. However, these methods deal with deterministic estimates of the gradient and are not based on linear transformations of the gradient. Therefore, this second line of research is also significantly different from what we do here.

3.2 Contributions

We now list the main contributions of this chapter.

- **Subspace oracle with non-separable regularizer.** SEGA is the first iterative proximal algorithm with a subspace gradient oracle that achieves linear convergence. Unlike coordinate descent, SEGA does not require the regularizer to be separable and thus has a much broader range of applications. It achieves by constructing control variance to progressively reduce the variance of stochastic gradient estimator.
- **Generality and Subspace SEGA .** We provide the convergence rate of SEGA under the full generality – we allow for arbitrary distribution of sketching matrices \mathbf{S} . In some scenarios, this might lead to a very fast convergence, especially when ∇f always belongs to a particular subspace.
- **Fast rates without ψ .** Given that $\psi \equiv 0$, we show that SEGA is, up to a small constant, as fast as the state-of-the-art coordinate descent. Specifically, we show that SEGA with importance sampling and acceleration converges, up to a constant, as fast as the analogous version of CD.

3.3 The SEGA algorithm

In this section we introduce a learning process for estimating the gradient from the sketched information provided by (3.2); this will be used as a subroutine of SEGA.

Let x^k be the current iterate, and let h^k be the current estimate of the gradient of f . We then query the oracle, and receive new gradient information in the form of the sketched gradient (3.2). At this point, we would like to update h^k based on this new information. We do this using a *sketch-and-project* process [61, 62, 176]: we set h^{k+1} to be the closest vector to h^k satisfying (3.2):

$$\begin{aligned} h^{k+1} &= \arg \min_{h \in \mathbb{R}^d} \|h - h^k\|^2 \\ &\text{subject to } \mathbf{S}_k^\top h = \mathbf{S}_k^\top \nabla f(x^k). \end{aligned} \quad (3.3)$$

The closed-form solution of (3.3) is

$$h^{k+1} = h^k - \mathbf{Z}_k(h^k - \nabla f(x^k)) = (\mathbf{I} - \mathbf{Z}_k)h^k + \mathbf{Z}_k \nabla f(x^k), \quad (3.4)$$

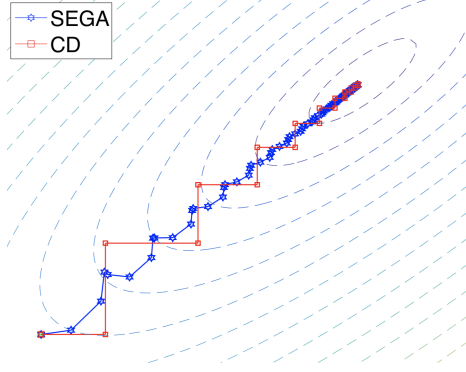


Figure 3.1: Iterates of SEGA and CD

where $\mathbf{Z}_k \stackrel{\text{def}}{=} \mathbf{S}_k (\mathbf{S}_k^\top \mathbf{S}_k)^\dagger \mathbf{S}_k^\top$. Notice that h^{k+1} is a *biased* estimator of $\nabla f(x^k)$. In order to obtain an unbiased gradient estimator, we introduce a random variable¹ $\theta_k = \theta(\mathbf{S}_k)$ for which

$$\mathbb{E} [\theta_k \mathbf{Z}_k] = \mathbf{I}. \quad (3.5)$$

If θ_k satisfies (3.5), it is straightforward to see that the random vector

$$g^k \stackrel{\text{def}}{=} (1 - \theta_k)h^k + \theta_k h^{k+1} \stackrel{(3.4)}{=} h^k + \theta_k \mathbf{Z}_k (\nabla f(x^k) - h^k) \quad (3.6)$$

is an *unbiased estimator* of the gradient:

$$\mathbb{E} [g^k] \stackrel{(3.5)+(3.6)}{=} \nabla f(x^k). \quad (3.7)$$

Finally, we use g^k instead of the true gradient, and perform a proximal step with respect to ψ . This leads to a new randomized optimization method, which we call *SkEtched Gradient Algorithm (SEGA)*. The method is formally described in Algorithm 6. We stress again that the method does not need the access to the full gradient.

Algorithm 6 SEGA (SkEtched Gradient Algorithm)

- 1: **Parameters:** $x^0, h^0 \in \mathbb{R}^d$; distribution \mathcal{D} ; stepsize $\alpha > 0$
 - 2: **for** $k = 0, 1, 2, \dots$ **do**
 - 3: Sample $\mathbf{S}_k \sim \mathcal{D}$
 - 4: $g^k = h^k + \theta_k \mathbf{Z}_k (\nabla f(x^k) - h^k)$
 - 5: $x^{k+1} = \text{prox}_{\alpha\psi}(x^k - \alpha g^k)$
 - 6: $h^{k+1} = h^k + \mathbf{Z}_k (\nabla f(x^k) - h^k)$
 - 7: **end for**
-

3.3.1 SEGA as a variance-reduced method

As we shall show, both h^k and g^k are becoming better at approximating $\nabla f(x^k)$ as the iterates x^k approach the optimum. Hence, the variance of g^k as an estimator of

¹Such a random variable may not exist. Some sufficient conditions are provided later.

the gradient tends to zero, which means that SEGA is a *variance-reduced* algorithm. The structure of SEGA is inspired by the JackSketch algorithm introduced in [65]. However, as JackSketch is aimed at solving a finite-sum optimization problem with many components, it does not make much sense to apply it to (3.1). Indeed, when applied to (3.1) (with $\psi \equiv 0$, since JackSketch was analyzed for smooth optimization only), JackSketch reduces to gradient descent. While JackSketch performs *Jacobian* sketching (i.e., multiplying the Jacobian by a random matrix from the right, effectively sampling a subset of the gradients forming the finite sum), SEGA multiplies the Jacobian by a random matrix from the left. In doing so, SEGA becomes oblivious to the finite-sum structure and transforms into the gradient sketching mechanism described in (3.2).

3.3.2 SEGA versus coordinate descent

We now illustrate the above general setup on the simple example when \mathcal{D} corresponds to a distribution over standard unit basis vectors in \mathbb{R}^d .

Example 3. Let \mathcal{D} be defined as follows. We choose $\mathbf{S}_k = e_i$ with probability $p_i > 0$, where e_1, e_2, \dots, e_d are the unit basis vectors in \mathbb{R}^d . Then

$$h^{k+1} \stackrel{(3.4)}{=} h^k + e_i^\top (\nabla f(x^k) - h^k) e_i, \quad (3.8)$$

which can equivalently be written as $h_i^{k+1} = e_i^\top \nabla f(x^k)$ and $h_j^{k+1} = h_j^k$ for $j \neq i$. If we choose $\theta_k = \theta(\mathbf{S}_k) = 1/p_i$, then

$$\mathbb{E}[\theta_k \mathbf{Z}_k] = \sum_{i=1}^d p_i \frac{1}{p_i} e_i (e_i^\top e_i)^{-1} e_i^\top = \sum_{i=1}^d e_i e_i^\top = \mathbf{I},$$

which means that θ_k is a bias-correcting random variable. We then get

$$g^k \stackrel{(3.6)}{=} h^k + \frac{1}{p_i} e_i^\top (\nabla f(x^k) - h^k) e_i. \quad (3.9)$$

In the setup of Example 3, both SEGA and CD obtain new gradient information in the form of a random partial derivative of f . However, the two methods process this information differently, and perform a different update:

- (i) While SEGA allows for arbitrary proximal term, CD allows for separable proximal term only [190, 122, 49].
- (ii) While SEGA updates all coordinates in every iteration, CD updates a single coordinate only.
- (iii) If we force $h^k = 0$ in SEGA and use coordinate sketches, the method transforms into CD.

Based on the above observations, we conclude that SEGA can be applied in more general settings for the price of potentially more expensive iterations². For intuition-

²Forming vector g and computing the prox.

building illustration of how SEGA works, Figure 3.1 shows the evolution of iterates of both SEGA and CD applied to minimizing a simple quadratic function in 2 dimensions. For more figures of this type, including the composite case where CD does not work, see Appendix 3.6.5.

In Section 3.5 we show that SEGA enjoys the same theoretical iteration complexity rates as CD, up to a small constant factor. This remains true when comparing state-of-the-art variants of CD utilizing importance-sampling, parallelism/mini-batching and acceleration with the appropriate corresponding variants of SEGA.

Remark 1. Nontrivial sketches S might, in some applications, bring a substantial speedup against the baseline choices mentioned in Example 3. Appendix C.3 provides one setting where this can happen: there are problems where the gradient of f always lies in a particular m -dimensional subspace of \mathbb{R}^d . In such a case, suitable choice of S leads to $\mathcal{O}\left(\frac{d}{m}\right)$ -times faster convergence compared to the setup of Example 3. In Section 3.6.3 we numerically verify this claim.

3.4 Convergence of SEGA for general sketches

In this section we state a linear convergence result for SEGA (Algorithm 6) for general sketch distributions \mathcal{D} under smoothness and strong convexity assumptions.

3.4.1 Smoothness assumptions

We will use the following general version of smoothness.

Assumption 3.4.1 (Q -smoothness). *Function f is Q -smooth for some $Q \succ 0$, that is, for all $x, y \in \mathbb{R}^d$, the following inequality is satisfied:*

$$f(x) - f(y) - \langle \nabla f(y), x - y \rangle \geq \frac{1}{2} \|\nabla f(x) - \nabla f(y)\|_Q^2. \quad (3.10)$$

Assumption 3.4.1 is not standard in the literature. However, as Lemma C.1.1 states, for twice differentiable f with $Q = M^{-1}$, Assumption 3.4.1 is equivalent to M -smoothness (see (2.6)), which is a common assumption in modern analysis of CD methods. As discussed in Chapter 2, M -smoothness appears naturally in various application such as empirical risk minimization with linear predictors and is a baseline in the development of minibatch CD methods [175, 166, 167, 168]. We will adopt this notion in Section 3.5, when comparing SEGA to coordinate descent. Until then, let us consider the almost equivalent Assumption 3.4.1.

3.4.2 Main result

We are now ready to present one of the key theorems of the chapter, which states that the iterates of SEGA converge linearly to the optimal solution.

Theorem 3.4.2. Assume that f is \mathbf{Q} -smooth and μ -strongly convex. Choose stepsize $\alpha > 0$ and Lyapunov parameter $\sigma > 0$ so that

$$\alpha(2(\mathbf{C} - \mathbf{I}) + \sigma\mu\mathbf{I}) \leq \sigma\mathbb{E}[\mathbf{Z}], \quad \alpha\mathbf{C} \leq \frac{1}{2}(\mathbf{Q} - \sigma\mathbb{E}[\mathbf{Z}]), \quad (3.11)$$

where $\mathbf{C} \stackrel{\text{def}}{=} \mathbb{E}[\theta_k^2 \mathbf{Z}_k]$. Fix $x^0, h^0 \in \text{dom}(F)$ and let x^k, h^k be the random iterates produced by SEGA. Then

$$\mathbb{E}[\Phi^k] \leq (1 - \alpha\mu)^k \Phi^0,$$

where $\Phi^k \stackrel{\text{def}}{=} \|x^k - x^*\|^2 + \sigma\alpha\|h^k - \nabla f(x^*)\|^2$ is a Lyapunov function and x^* is the solution of (3.1).

Note that the convergence of the Lyapunov function Φ^k implies both $x^k \rightarrow x^*$ and $h^k \rightarrow \nabla f(x^*)$. The latter means that SEGA is *variance reduced*, in contrast to CD in the proximal setup with non-separable ψ , which does not converge to the solution.

To clarify on the assumptions, let us mention that if σ is small enough so that $\mathbf{Q} - \sigma\mathbb{E}[\mathbf{Z}] \succ 0$, one can always choose stepsize α satisfying

$$\alpha \leq \min \left\{ \frac{\lambda_{\min}(\mathbb{E}[\mathbf{Z}])}{\lambda_{\max}(2\sigma^{-1}(\mathbf{C} - \mathbf{I}) + \mu\mathbf{I})}, \frac{\lambda_{\min}(\mathbf{Q} - \sigma\mathbb{E}[\mathbf{Z}])}{2\lambda_{\max}(\mathbf{C})} \right\} \quad (3.12)$$

and inequalities (3.11) will hold. Therefore, we get the next corollary.

Corollary 3.4.3. If $\sigma < \frac{\lambda_{\min}(\mathbf{Q})}{\lambda_{\max}(\mathbb{E}[\mathbf{Z}])}$, α satisfies (3.12) and $k \geq \frac{1}{\alpha\mu} \log \frac{\Phi^0}{\epsilon}$, then

$$\mathbb{E}[\|x^k - x^*\|^2] \leq \epsilon.$$

As Theorem 3.4.2 is rather general, we also provide a simplified version thereof, complete with a simplified analysis (Theorem C.4.1 in Appendix C.4). In the simplified version we remove the proximal setting (i.e., we set $\psi \equiv 0$), assume L -smoothness³, and only consider coordinate sketches with uniform probabilities. The result is provided as Corollary 3.4.4.

Corollary 3.4.4. Let \mathcal{D} be the uniform distribution over the standard unit basis vectors in \mathbb{R}^d . If the stepsize satisfies

$$0 < \alpha \leq \min \left\{ \frac{1 - \frac{L\sigma}{d}}{2Ld}, \frac{1}{n \left(\mu + \frac{2(d-1)}{\sigma} \right)} \right\},$$

then

$$\mathbb{E}[\Phi^k] \leq (1 - \alpha\mu)^k \Phi^0.$$

Therefore, the iteration complexity is $\tilde{\mathcal{O}}(dL/\mu)$.

³The standard L -smoothness assumption is a special case of \mathbf{M} -smoothness for $\mathbf{M} = L\mathbf{I}$ and special case of \mathbf{Q} -smoothness for $\mathbf{Q} = L^{-1}\mathbf{I}$.

	CD	SEGA
Nonaccelerated method importance sampling, $b = 1$	$\frac{\text{Tr}(\mathbf{M})}{\mu} \log \frac{1}{\epsilon}$ [152]	$8.55 \cdot \frac{\text{Tr}(\mathbf{M})}{\mu} \log \frac{1}{\epsilon}$
Nonaccelerated method arbitrary sampling	$\left(\max_i \frac{v_i}{p_i \mu}\right) \log \frac{1}{\epsilon}$ [175]	$8.55 \cdot \left(\max_i \frac{v_i}{p_i \mu}\right) \log \frac{1}{\epsilon}$
Accelerated method importance sampling, $b = 1$	$1.62 \cdot \frac{\sum_i \sqrt{\mathbf{M}_{ii}}}{\sqrt{\mu}} \log \frac{1}{\epsilon}$ [7]	$9.8 \cdot \frac{\sum_i \sqrt{\mathbf{M}_{ii}}}{\sqrt{\mu}} \log \frac{1}{\epsilon}$
Accelerated method arbitrary sampling	$1.62 \cdot \sqrt{\max_i \frac{v_i}{p_i^2 \mu}} \log \frac{1}{\epsilon}$ [78]	$9.8 \cdot \sqrt{\max_i \frac{v_i}{p_i^2 \mu}} \log \frac{1}{\epsilon}$

Table 3.1: Complexity results for coordinate descent (CD) and our sketched gradient method (SEGA), specialized to coordinate sketching, for \mathbf{M} -smooth and μ -strongly convex functions.

Remark 2. In the fully general setting, one might choose α to be bigger than bound (3.12), which depends on eigen properties of matrices $\mathbb{E}[\mathbf{Z}]$, \mathbf{C} , \mathbf{Q} , leading to a better overall complexity according to Corollary 3.4.3. However, in the simple case with $\mathbf{Q} = \mathbf{I}$ and $\mathbf{S}_k = e_{i_k}$ with uniform probabilities, bound (3.12) is tight.

3.5 Convergence of SEGA for coordinate sketches

In this section we compare SEGA with coordinate descent. We demonstrate that, specialized to a particular choice of the distribution \mathcal{D} (where \mathbf{S} is a random column submatrix of the identity matrix), which makes SEGA use the same random gradient information as that used in modern state-of-the-art randomized CD methods, SEGA attains, up to a small constant factor, the same convergence rate as CD methods.

Firstly, in Section 3.5.2 we develop SEGA with arbitrary “coordinate sketches” (Theorem 3.5.2). Then, in Section 3.5.3 we develop an *accelerated variant of SEGA* in a very general setup known as *arbitrary sampling* (see Theorem C.2.5) [175, 169, 166, 167]. Lastly, Corollary 3.5.3 and Corollary 3.5.4 provide us with *importance sampling* for both nonaccelerated and accelerated method, which matches up to a constant factor cutting-edge coordinate descent rates [175, 7] under the same oracle and assumptions⁴. Table 3.1 summarizes the results of this section. We provide a dedicated analysis for the methods from this section in Appendix C.2.

We now describe the setup and technical assumptions for this section. In order to facilitate a direct comparison with CD (which does not work with non-separable regularizer ψ), for simplicity we consider problem (3.1) in the simplified setting with $\psi \equiv 0$. Further, function f is assumed to be \mathbf{M} -smooth (2.6) and μ -strongly convex.

⁴There was recently introduced a notion of importance minibatch sampling for coordinate descent [78]. We state, without a proof, that SEGA with block coordinate sketches allows for the same importance sampling as developed in the mentioned chapter.

3.5.1 Defining \mathcal{D} : samplings

In order to draw a direct comparison with general variants of CD methods (i.e., with those analyzed in the *arbitrary sampling* paradigm), we consider sketches in (3.3) that are column submatrices of the identity matrix: $\mathbf{S} = \mathbf{I}_S$, where S is a random subset (aka *sampling*) of $[d] \stackrel{\text{def}}{=} \{1, 2, \dots, d\}$. Note that the columns of \mathbf{I}_S are the standard basis vectors e_i for $i \in S$ and hence

$$\mathbf{Range}(\mathbf{S}) = \mathbf{Range}(e_i : i \in S).$$

So, distribution \mathcal{D} from which we draw matrices is uniquely determined by the distribution of sampling S . Given a sampling S , define $p = (p_1, \dots, p_d) \in \mathbb{R}^d$ to be the vector satisfying $p_i = \mathbb{P}(e_i \in \mathbf{Range}(\mathbf{S})) = \mathbb{P}(i \in S)$, and \mathbf{P} to be the matrix for which

$$\mathbf{P}_{ij} = \mathbb{P}(\{i, j\} \subseteq S).$$

Note that p and \mathbf{P} are the *probability vector* and *probability matrix* of sampling S , respectively [167]. We assume throughout the chapter that S is proper, i.e., we assume that $p_i > 0$ for all i . State-of-the-art minibatch CD methods (including the ones we compare against [175, 78]) utilize large stepsizes related to the so-called ESO *Expected Separable Overapproximation* (ESO) [167] parameters $v = (v_1, \dots, v_d)$. ESO parameters play a key role in SEGA as well, and are defined next.

Assumption 3.5.1 (ESO). *There exists a vector v satisfying the following inequality*

$$\mathbf{P} \circ \mathbf{M} \preceq \mathbf{Diag}(p) \mathbf{Diag}(v), \quad (3.13)$$

where \circ denotes the Hadamard (i.e., element-wise) product of matrices.

In case of single coordinate sketches, parameters v are equal to coordinate-wise smoothness constants of f . An extensive study on how to choose them in general was performed in [167]. For notational brevity, let us set $\hat{\mathbf{P}} \stackrel{\text{def}}{=} \mathbf{Diag}(p)$ and $\hat{\mathbf{V}} \stackrel{\text{def}}{=} \mathbf{Diag}(v)$ throughout this section.

3.5.2 Non-accelerated method

We now state the convergence rate of (non-accelerated) SEGA for coordinate sketches with *arbitrary sampling* of subsets of coordinates. The corresponding CD method was developed in [175].

Theorem 3.5.2. *Assume that f is \mathbf{M} -smooth and μ -strongly convex. Denote $\Psi^k \stackrel{\text{def}}{=} f(x^k) - f(x^*) + \sigma \|h^k\|_{\hat{\mathbf{P}}^{-1}}^2$. Choose $\alpha, \sigma > 0$ such that*

$$\sigma \mathbf{I} - \alpha^2 (\hat{\mathbf{V}} \hat{\mathbf{P}}^{-1} - \mathbf{M}) \succeq \gamma \mu \sigma \hat{\mathbf{P}}^{-1}, \quad (3.14)$$

where $\gamma \stackrel{\text{def}}{=} \alpha - \alpha^2 \max_i \{ \frac{v_i}{p_i} \} - \sigma$. Then the iterates of SEGA satisfy

$$\mathbb{E} [\Psi^k] \leq (1 - \gamma \mu)^k \Psi^0.$$

We now give an importance sampling result for a coordinate version of SEGA. We recover, up to a constant factor, the same convergence rate as standard CD [152]. The probabilities we chose are optimal in our analysis and are proportional to the diagonal elements of matrix \mathbf{M} .

Corollary 3.5.3. *Assume that f is \mathbf{M} -smooth and μ -strongly convex. Suppose that \mathcal{D} is such that at each iteration standard unit basis vector e_i is sampled with probability $p_i \propto \mathbf{M}_{ii}$. If we choose $\alpha = \frac{0.232}{\text{Tr}(\mathbf{M})}$, $\sigma = \frac{0.061}{\text{Tr}(\mathbf{M})}$, then*

$$\mathbb{E} [\Psi^k] \leq \left(1 - \frac{0.117\mu}{\text{Tr}(\mathbf{M})}\right)^k \Psi^0.$$

The iteration complexities provided in Theorem 3.5.2 and Corollary 3.5.3 are summarized in Table 3.1. We also state that σ, α can be chosen so that (3.14) holds, and the rate from Theorem 3.5.2 coincides with the rate from Table 3.1.

Remark 3. Theorem 3.5.2 and Corollary 3.5.3 hold even under a non-convex relaxation of strong convexity – Polyak–Łojasiewicz inequality: $\mu(f(x) - f(x^*)) \leq \frac{1}{2}\|\nabla f(x)\|_2^2$. Therefore, SEGA also converges for a certain class of non-convex problems. For an overview on different relaxations of strong convexity, see [91].

3.5.3 Accelerated method

In this section, we propose an accelerated (in the sense of Nesterov’s method [149, 154]) version of SEGA, which we call ASEGA. The analogous accelerated CD method, in which a single coordinate is sampled in every iteration, was developed and analyzed in [7]. The general variant utilizing arbitrary sampling was developed and analyzed in [78].

Algorithm 7 ASEGA: Accelerated SEGA

- 1: **Parameters:** $x^0 = y^0 = z^0 \in \mathbb{R}^d$; $h^0 \in \mathbb{R}^d$; S ; parameters $\alpha, \beta, \eta, \mu > 0$
 - 2: **for** $k = 0, 1, 2, \dots$ **do**
 - 3: $x^k = (1 - \eta)y^{k-1} + \eta z^{k-1}$
 - 4: Sample $\mathbf{S}_k = \mathbf{I}_{S_k}$, where $S_k \sim S$, and compute g^k, h^{k+1} according to (3.4), (3.6)
 - 5: $y^k = x^k - \alpha \hat{\mathbf{P}}^{-1} g^k$
 - 6: $z^k = \frac{1}{1+\beta\mu}(z^k + \beta\mu x^k - \beta g^k)$
 - 7: **end for**
-

The method and analysis is inspired by [6]. Due to space limitations and technicality of the content, we state the main theorem of this section in Appendix C.2.4. Here, we provide Corollary 3.5.4, which shows that Algorithm 7 with single coordinate sampling enjoys, up to a constant factor, the same convergence rate as state-of-the-art accelerated coordinate descent method NUACDM of Allen-Zhu et al. [7].

Corollary 3.5.4. *Let the sampling be defined as follows: $S = \{i\}$ with probability $p_i \propto \sqrt{\mathbf{M}_{ii}}$, for $i \in [d]$. Then there exist acceleration parameters and a Lyapunov function Υ^k*

such that $f(y^k) - f(x^*) \leq \Upsilon^k$ and

$$\mathbb{E} [\Upsilon^k] \leq (1 - \eta)^k \Upsilon^0 = \left(1 - \mathcal{O}\left(\frac{\sqrt{\mu}}{\sum_i \sqrt{\mathbf{M}_{ii}}}\right)\right)^k \Upsilon^0.$$

The iteration complexity guarantees provided by Theorem C.2.5 and Corollary 3.5.4 are summarized in Table 3.1.

3.6 Experiments

In this section we perform numerical experiments to illustrate the potential of SEGA. Firstly, in Section 3.6.1, we compare it to projected gradient descent (PGD) algorithm. Then in Section 3.6.2, we study the performance of zeroth-order SEGA (when sketched gradients are being estimated through function value evaluations) and compare it to the analogous zeroth-order method. Next, in Section 3.6.3 we verify the claim from Remark 2 that in some applications, particular sketches might lead to a significantly faster convergence. Lastly, Section 3.6.4 demonstrates that SEGA is competitive to CD methods when $\psi \equiv 0$ as the results from Section 3.5 predict.

In the all experiments where theory-supported stepsizes were used – we obtained them by precomputing strong convexity and smoothness measures.

3.6.1 Comparison to projected gradient descent

In this experiment, we illustrate the potential superiority of our method to PGD. We consider the ℓ_2 ball constrained problem (ψ is the indicator function of the unit ball) with the oracle providing the sketched gradient in the random Gaussian direction. As we mentioned in the introduction, a method moving in the gradient direction (analogue of CD), will not converge due to the proximal nature of the problem. Therefore, we can only compare against the projected gradient. However, in order to obtain the full gradient, one needs to gather n sketched gradients and solve a linear system to recover the gradient. To illustrate this, we choose 4 different quadratic problems of the form

$$f(x) \stackrel{\text{def}}{=} \frac{1}{2} x^\top \mathbf{M} x - b^\top x,$$

where b is a random vector with independent entries from $\mathcal{N}(0, 1)$ and $\mathbf{M} \stackrel{\text{def}}{=} \mathbf{U} \Sigma \mathbf{U}^\top$ according to Table 3.2 for \mathbf{U} obtained from QR decomposition of random matrix with independent entries from $\mathcal{N}(0, 1)$. For each problem, the starting point was chosen to be a vector with independent entries from $\mathcal{N}(0, 1)$. We stress that these are synthetic problems generated for the purpose of illustrating the potential of our method against a natural baseline. Figure 3.2 compares SEGA and PGD under various relative cost scenarios of solving the linear system compared to the cost of the oracle calls. The results show that SEGA significantly outperforms PGD as soon as solving the linear system is expensive, and is as fast as PGD even if solving the linear system comes for free.

Type	Σ
1	Diagonal matrix with first $n/2$ components equal to 1, the rest equal to n
2	Diagonal matrix with first $n - 1$ components equal to 1, the last one equal to n
3	Diagonal matrix with i th component equal to i
4	Diagonal matrix with components coming from uniform distribution over $[0, 1]$

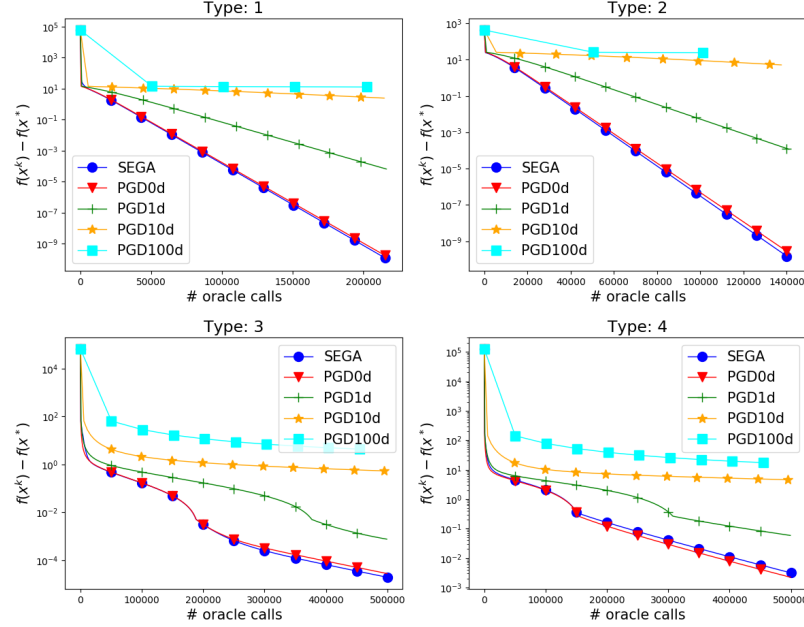
Table 3.2: Spectrum of \mathbf{M} .

Figure 3.2: Convergence of SEGA and PGD on synthetic problems with $d = 500$. The indicator “Xd” in the label indicates the setting where the cost of solving linear system is Xd times higher comparing to the cost of evaluating a single directional derivative. Recall that a linear system is solved after each d oracle calls. Stepsizes $1/\lambda_{\max}(\mathbf{M})$ and $1/(d\lambda_{\max}(\mathbf{M}))$ were used for PGD and SEGA, respectively.

3.6.2 Comparison to zeroth-order optimization methods

In this section, we compare SEGA to the *random direct search* (RDS) method [12] under a zeroth-order oracle for unconstrained optimization. For SEGA, we estimate the sketched gradient using finite differences. Note that RDS is a randomized version of the classical direct search method [84, 99, 100]. At iteration k , RDS moves to

$$\arg \min (f(x^k + \alpha^k s^k), f(x^k - \alpha^k s^k), f(x^k))$$

for a random direction $s^k \sim \mathcal{D}$ and a suitable stepsize α^k . For illustration, we choose f to be a quadratic problem based on Table 3.2 and compare both Gaussian and coordinate directions. Figure 3.3 shows that SEGA outperforms RDS.

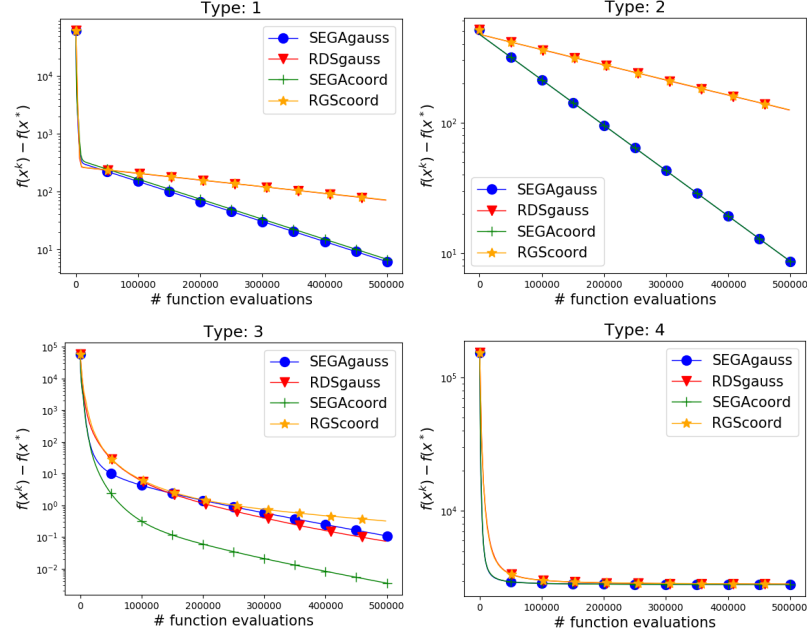


Figure 3.3: Comparison of SEGA and randomized direct search for various problems. Theory supported stepsizes were chosen for both methods. 500 dimensional problem.

3.6.3 Subspace SEGA

As mentioned in Remark 2, well designed sketches are capable of exploiting structure of f and lead to a better rate. We address this in detail Appendix C.3 where we develop and analyze a subspace variant of SEGA.

To illustrate this phenomenon in a simple setting, we perform experiments for problem (3.1) with $f(x) = \|\mathbf{A}x - b\|^2$, where $b \in \mathbb{R}^m$ and $\mathbf{A} \in \mathbb{R}^{m \times d}$ has orthogonal rows, and with ψ being the indicator function of the unit ball in \mathbb{R}^d . That is, we solve the problem

$$\min_{\|x\|_2 \leq 1} \|\mathbf{A}x - b\|^2.$$

We assume that $d \gg m$. We compare two methods: naiveSEGA, which uses coordinate sketches, and subspaceSEGA, where sketches are chosen as rows of \mathbf{A} . Figure 3.4 indicates that subspaceSEGA outperforms naiveSEGA roughly by the factor $\frac{d}{m}$, as claimed in Appendix C.3.

3.6.4 Comparison to randomized coordinate descent

In this section we numerically compare the results from Section 3.5 to analogous results for coordinate descent (as indicated in Table 3.1). We consider the ridge regression problem on LibSVM [23] data, for both primal and dual formulation. For all methods, we have chosen parameters as suggested from theory Figure 3.5 shows the results. We can see that in all cases, SEGA is slower to the corresponding coordinate descent method, but still is competitive. We however observe only constant times difference in terms of the speed, as suggested by Table 3.1.

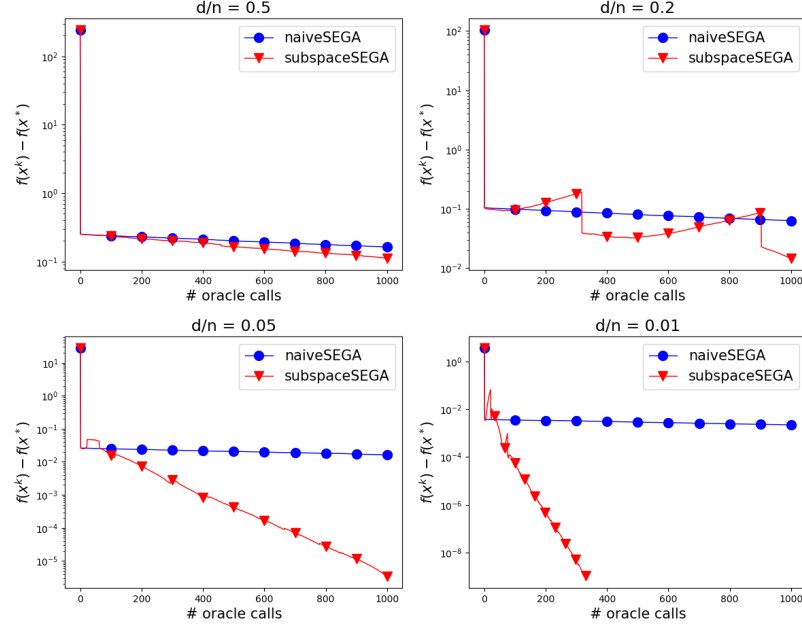


Figure 3.4: Comparison of SEGA with sketches from a correct subspace versus coordinate sketches naiveSEGA. Stepsize chosen according to theory. 1000 dimensional problem.

3.6.5 Evolution of iterates: Extra plots

Here we show some additional plots similar to Figure 3.1, which we believe help to build intuition about how the iterates of SEGA behave. We also include plots for biasSEGA, which uses biased estimators of the gradient instead. We found that the iterates of biasSEGA often behave in a more stable way, as could be expected given the fact that they enjoy lower variance. However, we do not have any theory supporting the convergence of biasSEGA; this is left for future research.

3.7 Conclusion

We proposed SEGA, a method for solving composite optimization problems under a novel stochastic linear first-order oracle. SEGA is variance-reduced, and this is achieved via sketch-and-project updates of gradient estimates. We provided an analysis for smooth and strongly convex functions and general sketches, and a refined analysis for coordinate sketches. For coordinate sketches we also proposed an accelerated variant of SEGA, and our theory matches that of state-of-the-art CD methods. However, in contrast to CD, SEGA can be used for optimization problems with a *non-separable* proximal term. We develop a more aggressive subspace variant of the method—subspaceSEGA—which leads to improvements in the $d \gg m$ regime. In the Appendix we give several further results, including simplified and alternative analyses of SEGA in the coordinate setup from Example 3. Our experiments are encouraging and substantiate our theoretical predictions.

Next, we point to several potential extensions of our work.

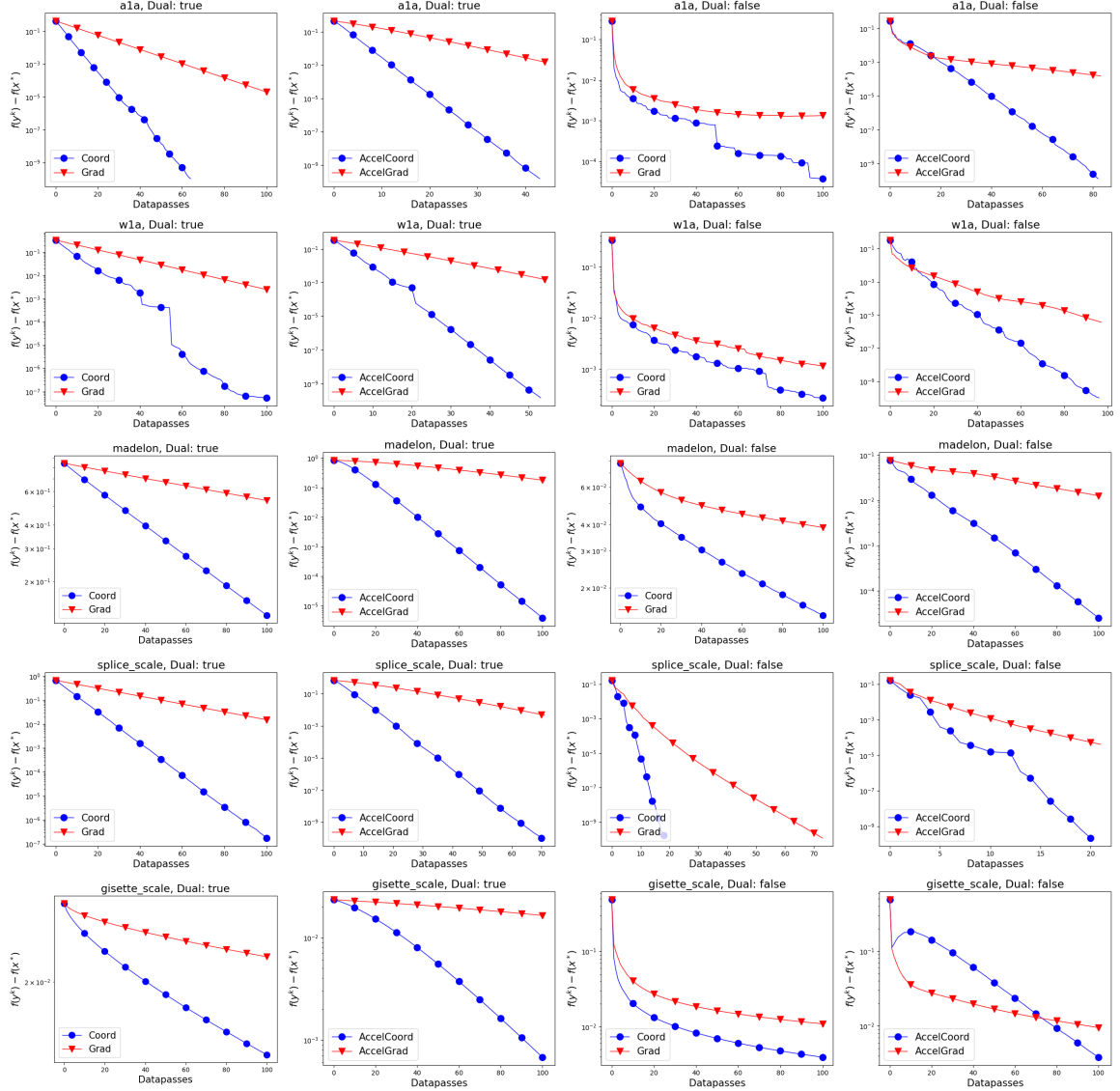


Figure 3.5: Comparison of SEGA and ASEG with corresponding coordinate descent methods for $\psi \equiv 0$.

Speeding up the general method. We believe that it should be possible to extend ASEG to the general setup from Theorem 3.4.2. In such a case, it might be possible a distribution of sketches \mathcal{D} so as to outperform accelerated proximal gradient methods [150, 9].

Biased gradient estimator. Recall that SEGA uses unbiased gradient estimator g^k for updating the iterates x^k in a similar way JacSketch [65] or SAGA [37] do this for the stochastic finite sum optimization. Recently, a stochastic method for finite sum optimization using biased gradient estimators was proven to be more efficient [160]. Therefore, it might be possible to establish better properties for a biased variant of SEGA. To demonstrate the potential of this approach, in Appendix 3.6.5 we plot the evolution of iterates

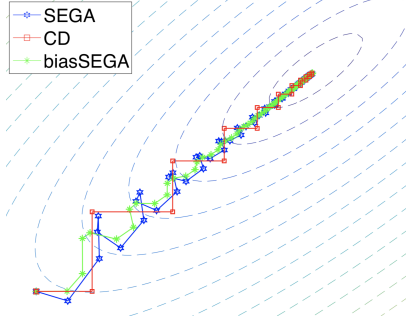


Figure 3.6: Evolution of iterates of SEGA, CD and biasSEGA (updates made via h^{k+1} instead of g^k).

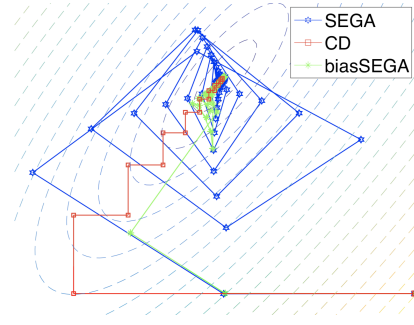


Figure 3.7: Iterates of SEGA, CD and biasSEGA (updates made via h^{k+1} instead of g^k). Different starting point.

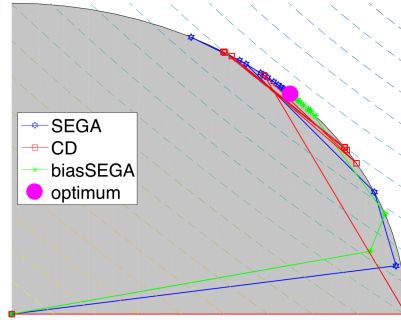


Figure 3.8: Iterates of projected SEGA, projected CD (which do not converge) and projected biasSEGA (updates made via h^{k+1} instead of g^k). The constraint set is represented by the shaded region.

for the very simple biased method which uses h^k as an update for line 3 in Algorithm 6.

Applications. We believe that SEGA might work well in applications where a zeroth-order approach is inevitable, such as reinforcement learning. We therefore believe that SEGA might be an efficient proximal method in some reinforcement learning applications. We also believe that communication-efficient variants of SEGA can be used for distributed training of machine learning models. This is because SEGA can be adapted to communicate sparse model updates only.

In the next chapter we introduce a different scenario where SEGA can be superior to CD even for problems without non-separable regularizer. The setup goes as follows: instead of minimizing a single function, we aim to minimize a finite sum. The oracle provides us with mutually independent random set of partial derivatives of each function from the sum. In such case, the gradient in the optimum does not have to be zero for each function, and thus SEGA trick might be necessary to keep fast convergence. However,

the motivation for the mentioned setup does not come from SEGA, but rather that the independent sampling of coordinates yields surprisingly fast convergence.

Chapter 4

99% of Worker-Master Communication in Distributed Optimization is Not Needed

In this work we are concerned with parallel/distributed algorithms for solving finite sum minimization problems

$$\min_{x \in \mathbb{R}^d} \left\{ f(x) \stackrel{\text{def}}{=} \frac{1}{n} \sum_{i=1}^n f_i(x) \right\}, \quad (4.1)$$

where each f_i is convex and smooth. In particular, we are interested in methods which employ n parallel units/workers/nodes/processors, each of which has access to a single function f_i and its gradients (or unbiased estimators thereof). Let x^* be an optimal solution of (4.1). In many practical scenarios, f_i is often of the form

$$f_i(x) = \mathbb{E}_{\xi} \phi_i(x; \xi), \quad (4.2)$$

where the expectation is with respect to a distribution of training examples stored locally at machine i . More typically, however, each machine contains a very large but finite number of examples (for simplicity, say there are l examples on each machine), and f_i is of the form

$$f_i(x) = \frac{1}{l} \sum_{j=1}^l f_{ij}(x). \quad (4.3)$$

In the rest of this section we provide some basic motivation and intuitions in support of our approach. To this purpose, assume, for simplicity of exposition, that f_i is of the finite-sum form (4.3). In typical modern machine learning workloads, the number of machines n is much smaller than the number of data points on each machine l . In a large scale regime (i.e., when the model size d , the number of data points nl , or both are large), problem (4.1) needs to be solved by a combination of efficient methods and modern hardware. In recent years there has been a lot of progress in designing new algorithms for solving this problem using techniques such as stochastic approximation [179], variance reduction [182, 88, 37], coordinate descent [152, 173, 215] and acceleration [149], resulting in excellent theoretical and practical performance.

The computational power of the hardware is increasing as well. In recent years, a very significant amount of such increase is due to parallelism. Since many methods, such as minibatch Stochastic Gradient Descent (SGD), are embarrassingly parallel, it is very simple to use them in big data applications. However, it has been observed in practice that adding more resources beyond a certain limit does not improve iteration complexity

significantly. Moreover, having more parallel units makes their synchronization harder due to so-called communication bottleneck. Minibatch versions of most variance reduced methods¹ such as SAGA [37] or SVRG [88] scale even worse in parallel setting – they do not guarantee, in the worst case, any speedup from using more than one function at a time. Unfortunately, numerical experiments show that this is not a proof flaw, but rather a real property of these methods [65]. A similar observation was made for SVRG by [224], where it was shown that only a small number of partial derivatives are needed at each iteration.

Since there are too many possible situations, we choose to focus on black-box optimization, although we admit that much can be achieved by assuming the sparsity structure. In fact, for any method there exists a toy situation where the method would scale perfectly – one simply needs to assume that each function f_i depends on its own subset of coordinates and minimize each f_i independently. This can be generalized assuming sparsity patterns [114, 115] to get almost linear scaling if any coordinate appears in a small number of functions. Our interest, however, is in explaining situations as in [65] where the models almost do not scale.

In this chapter, we demonstrate that a simple trick of *independent* block sampling can remedy the problem of scaling, to a substantial but limited extent. To illustrate one of the key insights on a simple example, in what follows consider a thought experiment in which GD is a baseline method we would want to improve on.

4.1 From gradient descent to block coordinate descent and back

A simple benchmark in the distributed setting is a parallel implementation of gradient descent (GD). GD arises as a special case of the more general class of block coordinate descent methods (BCD) [152]. The conventional way to run BCD for problem (4.1) is to update a single or several blocks² of x , chosen at random, on all n machines [152, 49], followed by an update aggregation step. Such updates on each worker typically involve a gradient step on a subspace corresponding to the selected blocks. Importantly, and this is a key structural property of BCD methods, *the same set of blocks is updated on each machine*. If communication is expensive, it often makes sense to do more work on each machine, which in the context of BCD means updating more blocks. A particular special case is to update *all* blocks, which leads to parallel implementation of GD for problem (4.1), as mentioned above. Moreover, it is known that the theoretical iteration complexity of BCD improves as the number of blocks updated increases [152, 166, 167]. For these and similar reasons, GD (or one of its variants, such as GD with momentum), is often the preferable method to BCD (in terms of iteration complexity). Having said that, we did not choose to describe BCD only to discard it at this point; we shall soon return to it, albeit with a twist.

¹We shall mention that there are already a few variance reduced methods that scale, up to some level, linearly in a parallel setup: Quartz for sparse data [169], Katyusha [4], or SAGA/SVRG/SARAH with importance sampling for non-convex problems [87].

²Assume the entries of x are partitioned into several non-overlapping blocks.

4.1.1 From gradient descent to independent block coordinate descent

Because of what we have just said, iteration complexity of GD will not improve by any variant running BCD; it can only get worse. Despite this, *we propose to run BCD, but a new variant which allows each worker to sample an independent subset of blocks* instead. This variant of BCD for (4.1) was not considered before. As we shall show, our *independent sampling* approach leads to a better-behaved aggregated gradient estimator when compared to that of BCD, which in turn leads to better overall iteration complexity. We call our method *independent block coordinate descent* (IBCD).

We provide a unified analysis of our method, allowing for a random subset of τm out of a total of m blocks to be sampled on each machine, independently from other machines. GD arises as a special case of this method by setting $\tau = 1$. However, as we show (see Corollary 4.4.3), *the same iteration complexity guarantee can be obtained by choosing τ as low as $\tau = \frac{1}{n}$* . The immediate consequence of this result is that *it is suboptimal to run GD in terms of communication complexity*. Indeed, GD needs to communicate all m blocks per machine, while IBCD achieves the same rate with $\frac{m}{n}$ blocks per machine only. Coming back to the abstract, consider an example with $n = 100$ machines. In this case, when compared to GD, IBCD only communicates 1% of the data. Because the iteration complexities of the two methods are the same, and if communication cost is dominant, this means that the problem can be solved in just 1% of the time. In contrast, and when compared to the potential of IBCD, parallel implementation of GD inevitably wastes 99% of the time.

The intuition behind why our approach works lies in the law of large numbers. By averaging independent noise we reduce the total variance of the resulting estimator by the factor of n . If, however, the noise is already tiny, as, in non-accelerated variance reduced methods, there is no improvement. On the other hand, (uniform) block coordinate descent (CD) has variance proportional to $\frac{1}{\tau}$ [212], where $\tau < 1$ is the ratio of used blocks. Therefore, after the averaging step the variance is $\frac{1}{\tau n}$, which illustrates why setting any $\tau > \frac{1}{n}$ should not yield a significant speedup when compared to the choice $\tau = \frac{1}{n}$. It also indicates that it should be possible to throw away a $(1 - \frac{1}{n})$ fraction of blocks while keeping the same convergence rate.

4.2 Contributions

The goal of the above discussion was to introduce one of the ideas of this chapter in a gentle way. However, our independent sampling idea has immense consequences beyond the realm of GD, as we show in the rest of the chapter. Let us summarize the contributions here:

- We show that the independent sampling idea can be coupled with variance reduction/SAGA (see Section 4.5), SGD for problem (4.1)+(4.2) (see Section 4.6), acceleration (under mild assumption on stochastic gradients; see Section 4.7) and regularization/SEGA (see Section 4.8). We call the new methods ISAGA, ISGD, IASGD and ISEGA, respectively. We also develop ISGD variant for asynchronous distributed optimization – IASGD (Section D.2).

- We present two versions of SAGA coupled with IBCD. The first one is for a distributed setting, where each machine owns a subset of data and runs a SAGA iteration with block sampling locally, followed by aggregation. The second version is in a shared data setting, where each machine has access to all functions. This allows for linear convergence even if $\nabla f_i(x^*) \neq 0$.
- We show that when combined with IBCD, the SEGA trick (Chapter 3) leads to a method that enjoys a linear rate for problems where $\nabla f_i(x^*) \neq 0$ and allows for more general objectives which may include a non-separable non-smooth regularizer.

A comprehensive summary of all algorithms proposed in this chapter is given in Table 4.1.

#	Name	Origin	$\nabla f_i(x^*) \neq 0$	Linear rate	Stochastic gradient	Note
8	IBCD	I + CD [152]	✗	✓	✗	Simplest
13	ISEGA	I + SEGA [152]	✓	✓	✗	Allows prox
25	IBGD	I + GD	✗	✓	✗	Bernoulli
9	ISAGA	+ SAGA [37]	✓	✓	✓	Shared memory
10	ISAGA	I + SAGA [37]	✗	✓	✓	
11	ISGD	I + SGD [179]	✓	✗	✓	+ Non-convex
12	IASGD	I + ASGD [208]	✓	✗	✓	Accelerated
26	IASGD	I + ASGD [170]	✓	✗	✓	Asynchronous

Table 4.1: Summary of all algorithms proposed in the chapter.

4.3 Practical implications and limitations

In this section, we outline some further limitations and practical implications of our framework.

4.3.1 Main limitation

The main limitation of this work is that independent sampling does not generally result in a sparse aggregated update. Indeed, since each machine might sample a different subset of blocks, all these updates add up to a dense one, and this problem gets worse as n increases, other things equal. For instance, if every parallel unit updates a single unique block³, the total number of updated blocks is equal n . In contrast, standard BCD, one that samples the *same* block on each worker, would update a single block only. For simple linear problems, such as logistic regression, sparse updates allow for a fast implementation of BCD via memorization of the residuals. However, this limitation is not crucial in common settings where broadcast is much faster than reduce.

³Assume x is partitioned into several “blocks” of variables.

4.3.2 Practical implications

The main body of this work focuses on theoretical analysis and on verifying our claims via experiments. However, there are several straightforward and important applications of our technique.

Distributed synchronous learning. A common way to run a distributed optimization method is to perform a local update, communicate the result to a parameter server using a reduce operation, and inform all workers using broadcast. Typically, if the number of workers is significantly large, the bottleneck of such a system is communication. In particular, the reduce operation takes much more time than broadcast as it requires to add up different vectors computed locally, while broadcast informs the workers about *the same* data (see [136] for a numerical validation that broadcast is 10-20 times faster across a wide range of dimensions). Nevertheless, if every worker can instead send to the parameter server only $\tau = \frac{1}{n}$ fraction of the d -dimensional update, essentially the server node will receive just one full d -dimensional vector, and thus our approach can compete against methods like QSGD [2], signSGD [13], TernGrad [213], DGC [123] or ATOMO [209]. In fact, our approach may completely remove the communication bottleneck.

Distributed asynchronous learning. The main difference with the synchronous case is that only one-to-one communications will be used instead of highly efficient reduce and broadcast. Clearly, the communication to the server will be much faster with $\tau = \frac{1}{n}$, so the main question is how to make the communication back fast as well. Hopefully, the parameter server can copy the current vector and send it using non-blocking communication, such as *isend()* in MPI4PY [33]. Then, the communication back will not prevent the server from receiving the new updates. We combine the IBCD approach with asynchronous updates, which leads to a new method: IASGD (Algorithm 26).

Distributed sparse learning. Large datasets, such as binary classification data from LibSVM, often have sparse gradients. In this case, the reduce operation is not efficient and one needs to communicate data by sending positions of nonzeros and their values. Moreover, as we prove later, one can use independent sampling with ℓ_1 -penalty, which makes the problem solution sparse. In that case, only communication from a worker to the parameter server is slow, so both synchronous and asynchronous methods gain in performance.

Methods with local subproblems. One can also try to extend our analysis to methods with exact block-coordinate minimization or primal-dual and proximal methods such as Point-SAGA [36], PDHG [22], DANE [193], etc. There, by restricting ourselves to a subset of coordinates, we may obtain a subproblem that is easier to solve by orders of magnitude.

Block-separable problems within machines. Given that the local problem on each machine is block coordinate-wise separable, partial derivative blocks can be evaluated $\frac{1}{\tau}$ times cheaper than the gradients. Thus, independent sampling improves scalability at no

cost. Such problems can be obtained considering the dual problem, as is done in [131], for example.

For a comprehensive list of frequently used notation that is specific to this chapter, see Table A.3 in the supplementary material.

4.4 Independent block coordinate descent

Before presenting the algorithm, we shall assume smoothness and convexity of the objective.

Assumption 4.4.1. *For every i , function f_i is convex, L -smooth while function f is μ -strongly convex.*

Let \mathbb{R}^d be partitioned into m blocks, u_1, \dots, u_m , of arbitrary sizes, so that the parameter space is $\mathbb{R}^{|u_1|} \times \dots \times \mathbb{R}^{|u_m|}$. For any vector $x \in \mathbb{R}^d$ and a set of blocks U we denote by x_U the vector that has the same coordinate as x in the set of blocks U and zeros elsewhere.

4.4.1 The IBCD algorithm

In order to provide a quick taste of our results, we first present the IBCD method described in the introduction and formalized as Algorithm 8.

Algorithm 8 Independent Block Coordinate Descent (IBCD)

- 1: **Input:** $x^0 \in \mathbb{R}^d$, partition of \mathbb{R}^d into m blocks u_1, \dots, u_m , ratio of blocks to be sampled τ , stepsize α , # of parallel units n
 - 2: **for** $k = 0, 1, 2, \dots$ **do**
 - 3: **for** $i = 1, \dots, n$ in parallel **do**
 - 4: Sample independently and uniformly a subset of τm blocks $U_i^k \subseteq \{u_1, \dots, u_m\}$
 - 5: $x_i^{k+1} = x^k - \alpha(\nabla f_i(x^k))_{U_i^k}$
 - 6: **end for**
 - 7: $x^{k+1} = \frac{1}{n} \sum_{i=1}^n x_i^{k+1}$
 - 8: **end for**
-

A key parameter of the method is $\frac{1}{m} \leq \tau \leq 1$ (chosen so that τm is an integer), representing a fraction of blocks to be sampled by each worker. At iteration k , each machine independently samples a subset of τm blocks $U_i^k \subseteq \{u_1, \dots, u_m\}$, uniformly at random. The i th worker then performs a subspace gradient step of the form $x_i^{k+1} = x^k - \alpha(\nabla f_i(x^k))_{U_i^k}$, where $\alpha > 0$ is a stepsize. Note that only coordinates of x^k belonging to U_i^k get updated. This is then followed by aggregating all n gradient updates: $x^{k+1} = \frac{1}{n} \sum_i x_i^{k+1}$.

4.4.2 Convergence of IBCD

Theorem 4.4.2 provides a convergence rate for Algorithm 8. Admittedly, the assumptions of Theorem 4.4.2 are somewhat restrictive; in particular, we require $\nabla f_i(x^*) = 0$

for all i . However, this is necessary. Indeed, in general one can not expect to have $\sum_{i=1}^n (\nabla f_i(x^*))_{U_i} = 0$ (which would be required for the method to converge to x^*) for independently sampled sets of blocks U_i unless $\nabla f_i(x^*) = 0$ for all i . As mentioned, the issue is resolved in Section 4.8 using the SEGA trick from Chapter 3.

Theorem 4.4.2. *Suppose that Assumptions 4.4.1 holds and $\nabla f_i(x^*) = 0$ for all i .⁴ For Algorithm 8 with $\alpha = \frac{n}{\tau n + 2(1-\tau)} \frac{1}{2L}$ we have*

$$\mathbb{E} [\|x^k - x^*\|^2] \leq \left(1 - \frac{\mu}{2L} \frac{\tau n}{\tau n + 2(1-\tau)}\right)^k \|x^0 - x^*\|^2.$$

As a consequence of Theorem 4.4.2, we can choose τ as small as $\frac{1}{n}$ and get, up to a constant factor, the same convergence rate as gradient descent, as described next.

Corollary 4.4.3. *If $\tau = \frac{1}{n}$, the iteration complexity⁵ of Algorithm 8 is $\mathcal{O}\left(\frac{L}{\mu} \log \frac{1}{\epsilon}\right)$.*

4.4.3 Optimal block sizes

If we naively use coordinates as blocks, i.e. all blocks have size equal 1, the update will be very sparse and the efficient way to send it is by providing positions of nonzeros and the corresponding values. If, however, we partition \mathbb{R}^d into blocks of size approximately equal d/n , then on average only one block will be updated by each worker. This means that it will be just enough for each worker to communicate the block number and its entries, which is twice less data sent than when using coordinates as blocks.

4.5 Variance reduction

As the first extension of IBCD, we inject independent coordinate sampling into SAGA⁶ [37], resulting in a new method we call ISAGA. We consider two different settings for ISAGA. The first one is standard distributed setup (4.1), where each f_i is of the fine-sum form (4.3). The idea is to run SAGA with independent coordinate sampling locally on each worker, followed by aggregating the updates. However, as for IBCD, we require $\nabla f_i(x^*) = 0$ for all i . The second setting is a *shared data/memory* setup; i.e., we assume that all workers have access to all functions from the finite sum.

⁴ The requirement of $\nabla f_i(x^*) = 0$ is only necessary for the plainest results; which we present to better explain the main idea of the chapter; and there are ways to go around it. In particular, in Section 4.5 we show that it can be dropped once the memory is shared among the machines. Further, in Section 4.8 we show that $\nabla f_i(x^*) = 0$ can be dropped even in the fully distributed setup using the SEGA trick. Lastly, $\nabla f_i(x^*) = 0$ is naturally satisfied in many applications. For example, in least squares setting $\min \|Ax - b\|^2$, it is equivalent to existence of x^* such that $Ax^* = b$. On the other hand, current state-of-the-art deep learning models are often overparameterized so that they allow zero training loss, which is again equivalent to $\nabla f_i(x^*) = 0$ for all i (however, such problems are typically non-convex).

⁵Number of iterations to reach ϵ accurate solution.

⁶Independent coordinate sampling is not limited to SAGA and can be similarly applied to other variance reduction techniques.

4.5.1 Shared data ISAGA

We now present a different setup for ISAGA in which the requirement $\nabla f_i(x^*) = 0$ is not needed. Instead of (4.1), we rather solve the problem

$$\min_{x \in \mathbb{R}^d} \left\{ f(x) \stackrel{\text{def}}{=} \frac{1}{N} \sum_{j=1}^N f'_j(x) \right\} \quad (4.4)$$

with n workers all of which have access to all data describing f . Therefore, all workers can evaluate $\nabla f'_j(x)$ for any $1 \leq j \leq N$. Similarly to plain SAGA, we remember the freshest gradient information in table \mathbf{J} , which we update as follows:

$$\mathbf{J}_{j_i^k}^{k+1} = \mathbf{J}_{j_i^k}^k + (\nabla f'_{j_i^k}(x^k) - \mathbf{J}_{j_i^k}^k)_{U_i^k}, \quad \mathbf{J}_{j'}^{k+1} = \mathbf{J}_{j'}^k, \quad (4.5)$$

where j_i^k is the index sampled at iteration k by machine i , and j' refers to all indices that were not sampled at iteration k by any machine. The iterate updates within each machine are taken only on a sampled set of coordinates, i.e., $x_i^{k+1} = x^k - \alpha(\nabla f'_{j_i^k}(x^k) - \mathbf{J}_{j_i^k}^k + \bar{\mathbf{J}}^k)_{U_i^k}$, where $\bar{\mathbf{J}}^k$ stands for the average of all \mathbf{J} , and thus it is a delayed estimate of $\nabla f(x^k)$. Lastly, we set the next iterate as the average of proposed iterates by each machine $x^{k+1} = \frac{1}{n} \sum_{i=1}^n x_i^{k+1}$. The formal statement of the algorithm is given in the supplementary as Algorithm 9.

Algorithm 9 ISAGA with shared data

- 1: **Input:** $x^0 \in \mathbb{R}^d$, $\mathbf{J}_1^0, \dots, \mathbf{J}_N^0$ partition of \mathbb{R}^d into m blocks u_1, \dots, u_m , ratio of blocks to be sampled τ , stepsize α , # parallel units n
 - 2: Set $\bar{\mathbf{J}}^0 \stackrel{\text{def}}{=} \frac{1}{N} \sum_{i=1}^N \mathbf{J}_i^0$
 - 3: **for** $k = 0, 1, 2, \dots$ **do**
 - 4: Sample uniformly set of indices $\{j_1^k, \dots, j_n^k\} \subseteq \{1, \dots, N\}$ without replacement
 - 5: **for** $i = 1, \dots, n$ **in parallel do**
 - 6: Sample independently and uniformly a subset of τm blocks U_t^i
 - 7: $x_i^{k+1} = x^k - \alpha(\nabla f'_{j_i^k}(x^k) - \mathbf{J}_{j_i^k}^k + \bar{\mathbf{J}}^k)_{U_i^k}$
 - 8: $(\mathbf{J}_{j_i^k}^{k+1})_{U_i^k} = \mathbf{J}_{j_i^k}^k + (\nabla f'_{j_i^k}(x^k) - \mathbf{J}_{j_i^k}^k)_{U_i^k}$
 - 9: **end for**
 - 10: For $j \notin \{j_1^k, \dots, j_n^k\}$ set $(\mathbf{J}_j^{k+1}) = \mathbf{J}_j^k$
 - 11: $x^{k+1} = \frac{1}{n} \sum_{i=1}^n x_i^{k+1}$
 - 12: $\bar{\mathbf{J}}^{k+1} = \frac{1}{n} \sum_{j=1}^N \mathbf{J}_j^{k+1}$
 - 13: **end for**
-

Theorem 4.5.1. Suppose that function f is μ -strongly convex and each f'_i is L smooth and convex. If $\alpha \leq \frac{1}{L(\frac{3}{n} + \tau)}$, then for iterates of Algorithm 9 we have

$$\mathbb{E} [\|x^k - x^*\|^2] \leq (1 - \vartheta)^k (\|x^0 - x^*\|^2 + c\alpha^2 \Psi^0),$$

where $\Psi^0 \stackrel{\text{def}}{=} \sum_j \|\mathbf{J}_j^0 - \nabla f_j'(x^*)\|^2$, $\vartheta \stackrel{\text{def}}{=} \tau \min \left\{ \alpha\mu, \frac{n}{N} - \frac{2}{nNc} \right\} \geq 0$ and $c \stackrel{\text{def}}{=} \frac{1}{n} \left(\frac{1}{\alpha L} - \frac{1}{n} - \tau \right) > 0$.

As in Section 4.5.2, the choice $\tau = \frac{1}{n}$ yields a convergence rate which is, up to a constant factor, the same as the convergence rate of SAGA. Therefore, Algorithm 9 enjoys the desired parallel linear scaling without extra assumptions. Corollary 4.5.2 formalizes the claim.

Corollary 4.5.2. *Consider the setting from Theorem 4.5.1. Set $\tau = \frac{1}{n}$ and $\alpha = \frac{n}{5L}$. Then $c = \frac{3}{n^2}$, $\rho = \min \left\{ \frac{\mu}{5L}, \frac{1}{3N} \right\}$ and the complexity of Algorithm 9 is $O \left(\max \left\{ \frac{L}{\mu}, N \right\} \log \frac{1}{\varepsilon} \right)$.*

4.5.2 Distributed ISAGA

In this section we consider problem (4.1) with f_i of the finite-sum structure (4.3). Just like SAGA, every machine remembers the freshest gradient information of all local functions (stored in arrays \mathbf{J}_{ij}), and updates them once a new gradient information is observed. Given that index j_i^k is sampled on i th machine at iteration k , the iterate update step within each machine is taken only on a sampled set of coordinates:

$$x_i^{k+1} = x^k - \alpha (\nabla f_{ij_i^k}(x^k) - \mathbf{J}_{ij_i^k}^k + \bar{\mathbf{J}}_i^k)_{U_i^k}.$$

Above, $\bar{\mathbf{J}}_i^k$ stands for the average of \mathbf{J} variables on i th machine, i.e. it is a delayed estimate of $\nabla f_i(x^k)$. Since the new gradient information is a set of partial derivatives of $\nabla f_{ij_i^k}(x^k)$, we shall update

$$\mathbf{J}_{ij}^{k+1} = \begin{cases} \mathbf{J}_{ij}^k + (\nabla f_{ij}(x^k) - \mathbf{J}_{ij}^k)_{U_i^k} & j = j_i^k \\ \mathbf{J}_{ij}^k & j \neq j_i^k \end{cases} \quad (4.6)$$

Lastly, the local results are aggregated. See Algorithm 10 for details.

The next result provides a convergence rate of distributed ISAGA.

Theorem 4.5.3. *Suppose that Assumption 4.4.1 holds and $\nabla f_i(x^*) = 0$ for all i . If $\alpha \leq \frac{1}{L(\frac{3}{n} + \tau)}$, for iterates of distributed ISAGA we have*

$$\mathbb{E} \|x^k - x^*\|^2 \leq (1 - \vartheta)^k (\|x^0 - x^*\|^2 + c\alpha^2 \Psi^0),$$

where $\Psi^0 \stackrel{\text{def}}{=} \sum_{i=1}^n \sum_{j=1}^l \|\mathbf{J}_{ij}^k - \nabla f_{ij}(x^*)\|^2$, $\vartheta \stackrel{\text{def}}{=} \tau \min \left\{ \alpha\mu, \frac{1}{l} - \frac{2}{n^2lc} \right\} \geq 0$ and $c \stackrel{\text{def}}{=} \frac{1}{n} \left(\frac{1}{\alpha L} - \frac{1}{n} - \tau \right) > 0$.

The choice $\tau = n^{-1}$ yields a convergence rate which is, up to a constant factor, the same as convergence rate of original SAGA. Thus, distributed ISAGA enjoys the desired parallel linear scaling. Corollary 4.5.4 formalizes this claim.

Corollary 4.5.4. *Consider the setting from Theorem 4.5.3. Set $\tau = \frac{1}{n}$ and $\alpha = \frac{n}{5L}$. Then $c = \frac{3}{n^2}$, $\rho = \min \left\{ \frac{\mu}{5L}, \frac{1}{3nl} \right\}$ and the complexity of distributed ISAGA is*

$$\mathcal{O} \left(\max \left\{ \frac{L}{\mu}, nl \right\} \log \frac{1}{\varepsilon} \right).$$

Algorithm 10 Distributed ISAGA

```

1: Input:  $x^0 \in \mathbb{R}^d$ , # parallel units  $n$ ,  $i$ th unit owns  $l$  functions  $f_{i1}, \dots, f_{il}$ , partition
   of  $\mathbb{R}^d$  into  $m$  blocks  $u_1, \dots, u_m$ , ratio of blocks to be sampled  $\tau$ , stepsize  $\alpha$ , initial
   vectors  $\mathbf{J}_{ij}^0 \in \mathbb{R}^d$  for  $1 \leq i \leq n, 1 \leq j \leq l$ 
2: Set  $\bar{\mathbf{J}}^0 \stackrel{\text{def}}{=} \frac{1}{N} \sum_{i=1}^n \mathbf{J}_i^0$ 
3: for  $k = 0, 1, 2, \dots$  do
4:   for  $i = 1, \dots, n$  in parallel do
5:     Sample independently & uniformly  $j_i^k \in [l]$ 
6:     Sample independently & uniformly a subset of  $\tau m$  blocks  $U_t^i$ 
7:      $x_i^{k+1} = x^k - \alpha(\nabla f_{ij_i^k}(x^k) - \mathbf{J}_{ij_i^k}^k + \bar{\mathbf{J}}_i^k)_{U_i^k}$ 
8:      $\mathbf{J}_{ij_i^k}^{k+1} = \mathbf{J}_{ij_i^k}^k + (\nabla f_{ij_i^k}(x^k) - \mathbf{J}_{ij_i^k}^k)_{U_i^k}$ 
9:     For any  $j \neq j_i^k$  set  $\mathbf{J}_{ij}^{k+1} = \mathbf{J}_{ij}^k$ 
10:     $\bar{\mathbf{J}}^{k+1} = \frac{1}{l} \sum_{j=1}^l \mathbf{J}_{ij}^{k+1}$ 
11:   end for
12:    $x^{k+1} = \frac{1}{n} \sum_{i=1}^n x_i^{k+1}$ 
13: end for

```

4.6 SGD

In this section, we apply independent sampling in a setup with a stochastic objective. In particular, we consider problem (4.1) where f_i is given as an expectation; see (4.2). We assume we have access to a stochastic gradient oracle which, when queried at x^k , outputs a random vector g_i^k whose mean is $\nabla f_i(x^k)$: $\mathbb{E}g_i^k = \nabla f_i(x^k)$.

Our proposed algorithm—ISGD—evaluates a subset of stochastic partial derivatives for the local objective and takes a step in the given direction for each machine. Next, the results are averaged and followed by the next iteration. We stress that the coordinate blocks have to be sampled independently within each machine.

Algorithm 11 ISGD

```

1: Input:  $x^0 \in \mathbb{R}^d$ , partition of  $\mathbb{R}^d$  into  $m$  blocks  $u_1, \dots, u_m$ , ratio of blocks to be
   sampled  $\tau$ , stepsize sequence  $\{\alpha^k\}_{k=1}^\infty$ , # parallel units  $n$ 
2: for  $k = 0, 1, 2, \dots$  do
3:   for  $i = 1, \dots, n$  in parallel do
4:     Sample independently and uniformly a subset of  $\tau m$  blocks  $U_i^k \subseteq \{u_1, \dots, u_m\}$ 
5:     Sample blocks of stochastic gradient  $(g_i^k)_{U_i^k}$  such that  $\mathbb{E}[g_i^k | x^k] = \nabla f_i(x^k)$ 
6:      $x_i^{k+1} = x^k - \alpha^k (g_i^k)_{U_i^k}$ 
7:   end for
8:    $x^{k+1} = \frac{1}{n} \sum_{i=1}^n x_i^{k+1}$ 
9: end for

```

In order to establish a convergence rate of ISGD, we shall assume boundedness of stochastic gradients for each worker.

Assumption 4.6.1. Consider a sequence of iterates $\{x^k\}_{k=0}^\infty$ of Algorithm 11. Assume that g_i^k is an unbiased estimator of $\nabla f_i(x^k)$ satisfying $\mathbb{E}\|g_i^k - \nabla f_i(x^k)\|^2 \leq \sigma^2$.

Assumption 4.6.2. Stochastic gradients of function f_i have bounded variance at the optimum of f : $\mathbb{E}\|g_i - \nabla f_i(x^*)\|^2 \leq \sigma^2$, where g_i is a random vector such that $\mathbb{E}g_i = \nabla f_i(x^*)$.

Next, we present the convergence rate of Algorithm 11. Since SGD is not a variance reduced algorithm, it does not enjoy a linear convergence rate and one shall use decreasing step sizes. As a consequence, it is not required to assume that $\nabla f_i(x^*) = 0$ for all i since there is no variance reduction property to be broken.

Theorem 4.6.3. Let Assumptions 4.4.1 and 4.6.1 hold. If $\alpha^k = \frac{1}{a+ck}$, where $a = 2\left(\tau + \frac{2(1-\tau)}{n}\right)L$, $c = \frac{1}{4}\mu\tau$, then for Algorithm 11 we can upper bound $\mathbb{E}[f(\hat{x}^k) - f(x^*)]$ by

$$\frac{a^2 \left(1 - \frac{\tau\mu}{a}\right) \|x^0 - x^*\|^2}{\tau(k+1)a + \frac{c\tau}{2}k(k+1)} + \frac{\sigma^2 + (1-\tau)\frac{2}{n} \sum_{i=1}^n \|\nabla f_i(x^*)\|^2}{n \left(1 + \frac{1}{k}\right) a + \frac{nc}{2}(k+1)},$$

where $\hat{x}^k \stackrel{\text{def}}{=} \frac{1}{(k+1)a + \frac{c}{2}k(k+1)} \sum_{t=0}^k (\alpha^t)^{-1} x^t$.

Note that the residuals decrease as $\mathcal{O}(k^{-1})$, which is a behavior one expects from standard SGD. Moreover, the leading complexity term scales linearly: if the number of workers n is doubled, one can afford to halve τ to keep the same complexity.

Corollary 4.6.4. Consider the setting from Theorem 4.6.3. Then, iteration complexity of Algorithm 11 is

$$\mathcal{O}\left(\frac{\sigma^2 + \frac{1}{n} \sum_{i=1}^n \|\nabla f_i(x^*)\|^2}{n\tau\mu\epsilon}\right).$$

Although problem (4.1) explicitly assumes convex f_i , we also consider a non-convex extension, where smoothness of each individual f_i is not required either. Theorem 4.6.5 provides the result.

Theorem 4.6.5 (Non-convex rate). Assume f is L smooth, Assumption 4.6.1 holds and for all $x \in \mathbb{R}^d$ the difference between gradients of f and f_i 's is bounded: $\frac{1}{n} \sum_{i=1}^n \|\nabla f(x) - \nabla f_i(x)\|^2 \leq \nu^2$ for some constant $\nu \geq 0$. If \hat{x}^k is sampled uniformly from $\{x^0, \dots, x^k\}$, then for Algorithm 11 we have

$$\mathbb{E}\|\nabla f(\hat{x}^k)\|^2 \leq \frac{\frac{f(x^0) - f^*}{k\tau\alpha} + \alpha L \frac{(1-\tau)\nu^2 + \frac{1}{2}\sigma^2}{n}}{1 - \frac{\alpha\tau L}{2} - \alpha L (1-\tau) \frac{1}{n}}.$$

Again, the convergence rate from Theorem 4.6.5 scales almost linearly with τ : with doubling the number of workers one can afford to halve τ to keep essentially the same guarantees. Note that if n is sufficiently large, increasing τ beyond a certain threshold does not improve convergence. This is a slightly weaker conclusion to the rest of our

results where increasing τ beyond n^{-1} might still offer speedup. The main reason behind this is the fact that SGD may be noisy enough on its own to still benefit from the averaging step.

Corollary 4.6.6. *Consider the setting from Theorem 4.6.5. i) Choose $\tau \geq \frac{1}{n}$ and $\alpha = \frac{\sqrt{n}}{L\sqrt{\tau k}} \leq \frac{1}{2L(\tau/2 + (1-\tau)/n)}$. Then*

$$\mathbb{E}\|\nabla f(\hat{x}^k)\|^2 \leq \frac{2}{\sqrt{k\tau n}} \left(\frac{f(x^0) - f^*}{L} + (1-\tau)\nu^2 \right) = O\left(\frac{1}{\sqrt{k}}\right).$$

ii) For any τ there is sufficiently large n such that choosing $\alpha = O\left(\frac{\epsilon}{\tau L^2}\right)$ yields complexity $O\left(\frac{L^2}{\epsilon^2}\right)$. The complexity does not improve significantly when τ is increased.

4.7 Acceleration

Here we describe an accelerated variant of IBCD in the sense of [149]. In fact, we will do something more general and accelerate ISGD, obtaining the IASGD algorithm. We again assume that machine i owns f_i , which is itself a stochastic objective as in (4.2) with an access to an unbiased stochastic gradient g^k every iteration: $\mathbb{E}g_i^k = \nabla f_i(x^k)$. A key assumption for the accelerated SGD used to derive the best known rates [208] is so the called strong growth of the unbiased gradient estimator.

Definition 4.7.1. *Function $\phi(x) = \mathbb{E}_\zeta \phi(x, \zeta)$ satisfies the strong growth condition with parameters ρ, σ^2 , if for all x we have*

$$\mathbb{E}_\zeta \|\nabla \phi(x, \zeta)\|^2 \leq \rho \|\nabla \phi(x)\|^2 + \sigma^2.$$

In order to derive a strong growth property of the gradient estimator coming from the independent block coordinate sampling, we require a strong growth condition on f with respect to f_1, \dots, f_n and also a variance bound on stochastic gradients of each individual f_i .

Assumption 4.7.2. *Function f satisfies the strong growth condition with respect to f_1, \dots, f_n :*

$$\frac{1}{n} \sum_{i=1}^n \|\nabla f_i(x)\|^2 \leq \tilde{\rho} \|\nabla f(x)\|^2 + \tilde{\sigma}^2. \quad (4.7)$$

Similarly, given that $g_i = g_i(x)$ provides an unbiased estimator of $\nabla f_i(x)$, i.e. $\mathbb{E}g_i = \nabla f_i(x)$, variance of g_i is bounded as follows for all i :

$$\mathbf{V}[g_i] \leq \bar{\rho} \|\nabla f_i(x)\|^2 + \bar{\sigma}^2. \quad (4.8)$$

Note that the variance bound (4.8) is weaker than the strong growth property as we always have $\mathbf{V}[g_i] \leq \mathbb{E}[\|g_i\|^2]$.

Given that Assumption 4.7.2 is satisfied, we derive a strong growth property for the unbiased gradient estimator $q \stackrel{\text{def}}{=} \frac{1}{n\tau} \sum_{i=1}^n (\nabla g_i)_{U_i}$ in Lemma 4.7.3. Next, IASGD is nothing

but the scheme from [208] applied to stochastic gradients q . For completeness, we state IASGD as Algorithm 12.

Algorithm 12 IASGD

```

1: Input: Starting point  $y^0 = v^0 \in \mathbb{R}^d$ , partition of  $\mathbb{R}^d$  into  $m$  blocks  $u_1, \dots, u_m$ ,
   ratio of blocks to be sampled  $\tau$ , stepsize  $\alpha$ , number of parallel units  $n$ , acceleration
   parameter sequences  $\{a, b, \eta\}_{k=0}^\infty$ 
2: for  $k = 0, 1, 2, \dots$  do
3:    $x^k = a^k v^k + (1 - a^k) y^k$ 
4:   for  $i = 1, \dots, n$  in parallel do
5:     Sample independently and uniformly a subset of  $\tau m$  blocks  $U_i^k \subset \{u_1, \dots, u_m\}$ 
6:     Sample blocks of stochastic gradient  $(g_i^k)_{U_i^k}$  such that  $\mathbb{E}[g_i^k | x^k] = \nabla f_i(x^k)$ 
7:   end for
8:    $q^k = \frac{1}{n\tau} \sum_{i=1}^n (g_i^k)_{U_i^k}$ 
9:    $y^{k+1} = x^k - \alpha q^k$ 
10:   $v^{k+1} = b^k v^k + (1 - b^k) x^k - \eta^k \gamma q^k$ .
11: end for

```

Lemma 4.7.3. *Suppose that Assumption 4.7.2 is satisfied. Then, we have $\mathbb{E}[\|q\|^2] \leq \hat{\rho} \|\nabla f(x)\|^2 + \hat{\sigma}^2$ for*

$$\hat{\rho} \stackrel{\text{def}}{=} \left(1 + \frac{\bar{\rho}}{n} \left(\frac{1}{\tau} - 1 + \frac{\bar{\rho}}{\tau} \right) \right), \quad (4.9)$$

$$\hat{\sigma}^2 \stackrel{\text{def}}{=} \frac{\bar{\sigma}^2}{n\tau} + \frac{\tilde{\sigma}^2}{n} \left(\frac{1}{\tau} - 1 + \frac{\bar{\rho}}{\tau} \right). \quad (4.10)$$

It remains to use the stochastic gradient q (with the strong growth bound from Lemma 4.7.3) as a gradient estimate in [208][Theorem 6], which we restate as Theorem 4.7.4 for completeness.

Theorem 4.7.4. *Suppose that f is L smooth, μ strongly convex and Assumption 4.7.2 holds. Then, for a specific choice of parameter sequences $\{a, b, \eta\}_{k=0}^\infty$ (See [208][Theorem 6] for details), iterates of IASGD admit an upper bound on $\mathbb{E}[f(x^{k+1})] - f(x^*)$ of the form*

$$\left(1 - \sqrt{\frac{\mu}{L\hat{\rho}^2}} \right)^k \left(f(x^0) - f(x^*) + \frac{\mu}{2} \|x^0 - x^*\|^2 \right) + \frac{\hat{\sigma}^2}{\hat{\rho}\sqrt{L\mu}}.$$

The next corollary provides a complexity of Algorithm 12 in a simplified setting where $\bar{\sigma}^2 = \tilde{\sigma}^2 = 0$. Note that $\tilde{\sigma}^2 = 0$ implies $\nabla f_i(x^*) = 0$ for all i . It again shows a desired linear scaling: given that we double the number of workers, we can halve the number of blocks to be evaluated on each machine and still keep the same convergence guarantees. It also shows that increasing τ beyond $\frac{\bar{\rho}}{n}$ does not improve the convergence significantly.

Corollary 4.7.5. *Suppose that $\bar{\sigma}^2 = \tilde{\sigma}^2 = 0$. Then, complexity of IASGD is*

$$\mathcal{O}\left(\frac{1}{\hat{\rho}}\sqrt{\frac{\mu}{L}}\log\frac{1}{\epsilon}\right) = \mathcal{O}\left(\frac{1}{1 + \frac{\hat{\rho}}{\tau n}(1 + \bar{\rho})}\sqrt{\frac{\mu}{L}}\log\frac{1}{\epsilon}\right).$$

Theorem 4.7.4 shows an accelerated rate for strongly convex functions applying [208, Theorem 6] to the bound. A non-strongly convex rate can be obtained analogously from [208, Theorem 7].

4.8 Beyond interpolation without shared data and regularization

For this section only, let us consider a regularized objective of the form

$$\min_{x \in \mathbb{R}^d} \left\{ f(x) \stackrel{\text{def}}{=} \frac{1}{n} \sum_{i=1}^n f_i(x) + \psi(x) \right\}, \quad (4.11)$$

where ψ is a closed convex regularizer such that its proximal operator,

$$\text{prox}_{\alpha\psi}(x) \stackrel{\text{def}}{=} \arg \min_y \left\{ \psi(y) + \frac{1}{2\alpha} \|y - x\|^2 \right\},$$

is computable. In this section we propose ISEGA: an independent sampling variant of SEGA. We do this in order to both i) avoid assuming $\nabla f_i(x^*) = 0$ (while keeping linear convergence) and ii) allow for R . Original SEGA learns gradients $\nabla f(x^k)$ from sketched gradient information via the so called sketch-and-project process [61], constructing a vector sequence h^k . In ISEGA on each machine i we iteratively construct a sequence of vectors h_i^k which play the role of estimates of $\nabla f_i(x^k)$. This is done via the following rule:

$$h_i^{k+1} = h_i^k + (\nabla f_i(x^k) - h_i^k)_{U_i^k}. \quad (4.12)$$

The key idea is again that these vectors are created from random blocks independently sampled on each machine. Next, using h^k , SEGA builds an unbiased gradient estimator g_i^k of $\nabla f_i(x^k)$ as follows:

$$g_i^k = h_i^k + \frac{1}{\tau} (\nabla f_i(x^k) - h_i^k)_{U_i^k}. \quad (4.13)$$

Then, we average the vectors g_i^k and take a proximal step.

Unlike coordinate descent, SEGA (or ISEGA) is not limited to separable proximal operators since, as follows from our analysis, $h_i^k \rightarrow \nabla f_i(x^*)$. Therefore, ISEGA can be seen as a variance reduced version of IBCD for problems with non-separable regularizers.

In order to be consistent with the rest of the chapter, we only develop a simple variant of ISEGA (Algorithm 13) in which we consider block coordinate sketches with uniform probabilities. While it is possible to develop the theory in full generality (done in Chapter 5) we avoid this for the sake of simplicity.

We next present the convergence rate of ISEGA (Algorithm 13).

Algorithm 13 ISEGA

```

1: Input:  $x^0 \in \mathbb{R}^d$ , initial gradient estimates  $h_1^0, \dots, h_n^0 \in \mathbb{R}^d$ , partition of  $\mathbb{R}^d$  into  $m$ 
   blocks  $u_1, \dots, u_m$ , ratio of blocks to be sampled  $\tau$ , stepsize  $\alpha$ , # parallel units  $n$ 
2: for  $k = 0, 1, 2, \dots$  do
3:   for  $i = 1, \dots, n$  in parallel do
4:     Sample independently and uniformly a subset of  $\tau m$  blocks  $U_i^k$ 
5:      $g_i^k = h_i^k + \frac{1}{\tau}(\nabla f_i(x^k) - h_i^k)_{S_i^k}$ 
6:      $h_i^{k+1} = h_i^k + \tau(g_i^k - h_i^k)$ 
7:   end for
8:    $x^{k+1} = \text{prox}_{\alpha\psi} \left( x^k - \alpha \frac{1}{n} \sum_{i=1}^n g_i^k \right)$ 
9: end for

```

Theorem 4.8.1. *Suppose Assumption 4.4.1 holds and choose stepsize*

$$\alpha = \min \left\{ \frac{1}{4L \left(1 + \frac{1}{n\tau}\right)}, \frac{1}{\frac{\mu}{\tau} + \frac{4L}{n\tau}} \right\}.$$

Then Algorithm 13 satisfies

$$\mathbb{E}[\|x^k - x^*\|^2] \leq (1 - \alpha\mu)^k \Phi^0,$$

where the Lyapunov function is given by $\Phi^0 \stackrel{\text{def}}{=} \|x^0 - x^*\|^2 + \frac{\alpha}{2L\tau n} \sum_{i=1}^n \|h_i^0 - \nabla f(x^*)\|^2$.

Note that if the condition number of the problem is not too small so that $n = \mathcal{O}(L/\mu)$ (which is usually the case in practice), ISEGA scales linearly in the parallel setting. In particular, when doubling the number of workers, each worker can afford to evaluate only half of the block partial derivatives while keeping the same convergence speed. Moreover, setting $\tau = \frac{1}{n}$, the rate corresponds, up to a constant factor, to the rate of gradient descent. Corollary 4.8.2 states the result.

Corollary 4.8.2. *Consider the setting from Theorem 4.8.1. Suppose that $\frac{L}{\mu} \geq n$ and choose $\tau = \frac{1}{n}$. The complexity of Algorithm 13 is $\mathcal{O}\left(\frac{L}{\mu} \log \frac{1}{\epsilon}\right)$.*

Remark 4. Parallel implementation Algorithm 13 would be to always send $(\nabla f_i(x^k))_{U_i^k}$ to the server; which keeps updating vector h^k and takes the prox step.

4.9 Experiments

In this section, we numerically verify our theoretical claims. Recall that there are various settings where it is possible to make practical experiments (see Section 4.3), however, we do not restrain ourselves to any of them in order to deliver as clear a message as possible.

We present exhaustive numerical experiments to verify the theoretical claims of the chapter. The experiments are performed in a simulated environment instead of the hon-

estly distributed setup, as we only aim to verify the iteration complexity of proposed methods.

First, in Section 4.9.1 provides the simplest setting in order to gain the best possible insight – Algorithm 8 is tested on the artificial quadratic minimization problem. We compare Algorithm 8 against both gradient descent (GD) and standard CD (in our setting: when each machine samples the same subset of coordinates). We also study the effect of changing τ on the convergence speed.

In the remaining parts, we consider a logistic regression problem on LibSVM data [23]. Recall that logistic regression problem is given as

$$f(x) \stackrel{\text{def}}{=} \frac{1}{N} \sum_{j=1}^N \left(\log(1 + \exp(\mathbf{A}_{j,:} x \cdot b_j)) + \frac{\lambda}{2} \|x\|^2 \right), \quad (4.14)$$

where \mathbf{A} is data matrix and b is vector of data labels: $b_j \in \{-1, 1\}$ ⁷. In the distributed scenario (everything except of Algorithm 9), we imitate that the data is evenly distributed to n workers (i.e. each worker owns a subset of rows of \mathbf{A} and corresponding labels, all subsets have almost the same size).

As our experiments are not aimed to be practical at this point (we aim to properly prove the conceptual idea), we consider multiple of rather smaller datasets: a1a ($d = 123, n = 1605$), mushrooms ($d = 112, n = 8124$), phishing ($d = 68, n = 11055$), w1a ($d = 300, n = 2477$). The experiments are essentially of 2 types: one shows that setting $n\tau = 1$ does not significantly violate the convergence of the original method. In the second type of experiments we study the behavior for varying τ , and show that beyond certain threshold, increasing τ does not significantly improve the convergence. The threshold is smaller as n increases, as predicted by theory.

4.9.1 Simple, well understood experiment

In this section we study the simplest possible setting – we test the behavior of Algorithm 8 on a quadratic minimization problem with artificial data. The considered quadratic objective is set as

$$f_i(x) \stackrel{\text{def}}{=} \frac{1}{2} x^\top \mathbf{M}_i x, \quad \mathbf{M}_i \stackrel{\text{def}}{=} vv^\top + \frac{(\mathbf{I} - vv^\top) \mathbf{A}_i \mathbf{A}_i^\top (\mathbf{I} - vv^\top)}{\lambda_{\max}(\mathbf{A}_i \mathbf{A}_i^\top)}, \quad v = \frac{v'}{\|v'\|}, \quad (4.15)$$

where entries of $v' \in \mathbb{R}^d$ and $\mathbf{A}_i \in \mathbb{R}^{d \times o}$ are sampled independently from standard normal distribution.

In the first experiment (Figure 4.1), we compare Algorithm 8 with $n\tau = 1$ against gradient descent (GD) and two versions of coordinate descent - a default version with stepsize $\frac{1}{L}$, and a coordinate descent with importance sampling (sample proportionally to coordinate-wise smoothness constants) and optimal step sizes (inverse of coordinate-wise smoothness constants). In all experiments, gradient descent enjoys twice better iteration complexity than Algorithm 8 which is caused by twice larger stepsize. However, in each

⁷ The datapoints (rows of \mathbf{A}) have been normalized so that each is of norm 1. Therefore, each f_i is $\frac{1}{4}$ smooth in all cases. We set regularization parameter as $\lambda = 0.00025$ in all cases.

case, Algorithm 8 requires fewer iterations to CD with importance sampling, which is itself significantly faster to plain CD.

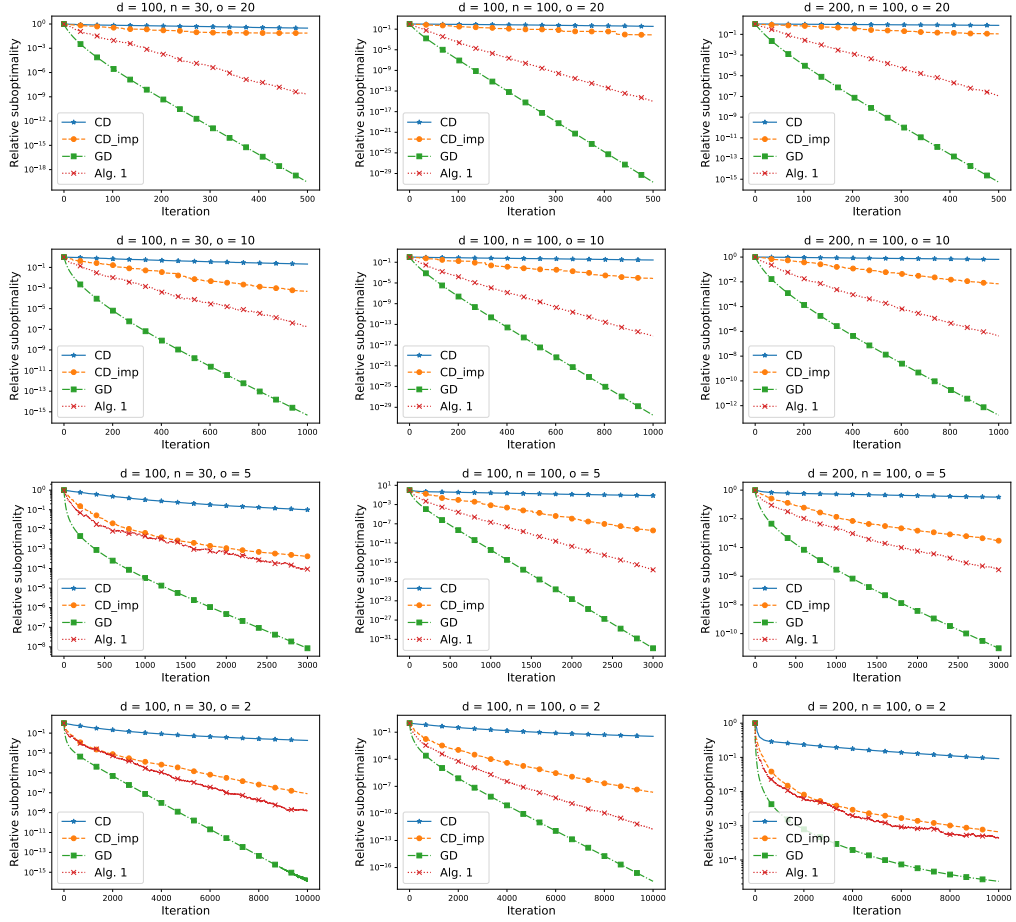


Figure 4.1: Comparison of gradient descent, (standard) coordinate descent, (standard) coordinate descent with importance sampling and Algorithm 8 on artificial quadratic problem (4.15).

Next, we study the effect of changing τ on the iteration complexity of Algorithm 8. Figure 4.2 provides the result. The behavior predicted from theory is observed – increasing τ over n^{-1} does not significantly improve the convergence speed, while decreasing it below n^{-1} slows the algorithm notably.

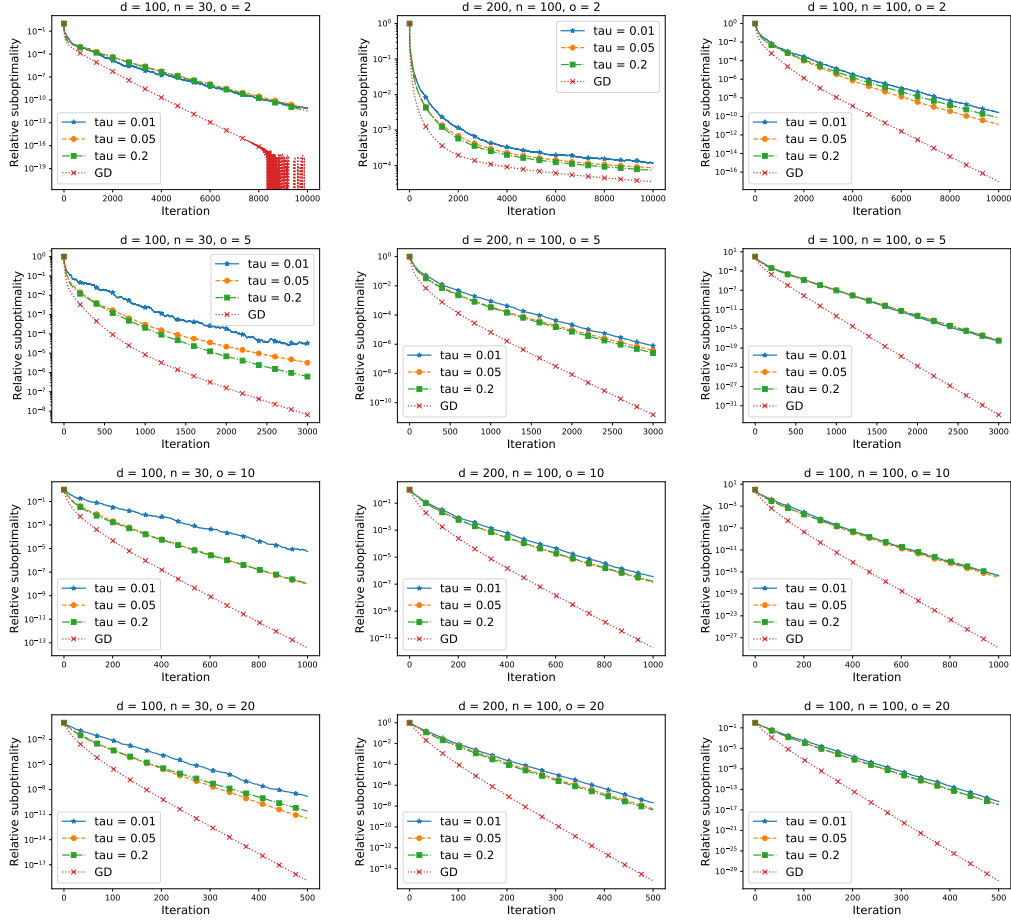


Figure 4.2: Behavior of Algorithm 8 for different τ on a simple artificial quadratic problem (4.15).

4.9.2 ISGD

In this section we numerically test Algorithm 11 for logistic regression problem. As mentioned, f_i consists of set of (uniformly distributed) rows of \mathbf{A} from (4.14). We consider the most natural unbiased stochastic oracle for the ∇f_i : the gradient computed on a subset of the data points from f_i .

In all experiments of this section, we consider constant step sizes in order to keep the setting as simple as possible and gain as much insight from the experiments as possible. Therefore, one can not expect convergence to the exact optimum.

In the first experiment, we compare standard SGD (stochastic gradient is computed on single, randomly chosen datapoint every iteration) against Algorithm 11 varying n and choosing $\tau = \frac{1}{n}$ for each n . The results are presented by Figure 4.3. We see that, as our theory suggests, SGD and Algorithm 11 have always very similar performance.

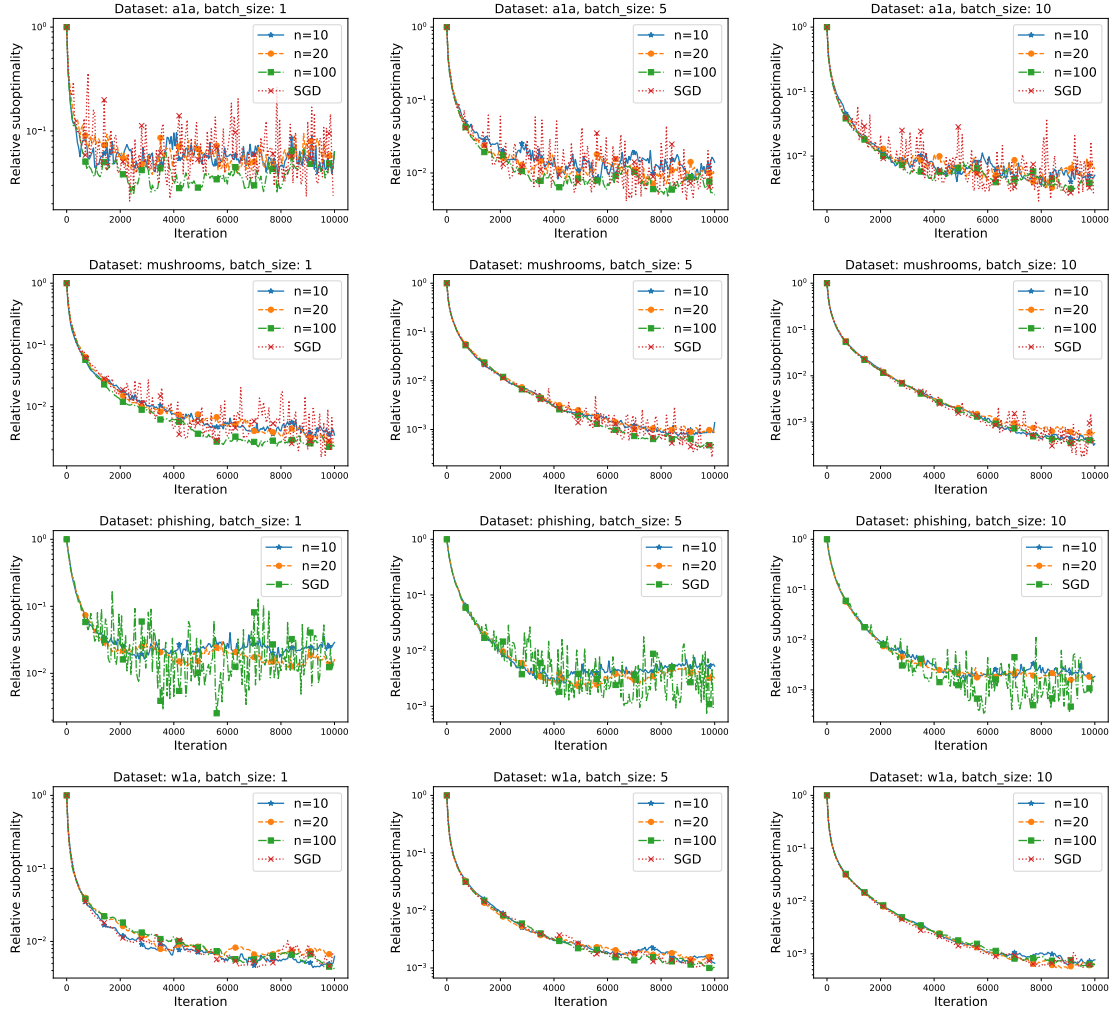


Figure 4.3: Comparison of SGD (gradient evaluated on a single datapoint) and Algorithm 11 with $n\tau = 1$. Constant $\alpha = \frac{1}{5L}$ was used for each algorithm. Label “batch_size” indicates how big minibatch was chosen for stochastic gradient of each worker’s objective.

Next, we study the dependence of the convergence speed on τ for various values of n . Figure 4.4 presents the results. In each case, τ influences the convergence rate (or the region where the iterates oscillate) significantly, however, the effect is much weaker for larger n . This is in correspondence with Corollary 4.6.4.

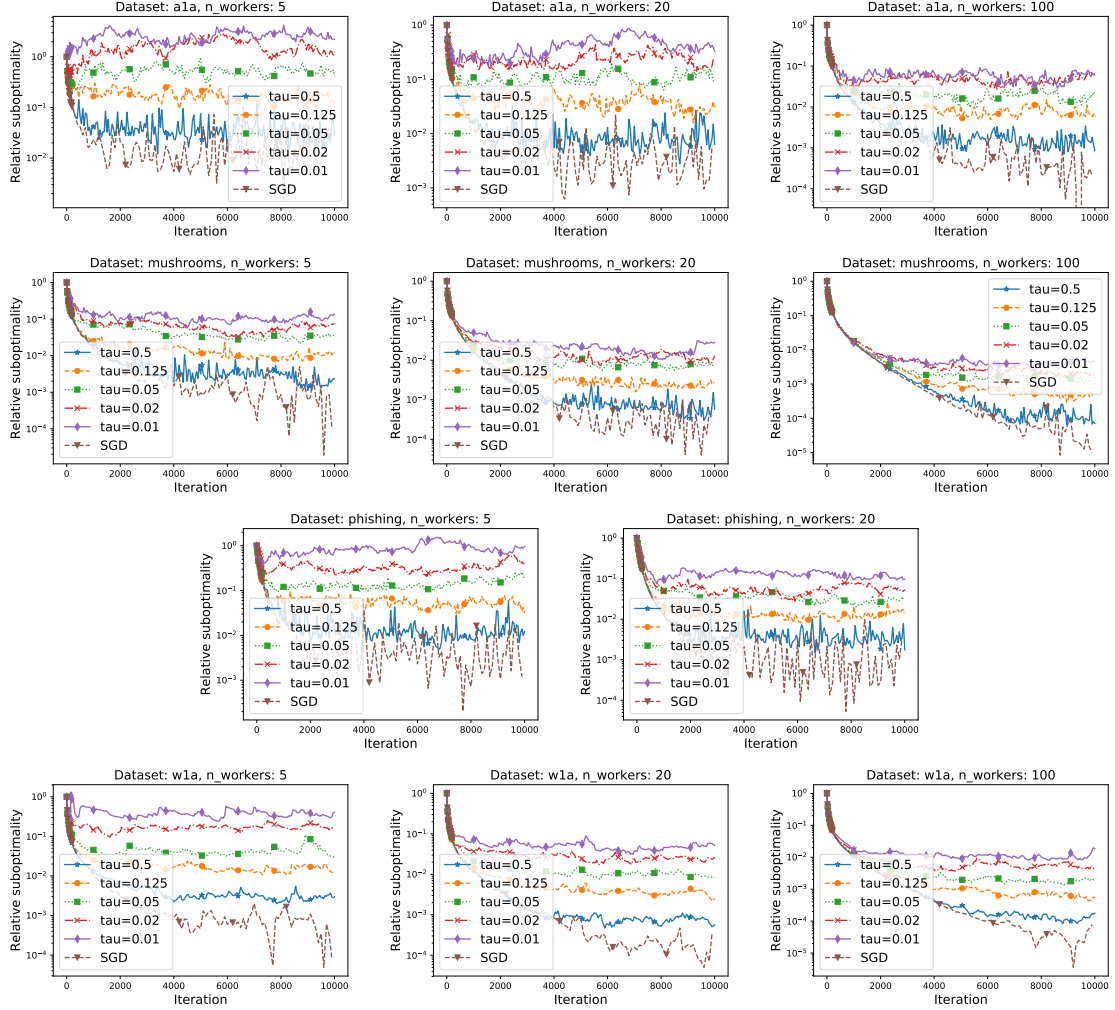


Figure 4.4: Behavior of Algorithm 11 while varying τ . Label “SGD” corresponds to the choice $n = 1, \tau = 1$. Stepsize $\alpha = \frac{1}{3L}$ was used in every case.

4.9.3 IASGD

In this section we numerically test Algorithm 12 for logistic regression problem. As in the last section, f_i consists of set of (uniformly distributed) rows of \mathbf{A} from (4.14). The stochastic gradient is taken as a gradient on a subset data points from each f_i . Note that Algorithm 12 depends on a priori unknown strong growth parameter $\hat{\rho}$ of unbiased stochastic gradient q^8 . Therefore, we first find empirically optimal $\hat{\rho}$ for each algorithm run by grid search and report only the best performance for each algorithm.

The first experiment (Figure 4.5) verifies the linearity claim – we vary (n, τ) such that $n\tau = 1$. As predicted by theory, the behavior of presented algorithms is almost indistinguishable.

⁸Formulas to obtain parameters of Algorithm 12 are given in [208].

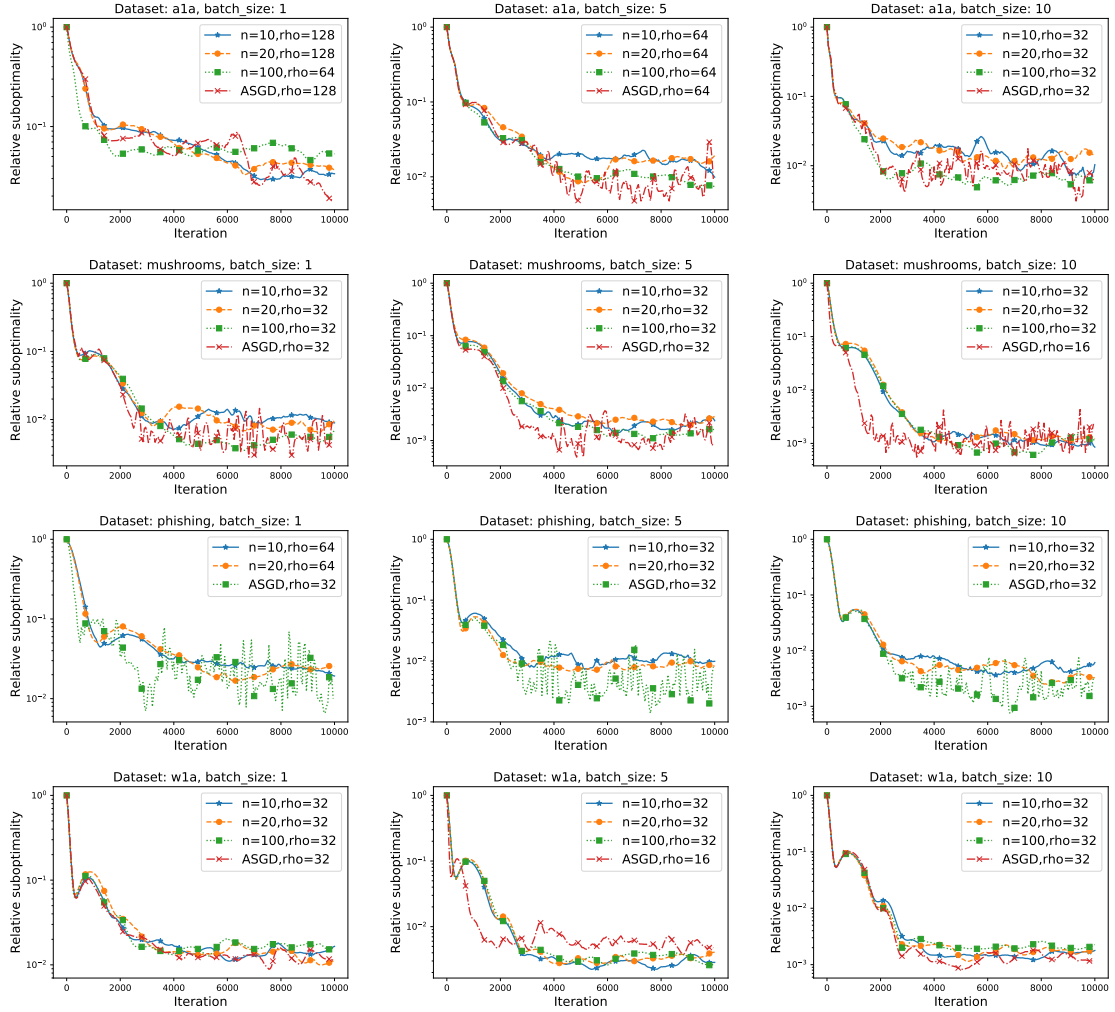


Figure 4.5: Comparison of Algorithm 12 for various (n, τ) such that $n\tau = 1$. Label “ASGD” corresponds to the choice $n = 1, \tau = 1$. Label “batch_size” indicates how big minibatch was chosen for stochastic gradient of each worker’s objective. Parameter ρ was chosen by grid search.

Now, we once again check how different values of τ affect the convergence speed for several values of n . Figure 4.6 presents the results. In every case, τ slightly influences the convergence rate (or the region where the iterates oscillate), although the effect is weaker for larger n . Note that theory predicts diminishing effect of τ only above $\frac{\bar{\rho}\hat{\rho}}{n}$, in contrast to other sections, where the limit is $\frac{1}{n}$.

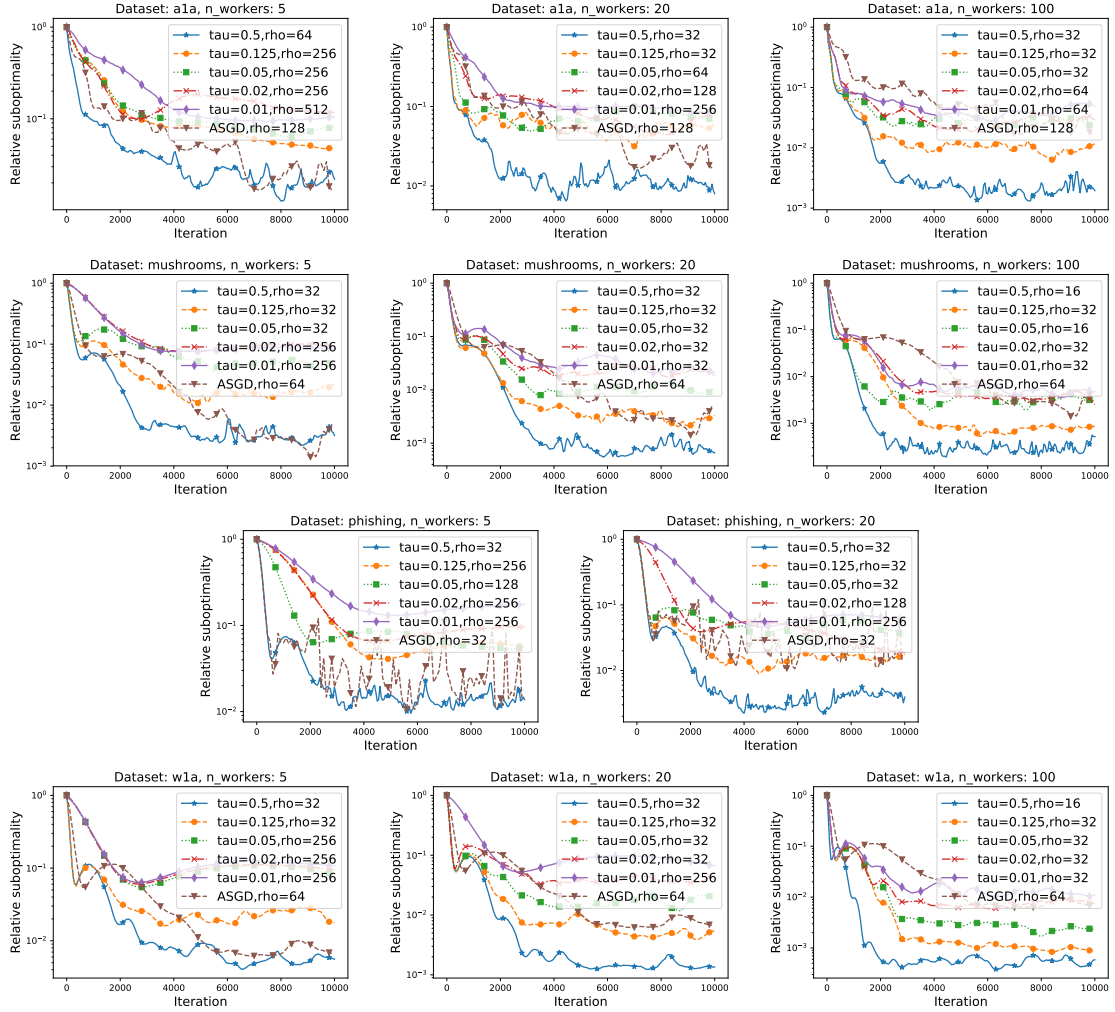


Figure 4.6: Behavior of Algorithm 12 while varying τ . Label “ASGD” corresponds to the choice $n = 1, \tau = 1$. Parameter ρ was chosen by grid search.

4.9.4 ISAGA

In the next experiment, we compare SAGA against ISAGA in a shared data setup (Algorithm 9) for various values of n with $\tau = \frac{1}{n}$ in order to demonstrate linear scaling. We consider logistic regression problem on LibSVM data [23]. The results (Figure 4.7) corroborate our theory: indeed, setting $n\tau = 1$ does not lead to a decrease in the convergence rate when compared to the original SAGA.

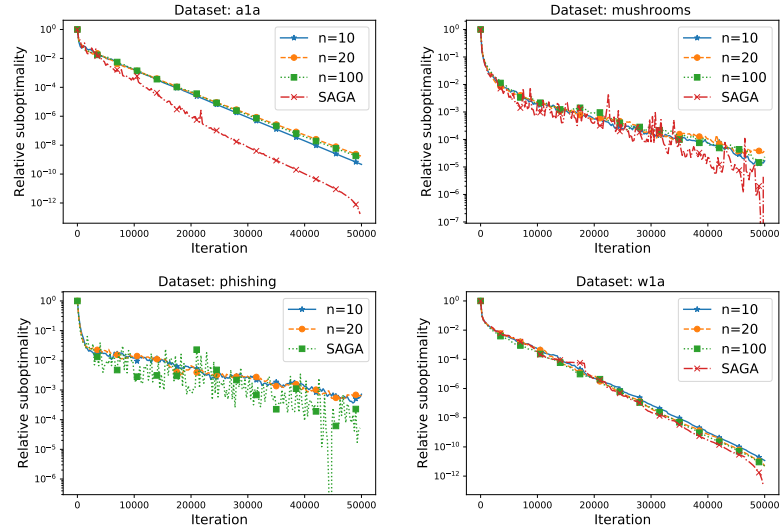


Figure 4.7: Comparison of SAGA and Algorithm 9 for various values n and $\tau = n^{-1}$. Stepsize $\alpha = \frac{1}{L(3n^{-1} + \tau)}$ is chosen in each case.

The second experiment of this section shows the convergence behavior for varying τ of Algorithm 9. The results (Figure 4.8) show that, for small n , the ratio of coordinates τ affects the speed heavily. However, as n increases, the effect of τ is diminishing.

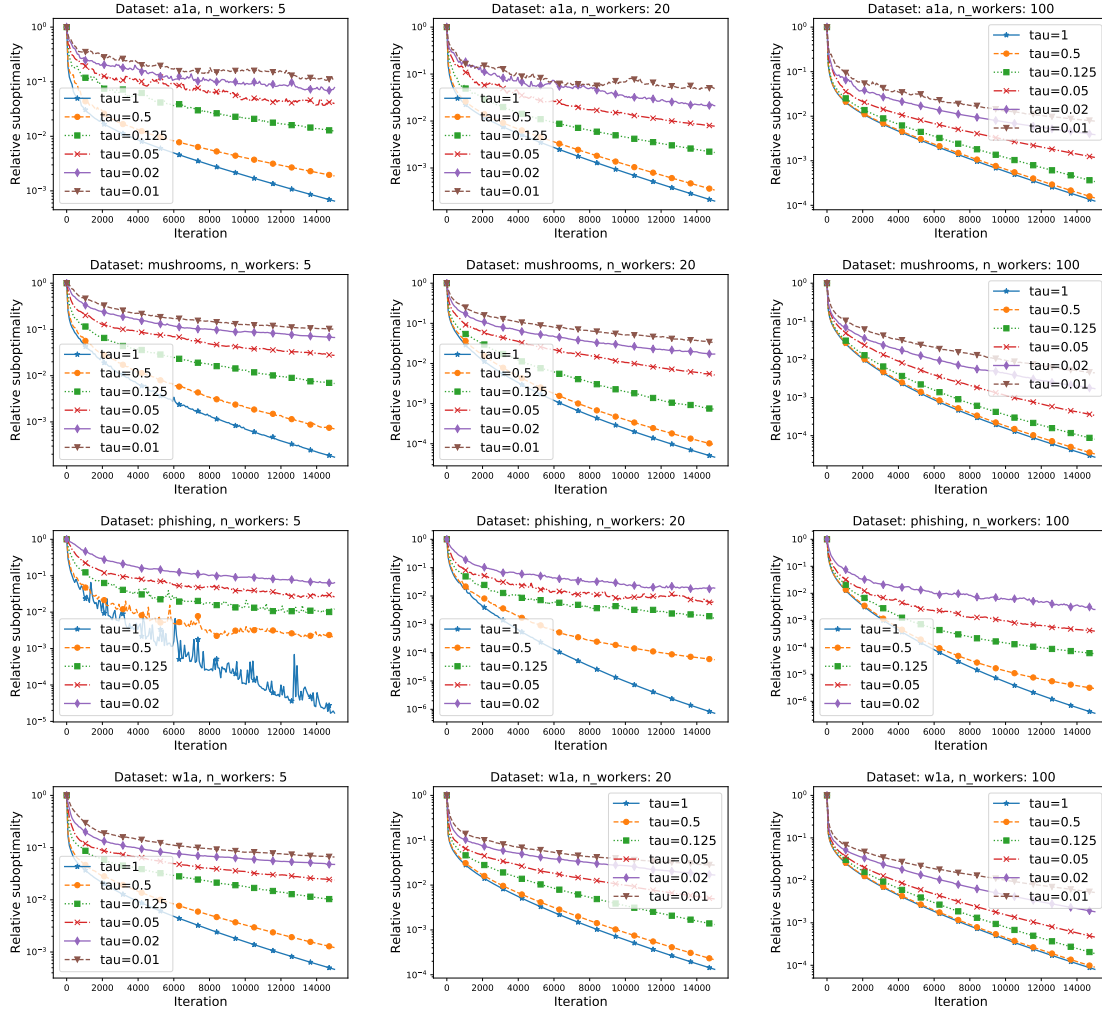


Figure 4.8: Comparison of Algorithm 9 for different values of τ . Stepsize $\alpha = \frac{1}{L(3n^{-1} + \tau)}$ is chosen in each case. For this experiment, we choose smaller regularization; $\lambda = 0.000025$.

4.9.5 ISEGA

Lastly, we numerically test Algorithm 13, and its linear convergence. For simplicity, we consider $\psi \equiv 0$ in (4.11).

In the first experiment (Figure 4.9), we compare Algorithm 13 for various (n, τ) such that $n\tau = 1$. For illustration, we also plot convergence of gradient descent with the analogous stepsize. As theory predicts, the method has almost same convergence speed.⁹

⁹We have chosen stepsize $\alpha = \frac{1}{2L}$ for GD, as this is the baseline to Algorithm 13 with zero variance. One can in fact set $\alpha = \frac{1}{L}$ for GD and get 2 times faster convergence. However, this is still only a constant factor.

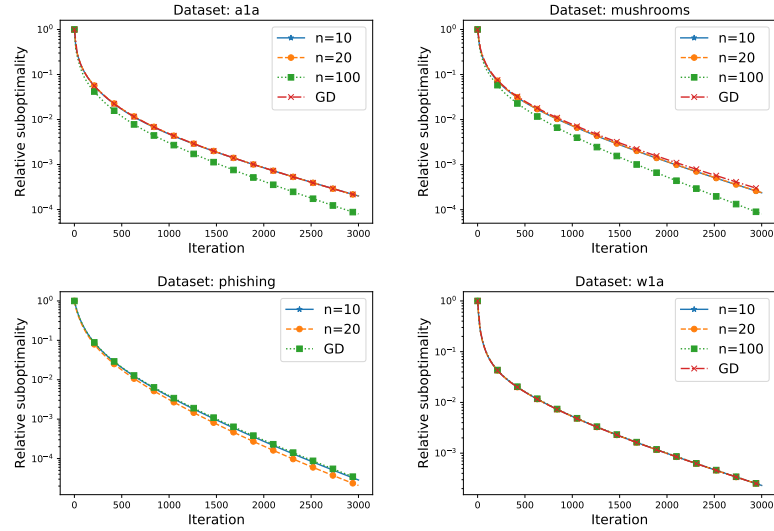


Figure 4.9: Comparison of Algorithm 13 for various (n, τ) such that $n\tau = 1$ and GD. Stepsize $\frac{1}{L(1+\frac{1}{n\tau})}$ was chosen for Algorithm 13 and $\frac{1}{2L}$ for GD.

The second experiment of this section shows the convergence behavior for varying τ of Algorithm 13. Again, the results (Figure 4.10) indicate that τ has a heavy impact on the convergence speed for small n . However, as n increases, the effect of τ is diminishing. In particular, for increasing τ beyond n^{-1} does not yield a significant speedup.

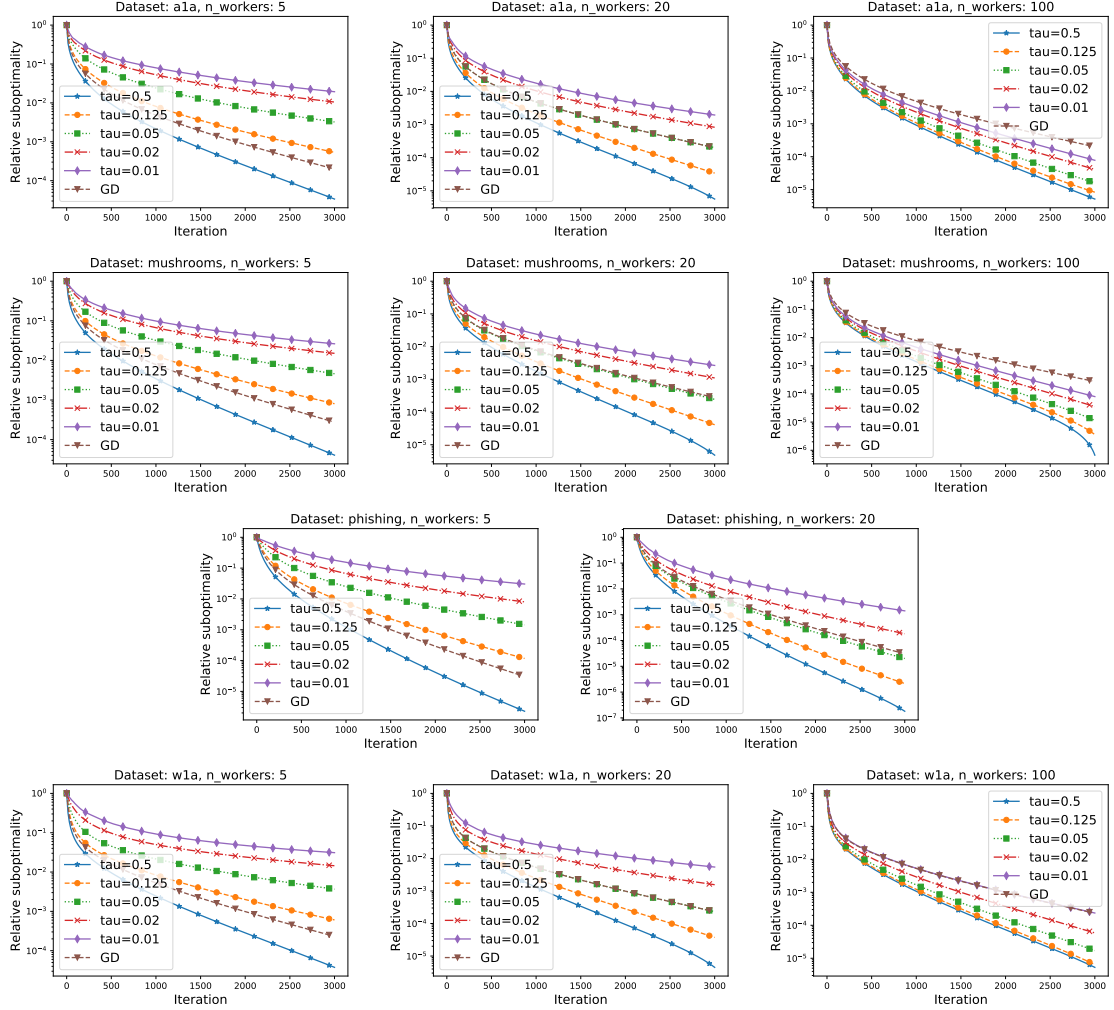


Figure 4.10: Comparison of Algorithm 13 for different values of τ . Step size $\alpha = \frac{1}{L(1+\frac{1}{n\tau})}$ is chosen in each case.

4.10 Conclusion

In this chapter, we have proposed a strategy for reducing the worker→server communication by $\mathcal{O}(\frac{n-1}{n}) \times 100\%$, where n is the number of workers. The algorithms we introduced are merely act as demonstrations of what can be achieved using our main insight, and many further extensions are possible. Specifically, in the next chapter we propose GJS: a new algorithm that obtains several further extensions of the methods developed in this chapter in special cases:

- Distributed ISAGA requires $\nabla f_i(x) = 0$. GJS allows to develop SEGA approach on top of it in order to drop this requirement.
- Standard coordinate descent is able to exploit a complex smoothness structure of objective in order to sample coordinates non-uniformly [167, 31]. As a special case

of GJS, we obtain importance sampling variants of multiple algorithms proposed here.

Chapter 5

One Method to Rule Them All: Variance Reduction for Data, Parameters and Many New Methods

In this chapter we finally consider problem (1.1) in its fully general form; i.e., we aim to solve the problem

$$\min_{x \in \mathbb{R}^d} \frac{1}{n} \sum_{j=1}^n f_j(x) + \psi(x). \quad (5.1)$$

We assume that the functions $f_j : \mathbb{R}^d \rightarrow \mathbb{R}$ are smooth and convex, and $\psi : \mathbb{R}^d \rightarrow \mathbb{R} \cup \{+\infty\}$ is a proper, closed and convex regularizer, admitting a cheap proximal operator. As usual, we write $f \stackrel{\text{def}}{=} \frac{1}{n} \sum_j f_j$.

Proximal gradient descent. A baseline method for solving problem (5.1) is (*proximal gradient descent*), described in detail in Section 1.1.4 of the introduction. For the sake of simplicity, let us call it PGD throughout this section. As already stressed, PGD performs well when both n and d are not too large. However, in the big data (large n) and/or big parameter (large d) case, the formation of the gradient becomes overly expensive, rendering PGD inefficient in both theory and practice. A typical remedy is to replace the gradient by a cheap-to-compute random approximation. Typically, one replaces $\nabla f(x^k)$ with a random vector g^k whose mean is the gradient: $\mathbb{E}[g^k] = \nabla f(x^k)$, i.e., with a stochastic gradient. This results in the (*proximal stochastic gradient descent*) (SGD) method:

$$x^{k+1} = \text{prox}_{\alpha\psi}(x^k - \alpha g^k). \quad (5.2)$$

Below we comment on the typical approaches to constructing g^k in the big n and big d regimes (this was, to some extent, mentioned in the introduction already).

Proximal SGD. In the big n regime, the simplest choice is to set

$$g^k = \nabla f_j(x^k) \quad (5.3)$$

for an index $j \in [n] \stackrel{\text{def}}{=} \{1, 2, \dots, n\}$ chosen uniformly at random. By construction, it is n times cheaper to compute this estimator than the gradient, which is a key driving force behind the efficiency of this variant of SGD. However, there is an infinite array of other possibilities of constructing an unbiased estimator [147, 60]. Depending on how g^k is formed, (5.2) specializes to one of the many existing variants of proximal SGD, each with different convergence properties and proofs.

Proximal RCD. In the big d regime (this is interesting even if $n = 1$), the simplest choice is to set

$$g^k = d \langle \nabla f(x^k), \mathbf{e}_i \rangle \mathbf{e}_i, \quad (5.4)$$

where $\langle x, y \rangle = \sum_i x_i y_i$ is the standard Euclidean inner product, \mathbf{e}_i is the i th standard unit basis vector in \mathbb{R}^d , and i is chosen uniformly at random from $[d] \stackrel{\text{def}}{=} \{1, 2, \dots, d\}$.¹ With this estimator, (5.2) specializes to (proximal) randomized coordinate decent (RCD). There are situations where it is d times cheaper to compute the partial derivative $\nabla_i f(x^k) \stackrel{\text{def}}{=} \langle \nabla f(x^k), \mathbf{e}_i \rangle$ than the gradient, which is a key driving force behind the efficiency of RCD [152]. However, there is an infinite array of other possibilities for constructing an unbiased estimator of the gradient in a similar way [175, 152, 167].

Issues. For the sake of argument in the rest of this section, assume that f is a μ -strongly convex function, and let x^* be the (necessarily) unique solution of (5.1). It is well known that in this case, method (5.2) with estimator g^k defined as in (5.3) does *not* in general converge to x^* . Instead, SGD converges linearly to a neighborhood of x^* of size proportional to the stepsize α , noise $\sigma^2 \stackrel{\text{def}}{=} \frac{1}{n} \sum_j \|\nabla f_j(x^*)\|^2$, and inversely proportional to μ [143, 146]. In the generic regime with $\sigma^2 > 0$, the neighbourhood is nonzero, causing issues with convergence. This situation does not change even when tricks such as *mini-batching* or *importance sampling* (or a combination of both) are applied [146, 147, 60]. While these tricks affect both the (linear convergence) rate and the size of the neighbourhood, they are incapable² of ensuring convergence to the solution.

However, a remedy does exist: the situation with non-convergence can be resolved by using one of the many *variance-reduction* strategies for constructing g^k developed over the last several years [182, 37, 88, 132, 191].

Further, while it is well known that method (5.2) with estimator g^k defined as in (5.4) (i.e., randomized coordinate descent) converges to x^* for $\psi \equiv 0$ [152, 173, 175], it is also known that it does *not* generally converge to x^* unless the regularizer ψ is separable (e.g., $\psi(x) = \|x\|_1$ or $\psi(x) = c_1 \|x\|_1 + c_2 \|x\|_2^2$). In [77], an alternative estimator (known as SEGA) was constructed from the same (random) partial derivative information $\nabla f_i(x^k)$, one that does not suffer from this incompatibility with general regularizers ψ . This work resolved a long standing open problem in the theory of RCD methods.

Notation. Let \mathbf{e} (resp. \mathbf{e}) be the vector of all ones in \mathbb{R}^n (resp. \mathbb{R}^d), and \mathbf{e}_j (resp. \mathbf{e}_i) be the j th (resp. i th) unit basis vector in \mathbb{R}^n (resp. \mathbb{R}^d). By $\|\cdot\|$ we denote the standard Euclidean norm in \mathbb{R}^d and \mathbb{R}^n . Matrices are denoted by upper-case bold letters. Given $\mathbf{X}, \mathbf{Y} \in \mathbb{R}^{d \times n}$, let $\langle \mathbf{X}, \mathbf{Y} \rangle \stackrel{\text{def}}{=} \text{Tr}(\mathbf{X}^\top \mathbf{Y})$ and $\|\mathbf{X}\| \stackrel{\text{def}}{=} \langle \mathbf{X}, \mathbf{X} \rangle^{1/2}$ be the Frobenius norm. By $\mathbf{X}_{:j}$ (resp. $\mathbf{X}_{i:}$) we denote the j th column (resp. i th row) of matrix \mathbf{X} . By \mathbf{I}_n (resp. \mathbf{I}_d) we denote the $n \times n$ (resp. $d \times d$) identity matrices. Upper-case calligraphic letters, such as $\mathcal{S}, \mathcal{U}, \mathcal{I}, \mathcal{M}, \mathcal{R}$, are used to denote (deterministic or random) linear operators mapping $\mathbb{R}^{d \times n}$ to $\mathbb{R}^{d \times n}$. Most used notation is summarized in Table A.4 in Appendix A.

¹The algorithm proposed in this chapter subsamples both the finite sum and the domain. For the notational simplicity, we distinguish the two different spaces using color where necessary.

²Unless, of course, in the special case when one uses the full batch approximation $g^k = \nabla f(x^k)$.

5.1 Contributions

Having experienced a “Cambrian explosion” in the last 10 years, the world of efficient SGD methods is remarkably complex. There is a large and growing set of rules for constructing the gradient estimators g^k , with differing levels of sophistication and varying theoretical and practical properties. It includes the classical estimator (5.3), as well as an infinite array of mini-batch [118] and importance sampling [146, 223] variants, and a growing list of variance-reduced variants [37]. Furthermore, there are estimators of the coordinate descent variety, including the simplest one based on (5.4) [152], more elaborate variants utilizing the arbitrary sampling paradigm [166], and variance reduced methods capable of handling general non-separable regularizers [77].

- **New general method and a single convergence theorem.** In this chapter we propose a *general method*—which we call GJS—which reduces to many of the aforementioned classical and several recently developed SGD type methods in special cases. We provide a *single convergence theorem*, establishing a linear convergence rate for GJS, assuming f to be smooth and quasi strongly convex. In particular, we obtain the following methods in special cases, or their generalizations, always recovering the best-known convergence guarantees or improving upon them: SAGA [37, 165, 52], JacSketch [65], LSVRG [83, 106], SEGA [77], and ISEGA [137] (see Table 5.1, in which we list 17 special cases). This is the first time such a direct connection is made between many of these methods, which previously required different intuitions and dedicated analyses. Our general method, and hence also all special cases we consider, can work with a regularizer. This provides novel (although not hard) results for some methods, such as LSVRG.
- **Unification of SGD and RCD.** As a by-product of the generality of GJS, we obtain the *unification of variance-reduced SGD and variance reduced RCD methods*. To the best of our knowledge, there is no algorithm besides GJS, one whose complexity is captured by a single theorem, which specializes to SGD and RCD type methods at the same time and recovers best known rates in both cases.³
- **Generalizations to arbitrary sampling.** Many specialized methods we develop are cast in a very general *arbitrary sampling* paradigm [175, 169, 166], which allows for the estimator g^k to be formed through information contained in a random subset $R^k \subseteq [n]$ (by computing $\nabla f_j(x^k)$ for $j \in R^k$) or a random subset $L^k \subseteq [d]$ (by computing $\nabla_i f(x^k)$ for $i \in L^k$), where these subsets are allowed to follow an arbitrary distribution. In particular, we extend SEGA [77], LSVRG [83, 106] or ISEGA [137] to this setup. Likewise, GJS specializes to an arbitrary sampling extension of the SGD-type method SAGA [37, 165], obtaining state-of-the-art rates. As a special case of the arbitrary sampling paradigm, we obtain *importance sampling* versions of all mentioned methods.

³A single theorem (not a single algorithm) to obtain rates for both variance-reduced SGD and variance reduced RCD methods was done in the concurrent work [55]. However, [55] focuses in orthogonal direction instead – it is a tool to analyze stochastic gradient algorithms which includes non-variance reduced methods as well.

- **New methods.** GJS can be specialized to many new specific methods. To illustrate this, we construct 10 specific *new* methods in special cases, some with intriguing structure and properties (see Section 5.5; Table 5.1; and Table E.1 for a summary of the rates).
- **Relation to JacSketch.** Our method can be seen as a vast generalization of the recently proposed Jacobian sketching method JacSketch [65] in several directions, notably by enabling *arbitrary randomized linear* (i.e., sketching) operators, allowing different linear operators to learning Jacobian and constructing control variates, extending the analysis to the proximal case, and replacing strong convexity assumption by quasi strong convexity or strong growth (see Appendix E.13). In particular, from all methods we recover, only variants of SAGA can be obtained from JacSketch [65] (even in that case, rates obtained from [65] are suboptimal).
- **Limitations.** We focus on developing methods capable of enjoying a linear convergence rate with a fixed stepsize α and do not consider the non-convex setting. Although there exist several *accelerated* variance reduced algorithms [113, 4, 227, 226, 106, 110], we do not consider such methods here.

5.2 Sketching

A key object in this chapter is the Jacobian matrix $\mathbf{G}(x) = [\nabla f_1(x), \dots, \nabla f_n(x)] \in \mathbb{R}^{d \times n}$. Note that

$$\nabla f(x) = \frac{1}{n} \mathbf{G}(x) \mathbf{e}. \quad (5.5)$$

Extending the insights from [65], one of the key observations of this work is that *random linear transformations* (sketches) of \mathbf{G} can be used to *construct* unbiased estimators of the gradient of f . For instance, $\mathbf{G}(x^k) \mathbf{e}_j$ leads to the simple SGD estimator (5.3), and $\frac{d}{n} \mathbf{e}_i \mathbf{e}_i^\top \mathbf{G}(x^k) \mathbf{e}$ gives the simple RCD estimator (5.4). We will consider more elaborate examples later on. It will be useful to embed these estimators into $\mathbb{R}^{d \times n}$. For instance, instead of $\mathbf{G}(x^k) \mathbf{e}_j$ we consider the matrix $\mathbf{G}(x^k) \mathbf{e}_j \mathbf{e}_j^\top$. Note that all columns of this matrix are zero, except for the j th column, which is equal to $\mathbf{G}(x^k) \mathbf{e}_j$. Similarly, instead of $\frac{d}{n} \mathbf{e}_i \mathbf{e}_i^\top \mathbf{G}(x^k) \mathbf{e}$ we will consider the matrix $\frac{d}{n} \mathbf{e}_i \mathbf{e}_i^\top \mathbf{G}(x^k)$. All rows of this matrix are zero, except for the i th row, which consists of the i th partial derivatives of functions $f_j(x^k)$ for $j \in [n]$, scaled by $\frac{d}{n}$.

Random projections. Generalizing from these examples, we consider a random linear operator (“sketch”) $\mathcal{A} : \mathbb{R}^{d \times n} \rightarrow \mathbb{R}^{d \times n}$. By \mathcal{A}^* we denote the adjoint of \mathcal{A} , i.e., linear operator satisfying $\langle \mathcal{A}\mathbf{X}, \mathbf{Y} \rangle = \langle \mathbf{X}, \mathcal{A}^*\mathbf{Y} \rangle$ for all $\mathbf{X}, \mathbf{Y} \in \mathbb{R}^{d \times n}$. Given \mathcal{A} , we let $\mathcal{P}_{\mathcal{A}}$ be the (random) projection operator onto $\text{Range}(\mathcal{A}^*)$. That is,

$$\mathcal{P}_{\mathcal{A}}(\mathbf{X}) = \arg \min_{\mathbf{Y} \in \text{Range}(\mathcal{A}^*)} \|\mathbf{X} - \mathbf{Y}\| = \mathcal{A}^* (\mathcal{A} \mathcal{A}^*)^\dagger \mathcal{A} \mathbf{X},$$

where † is the Moore-Penrose pseudoinverse. The identity operator is denoted by \mathcal{I} . We say that \mathcal{A} is *identity in expectation*, or *unbiased* when $\mathbb{E}[\mathcal{A}] = \mathcal{I}$; i.e., when if

$\mathbb{E}[\mathcal{A}\mathbf{X}] = \mathbf{X}$ for all $\mathbf{X} \in \mathbb{R}^{d \times n}$.

Definition 5.2.1. We will often consider the following⁴ sketching operators \mathcal{A} :

- (i) **Right sketch.** Let $\mathbf{R} \in \mathbb{R}^{n \times n}$ be a random matrix. Define \mathcal{A} by $\mathcal{A}\mathbf{X} = \mathbf{X}\mathbf{R}$ (“R-sketch”). Notice that $\mathcal{A}^*\mathbf{X} = \mathbf{X}\mathbf{R}^\top$. In particular, if R is random subset of $[n]$, we can define $\mathbf{R} = \sum_{j \in R} \mathbf{e}_j \mathbf{e}_j^\top$. The resulting operator \mathcal{A} (“R-sampling”) satisfies: $\mathcal{A} = \mathcal{A}^* = \mathcal{A}^2 = \mathcal{P}_{\mathcal{A}}$. If we let $p_j \stackrel{\text{def}}{=} \mathbb{P}(j \in R)$, and instead define $\mathbf{R} = \sum_{j \in R} \frac{1}{p_j} \mathbf{e}_j \mathbf{e}_j^\top$, then $\mathbb{E}[\mathbf{R}] = \mathbf{I}_n$ and hence \mathcal{A} is unbiased.
- (ii) **Left sketch.** Let $\mathbf{L} \in \mathbb{R}^{d \times d}$ be a random matrix. Define \mathcal{A} by $\mathcal{A}\mathbf{X} = \mathbf{L}\mathbf{X}$ (“L-sketch”). Notice that $\mathcal{A}^*\mathbf{X} = \mathbf{L}^\top \mathbf{X}$. In particular, if L is random subset of $[d]$, we can define $\mathbf{L} = \sum_{i \in L} \mathbf{e}_i \mathbf{e}_i^\top$. The resulting operator \mathcal{A} (“L-sampling”) satisfies: $\mathcal{A} = \mathcal{A}^* = \mathcal{A}^2 = \mathcal{P}_{\mathcal{A}}$. If we let $p_i \stackrel{\text{def}}{=} \mathbb{P}(i \in L)$, and instead define $\mathbf{L} = \sum_{i \in L} \frac{1}{p_i} \mathbf{e}_i \mathbf{e}_i^\top$, then $\mathbb{E}[\mathbf{L}] = \mathbf{I}_d$ and hence \mathcal{A} is unbiased.
- (iii) **Scaling/Bernoulli.** Let ξ be a Bernoulli random variable, i.e., $\xi = 1$ with probability ρ and $\xi = 0$ with probability $1 - \rho$, where $\rho \in [0, 1]$. Define \mathcal{A} by $\mathcal{A}\mathbf{X} = \xi \mathbf{X}$ (“scaling”). Then $\mathcal{A} = \mathcal{A}^* = \mathcal{A}^2 = \mathcal{P}_{\mathcal{A}}$. If we instead define $\mathcal{A}\mathbf{X} = \frac{1}{\rho} \xi \mathbf{X}$, then \mathcal{A} is unbiased.
- (iv) **LR sketch.** All the above operators can be combined. In particular, we can define $\mathcal{A}\mathbf{X} = \xi \mathbf{L} \mathbf{X} \mathbf{R}$. All of the above arise as special cases of this: (i) arises for $\xi \equiv 1$ and $\mathbf{L} \equiv \mathbf{I}_d$, (ii) for $\xi \equiv 1$ and $\mathbf{R} \equiv \mathbf{I}_n$, and (iii) for $\mathbf{L} \equiv \mathbf{I}_d$ and $\mathbf{R} \equiv \mathbf{I}_n$.

5.3 The GJS algorithm

We are now ready to describe our method (formalized as Algorithm 14). Let \mathcal{S} be a

Algorithm 14 Generalized JacSketch (GJS)

- 1: **Parameters:** Stepsize $\alpha > 0$, random projector \mathcal{S} and unbiased sketch \mathcal{U}
 - 2: **Initialization:** Choose solution estimate $x^0 \in \mathbb{R}^d$ and Jacobian estimate $\mathbf{J}^0 \in \mathbb{R}^{d \times n}$
 - 3: **for** $k = 0, 1, 2, \dots$ **do**
 - 4: Sample realizations of \mathcal{S} and \mathcal{U} , and perform sketches $\mathcal{S}\mathbf{G}(x^k)$ and $\mathcal{U}\mathbf{G}(x^k)$
 - 5: $\mathbf{J}^{k+1} = \mathbf{J}^k - \mathcal{S}(\mathbf{J}^k - \mathbf{G}(x^k))$ update the Jacobian estimate via (5.8)
 - 6: $g^k = \frac{1}{n} \mathbf{J}^k \mathbf{e} + \frac{1}{n} \mathcal{U}(\mathbf{G}(x^k) - \mathbf{J}^k) \mathbf{e}$ construct the gradient estimator via (5.6)
 - 7: $x^{k+1} = \text{prox}_{\alpha\psi}(x^k - \alpha g^k)$ perform the proximal SGD step (5.2)
 - 8: **end for**
-

random linear operator (e.g., right sketch, left sketch, or scaling) such that $\mathcal{S} = \mathcal{P}_{\mathcal{S}}$ and

⁴The algorithm we develop is, however, not limited to such sketches.

let \mathcal{U} be an unbiased operator. We propose to construct the gradient estimator as

$$g^k = \frac{1}{n} \mathbf{J}^k e + \frac{1}{n} \mathcal{U}(\mathbf{G}(x^k) - \mathbf{J}^k) e, \quad (5.6)$$

where the matrices $\mathbf{J}^k \in \mathbb{R}^{d \times n}$ are constructed iteratively. Note that, taking expectation in \mathcal{U} , we get

$$\mathbb{E}[g^k] \stackrel{(5.6)}{=} \frac{1}{n} \mathbf{J}^k e + \frac{1}{n} (\mathbf{G}(x^k) - \mathbf{J}^k) e = \frac{1}{n} \mathbf{G}(x^k) e \stackrel{(5.5)}{=} \nabla f(x^k), \quad (5.7)$$

and hence g^k is indeed unbiased. We will construct \mathbf{J}^k so that $\mathbf{J}^k \rightarrow \mathbf{G}(x^*)$. By doing so, the variance of g^k decreases throughout the iterations, completely vanishing at x^* . The sequence $\{\mathbf{J}^k\}$ is updated as follows:

$$\mathbf{J}^{k+1} = \arg \min_{\mathbf{J}} \{ \|\mathbf{J} - \mathbf{J}^k\| : \mathcal{S}\mathbf{J} = \mathcal{S}\mathbf{G}(x^k) \} = \mathbf{J}^k - \mathcal{S}(\mathbf{J}^k - \mathbf{G}(x^k)). \quad (5.8)$$

That is, we sketch the Jacobian $\mathbf{G}(x^k)$, obtaining the sketch $\mathcal{S}\mathbf{G}(x^k)$, and seek to use this information to construct a new matrix \mathbf{J}^{k+1} which is consistent with this sketch, and as close to \mathbf{J}^k as possible. The intuition here is as follows: if we repeated the sketch-and-project process (5.8) for fixed x^k , the matrices \mathbf{J}^k would converge to $\mathbf{G}(x^k)$, at a linear rate [61, 64]. This process can be seen as SGD applied to a certain quadratic stochastic optimization problem [176, 65]. Instead, we take just one step of this iterative process, change x^k , and repeat. Note that the unbiased sketch \mathcal{U} in (5.6) also claims access to $\mathbf{G}(x^k)$. Specific variants of GJS are obtained by choosing specific operators \mathcal{S} and \mathcal{U} (see Section 5.5).

5.4 Theory

We now describe the main result of this chapter, which depends on a relaxed strong convexity assumption and a more precise smoothness assumption on f .

Assumption 5.4.1. *Problem (5.1) has a unique minimizer x^* , and f is μ -quasi strongly convex, i.e.,*

$$f(x^*) \geq f(y) + \langle \nabla f(y), x^* - y \rangle + \frac{\mu}{2} \|y - x^*\|^2, \quad \forall y \in \mathbb{R}^d, \quad (5.9)$$

Functions f_j are convex and \mathbf{M}_j -smooth for some $\mathbf{M}_j \succeq 0$, i.e.,

$$f_j(y) + \langle \nabla f_j(y), x - y \rangle \leq f_j(x) \leq f_j(y) + \langle \nabla f_j(y), x - y \rangle + \frac{1}{2} \|y - x\|_{\mathbf{M}_j}^2, \quad \forall x, y \in \mathbb{R}^d. \quad (5.10)$$

Assumption 5.10 generalizes classical L -smoothness, which is obtained in the special case $\mathbf{M}_j = L\mathbf{I}_d$. The usefulness of this assumption comes from i) the fact that ERM problems typically satisfy (5.10) in a non-trivial way [167, 60], ii) our method is able to utilize the full information contained in these matrices for further acceleration (via

increased stepsizes). Given matrices $\{\mathbf{M}_j\}$ from Assumption 5.4.1, let \mathcal{M} be the linear operator defined via $(\mathcal{M}\mathbf{X})_{:j} = \mathbf{M}_j\mathbf{X}_{:j}$ for $j \in [n]$. It is easy to check that this operator is self-adjoint and positive semi-definite, and that its square root is given by

$$\left(\mathcal{M}^{\frac{1}{2}}\mathbf{X}\right)_{:j} = \mathbf{M}_j^{\frac{1}{2}}\mathbf{X}_{:j}.$$

The pseudoinverse \mathcal{M}^\dagger of this operator plays an important role in our main result.

Theorem 5.4.2. *Let Assumption 5.4.1 hold. Let \mathcal{B} be any linear operator commuting with \mathcal{S} , and assume $\mathcal{M}^{\dagger\frac{1}{2}}$ commutes with \mathcal{S} . Let \mathcal{R} be any linear operator for which $\mathcal{R}(\mathbf{J}^k) = \mathcal{R}(\mathbf{G}(x^*))$ for every $k \geq 0$. Define the Lyapunov function*

$$\Psi^k \stackrel{\text{def}}{=} \|x^k - x^*\|^2 + \alpha \left\| \mathcal{B}\mathcal{M}^{\dagger\frac{1}{2}}(\mathbf{J}^k - \mathbf{G}(x^*)) \right\|^2, \quad (5.11)$$

where $\{x^k\}$ and $\{\mathbf{J}^k\}$ are the random iterates produced by Algorithm 14 with stepsize $\alpha > 0$. Suppose that α and \mathcal{B} are chosen so that

$$\frac{2\alpha}{n^2} \mathbb{E} [\|\mathcal{U}\mathbf{X}_e\|^2] + \left\| (\mathcal{I} - \mathbb{E}[\mathcal{S}])^{\frac{1}{2}} \mathcal{B}\mathcal{M}^{\dagger\frac{1}{2}}\mathbf{X} \right\|^2 \leq (1 - \alpha\mu) \left\| \mathcal{B}\mathcal{M}^{\dagger\frac{1}{2}}\mathbf{X} \right\|^2 \quad (5.12)$$

whenever $\mathbf{X} \in \text{Range}(\mathcal{R})^\perp$ and

$$\frac{2\alpha}{n^2} \mathbb{E} [\|\mathcal{U}\mathbf{X}_e\|^2] + \left\| (\mathbb{E}[\mathcal{S}])^{\frac{1}{2}} \mathcal{B}\mathcal{M}^{\dagger\frac{1}{2}}\mathbf{X} \right\|^2 \leq \frac{1}{n} \left\| \mathcal{M}^{\dagger\frac{1}{2}}\mathbf{X} \right\|^2. \quad (5.13)$$

for all $\mathbf{X} \in \mathbb{R}^{d \times n}$. Then for all $k \geq 0$, we have $\mathbb{E}[\Psi^k] \leq (1 - \alpha\mu)^k \Psi^0$.

The above theorem is very general as it applies to essentially arbitrary random linear operators \mathcal{S} and \mathcal{U} . It postulates a linear convergence rate of a Lyapunov function composed of two terms: distance of x^k from x^* , and weighted distance of the Jacobian \mathbf{J}^k from $\mathbf{G}(x^*)$. Hence, we obtain convergence of both the iterates and the Jacobian to x^* and $\mathbf{G}(x^*)$, respectively. Inequalities (5.12) and (5.13) are mainly assumptions one stepsize α , and are used to define suitable weight operator \mathcal{B} . See Lemma E.2.1 for a general statement on when these inequalities are satisfied. However, we give concrete and simple answers in all special cases of GJS in the appendix. For a summary of how the operator \mathcal{B} is chosen in special cases, and the particular complexity results derived from this theorem, we refer to Table E.1.

Remark 5. We use the trivial choice $\mathcal{R} \equiv 0$ in almost all special cases. With this choice of \mathcal{R} , the condition $\mathcal{R}(\mathbf{J}^k) = \mathcal{R}(\mathbf{G}(x^*))$ is automatically satisfied, and inequality (5.13) is requested to hold for *all* matrices $\mathbf{X} \in \mathbb{R}^{d \times n}$. However, a non-trivial choice of \mathcal{R} is sometimes useful; e.g., in the analysis of a subspace variant of SEGA [77]. Further, the results of Theorem 5.4.2 can be generalized from a quasi strong convexity to a strong growth condition [91] on f (see Appendix E.13). While interesting, these are not the key results of this work and we therefore suppress them to the appendix.

Choice of random operators \mathcal{S} and \mathcal{U} defining Algorithm 14		Algorithm		
$\mathcal{S}\mathbf{X}$	$\mathcal{U}\mathbf{X}$	#	Name	Comment
$\mathbf{X}e_j e_j^\top$ w.p. $p_j = \frac{1}{n}$	$\mathbf{X}n e_j e_j^\top$ w.p. $p_j = \frac{1}{n}$	27	SAGA	basic variant of SAGA [37]
$\mathbf{X} \sum_{j \in R} e_j e_j^\top$ w.p. p_R	$\mathbf{X} \sum_{j \in R} \frac{1}{p_j} e_j e_j^\top$ w.p. p_R	28	SAGA	SAGA with AS [165]
$e_i e_i^\top \mathbf{X}$ w.p. $p_i = \frac{1}{d}$	$d e_i e_i^\top \mathbf{X}$ w.p. $p_i = \frac{1}{d}$	29	SEGA	basic variant of SEGA [77]
$\sum_{i \in L} e_i e_i^\top \mathbf{X}$ w.p. p_L	$\sum_{i \in L} \frac{1}{p_i} e_i e_i^\top \mathbf{X}$ w.p. p_L	30	SEGA	SEGA [77] with AS and prox
$= \begin{cases} 0 & \text{w.p. } 1 - \rho \\ \mathbf{X} & \text{w.p. } \rho \end{cases}$	$\sum_{i \in L} \frac{1}{p_i} e_i e_i^\top \mathbf{X}$ w.p. p_L	31	SVRCD	NEW
0	$\mathbf{X} \sum_{j \in R} \frac{1}{p_j} e_j e_j^\top$ w.p. p_R	32	SGD-star	SGD-star [55] with AS
$= \begin{cases} 0 & \text{w.p. } 1 - \rho \\ \mathbf{X} & \text{w.p. } \rho \end{cases}$	$\mathbf{X} \sum_{j \in R} \frac{1}{p_j} e_j e_j^\top$ w.p. p_R	33	LSVRG	LSVRG [106] with AS and prox
$= \begin{cases} 0 & \text{w.p. } 1 - \rho \\ \mathbf{X} & \text{w.p. } \rho \end{cases}$	$= \begin{cases} 0 & \text{w.p. } 1 - \delta \\ \frac{1}{\delta} \mathbf{X} & \text{w.p. } \delta \end{cases}$	34	B2	NEW
$\mathbf{X} \sum_{j \in R} e_j e_j^\top$ w.p. p_R	$= \begin{cases} 0 & \text{w.p. } 1 - \delta \\ \frac{1}{\delta} \mathbf{X} & \text{w.p. } \delta \end{cases}$	35	LSVRG-inv	NEW
$\sum_{i \in L} e_i e_i^\top \mathbf{X}$ w.p. p_L	$= \begin{cases} 0 & \text{w.p. } 1 - \delta \\ \frac{1}{\delta} \mathbf{X} & \text{w.p. } \delta \end{cases}$	36	SVRCD-inv	NEW
$\mathbf{X} \sum_{j \in R} e_j e_j^\top$ w.p. p_R	$\sum_{i \in L} \frac{1}{p_i} e_i e_i^\top \mathbf{X}$ w.p. p_L	37	RL	NEW
$\sum_{i \in L} e_i e_i^\top \mathbf{X}$ w.p. p_L	$\mathbf{X} \sum_{j \in R} \frac{1}{p_j} e_j e_j^\top$ w.p. p_R	38	LR	NEW
$\mathbf{I}_L; \mathbf{X} \mathbf{I}_{:R}$ w.p. $p_L p_R$	$\mathbf{I}_L; \left(\left(p^{-1} (p^{-1})^\top \right) \circ \mathbf{X} \right) \mathbf{I}_{:R}$ w.p. $p_L p_R$	39	SAEGA	NEW
$= \begin{cases} 0 & \text{w.p. } 1 - \rho \\ \mathbf{X} & \text{w.p. } \rho \end{cases}$	$\mathbf{I}_L; \left(\left(p^{-1} (p^{-1})^\top \right) \circ \mathbf{X} \right) \mathbf{I}_{:R}$ w.p. $p_L p_R$	40	SVRCDG	NEW
$\sum_{t=1}^T \mathbf{I}_{L_t}; \mathbf{X}_{:N_t} \mathbf{I}_{:R_t}$	$\sum_{t=1}^T \left((p^t)^{-1} (p^t)^{-1\top} \right) \circ (\mathbf{I}_{L_t}; \mathbf{X}_{:N_t} \mathbf{I}_{:R_t})$	41	ISAEGA	NEW (reminiscent of [137])
$\sum_{t=1}^T \mathbf{I}_{L_t}; \mathbf{X}_{:N_t}$	$\sum_{t=1}^T \left((p^t)^{-1} e^\top \right) \circ (\mathbf{I}_{L_t}; \mathbf{X}_{:N_t})$	42	ISEGA	ISEGA [137] with AS
$\mathbf{X}\mathbf{R}$	$\mathbf{X}\mathbf{R}\mathbf{E}[\mathbf{R}]^{-1}$	43	JS	JacSketch [65] with AS and prox

Table 5.1: Selected special cases of GJS (Algorithm 14) arising by choosing operators \mathcal{S} and \mathcal{U} in particular ways. R is a random subset of $[n]$, L is a random subset of $[d]$, $p_i = \mathbb{P}(i \in L)$, $p_j = \mathbb{P}(j \in R)$.

5.5 Special cases

As outlined in the introduction, GJS (Algorithm 14) is a surprisingly versatile method. In Table 5.1 we list 7 *existing methods* (in some cases, generalizations of existing methods), and construct also 10 *new variance reduced methods*. We also provide a summary of all specialized iteration complexity results, and a guide to the corollaries which state them (see Table E.1 in the appendix).

- **SGD-star**. In order to illustrate why variance reduction is needed in the first place, let us start by describing one of the methods—SGD-star (Algorithm 32)—which happens to be particularly suitable to shed light on this issue. In SGD-star we assume that the Jacobian at optimum, $\mathbf{G}(x^*)$, is known. While this is clearly an unrealistic assumption, let us see where it leads us. If this is the case, we can choose $\mathbf{J}^0 = \mathbf{G}(x^*)$, and let $\mathcal{S} \equiv 0$. This implies that $\mathbf{J}^k = \mathbf{J}^0$ for all k . We then choose \mathcal{U} to be the right unbiased sampling operator, i.e., $\mathcal{U}\mathbf{X} = \mathbf{X} \sum_{j \in R} \frac{1}{p_j} e_j e_j^\top$, which gives

$$g^k = \frac{1}{n} \sum_{j=1}^n \nabla f_j(x^*) + \sum_{j \in R^k} \frac{1}{n p_j} (\nabla f_j(x^k) - \nabla f_j(x^*)).$$

This method does not need to learn the Jacobian at x^* as it is known, and instead moves in a direction of average gradient at the optimum, perturbed by a random

estimator of the direction $\nabla f(x^k) - \nabla f(x^*)$ formed via sub-sampling $j \in R^k \subseteq [n]$. What is special about this perturbation? As the method converges, $x^k \rightarrow x^*$ and the perturbations converge to zero, for *any* realization of the random set $R^k \subseteq [n]$. So, gradient estimation stabilizes, we get $g^k \rightarrow \nabla f(x^*)$, and hence the variance of g^k converges to zero. In view of Corollary E.6.1 of our main result (Theorem 5.4.2), the iteration complexity of SGD-star is $\max_j \frac{v_j}{\mu n p_j} \log \frac{1}{\epsilon}$, where μ is the quasi strong convexity parameter of f , and the smoothness constants v_j are defined in Appendix E.6.

Since knowing $G(x^*)$ is unrealistic, GJS is instead *learning* these perturbations on the fly. Different variants of GJS do this differently, but ultimately all attempt to learn the gradients $\nabla f_j(x^*)$ and use this information to stabilize the gradient estimation. Due to space restrictions, we do not describe all remaining 9 new methods in the main body of the chapter, let alone the all 17 methods. We will briefly outline 2 more (not necessarily the most interesting) new methods:

- SVRCD. This method belongs to the RCD variety, and constructs the gradient estimator via the rule

$$g^k = h^k + \sum_{i \in L^k} \frac{1}{\textcolor{red}{p}_i} (\nabla_i f(x^k) - h_i^k) \textcolor{red}{e}_i,$$

where $L^k \subseteq [d]$ is sampled afresh in each iteration. The auxiliary vector h^k is updated using a simple biased coin flip: $h^{k+1} = h^k$ with probability $1 - \rho$, and $h^{k+1} = \nabla f(x^k)$ with probability ρ . So, a full pass over all coordinates is made in each iteration with probability ρ , and a partial derivatives $\nabla_i f(x^k)$ for $i \in L^k$ are computed in each iteration. This method has a similar structure to LSVRG, which instead sub-sampling coordinates sub-samples functions f_j for $j \in R^k$ (see Table 5.1). The iteration complexity of this method is $\left(\frac{1}{\rho} + \max_i \frac{1}{\textcolor{red}{p}_i} \frac{4m_i}{\mu}\right) \log \frac{1}{\epsilon}$, where m_i is a smoothness parameter of f associated with coordinate i (see Table E.1 and Corollary E.5.3).

- ISAEGA. In Chapter 4, a strategy of running RCD on top of a parallel implementation of optimization algorithms such as PGD, SGD or SAGA was proposed. Surprisingly, it was shown that the runtime of the overall algorithm is unaffected whether one computes and communicates *all entries* of the stochastic gradient on each worker, or only a *fraction* of all entries of size inversely proportional to the number of all workers. However, ISAGA [137] (distributed SAGA with RCD on top of it), as proposed, requires the gradients with respect to the data owned by a given machine to be zero at the optimum. On the other hand, ISEGA [137] does not have the issue, but it requires a computation of the exact partial derivatives on each machine and thus is expensive. As a special case of GJS we propose ISAEGA – a method which cherry-picks the best properties from both ISAGA (allowing for stochastic partial derivatives) and ISEGA (not requiring zero gradients at the optimum). Further, we present the method in the arbitrary sampling paradigm. See Appendix E.10.3 for more details.

5.6 Experiments

We perform extensive numerical testing for various special cases of Algorithm 14. We first start with perfectly understood example – minimizing artificial quadratics. After that, we present experiments on logistic regression with real-world data.

5.6.1 SEGA and SVRCD with importance sampling

In Sections E.5.2 and E.5.3 we develop an arbitrary (and thus importance in special case) sampling for SEGA, as well as new method SVRCD with arbitrary sampling. In this experiment, we compare them to its natural competitors – basic SEGA from [77] and proximal gradient descent.

Consider artificial quadratic minimization with regularizer ψ being an indicator of the unit ball⁵:

$$f(x) = x^\top \mathbf{M}x - b^\top x, \quad \psi(x) = \begin{cases} x & 0 \leq 1 \\ \infty & \|x\| > 1 \end{cases}.$$

Specific choices of \mathbf{M}, b are given by Table 5.2. As both SEGA and SVRCD (from Section E.5.2 and E.5.3) require a diagonal smoothness matrix, we shall further consider vector m such that the upper bound $\mathbf{M} \preceq \text{Diag}(m)$ holds. As the choice of m is not unique, we shall choose the one which minimizes $\sum_{i=1}^d m_i$ for importance sampling and $m = \lambda_{\max}(\mathbf{M})\mathbf{e}$ for uniform. Further, stepsize $\gamma = \frac{1}{4 \sum_{i=1}^d m_i}$ was chosen in each case. Figure 5.1 shows the results of this experiment. As theory suggests, importance sampling for both SEGA and SVRCD outperform both plain SEGA and proximal gradient always. The performance difference depends on the data; the closer \mathbf{M} is to a diagonal matrix with non-uniform elements, the larger stronger is the effect of importance sampling.

Type	\mathbf{M}	b
1	$\text{Diag}(1.3^{[d]})$	γu
2	$\text{Diag}((d, 1, 1, \dots, 1))$	γu
3	$\text{Diag}(1.1^{[d]}) + \mathbf{N}\mathbf{N}^\top \frac{1.1^d}{1000d}, \mathbf{N} \sim N(0, \mathbf{I})$	γu
4	$\mathbf{N}\mathbf{N}^\top, \mathbf{N} \sim N(0, \mathbf{I})$	γu

Table 5.2: Four types of quadratic problems. We choose $u \sim N(0, \mathbf{I}_d)$, and γ to be such that $\|\gamma \mathbf{M}^{-1}u\| = \frac{3}{2}$. Notation $c^{[d]}$ stands for a vector (c, c^2, \dots, c^d) .

5.6.2 SVRCD: effect of ρ

In this experiment we demonstrate very broad range of ρ can be chosen to still attain almost best possible rate for SVRCD for problems from Table 5.2 and m, γ as described in Section 5.6.1 Results can be found in Figure 5.2. They indeed show that in many cases, varying ρ from $\frac{1}{n}$ down to $\frac{2\lambda_{\min}(\mathbf{M})}{\sum_{i=1}^d m_i}$ does not influence the complexity significantly.

⁵In such case, proximal operator of ψ becomes a projection onto the unit ball.

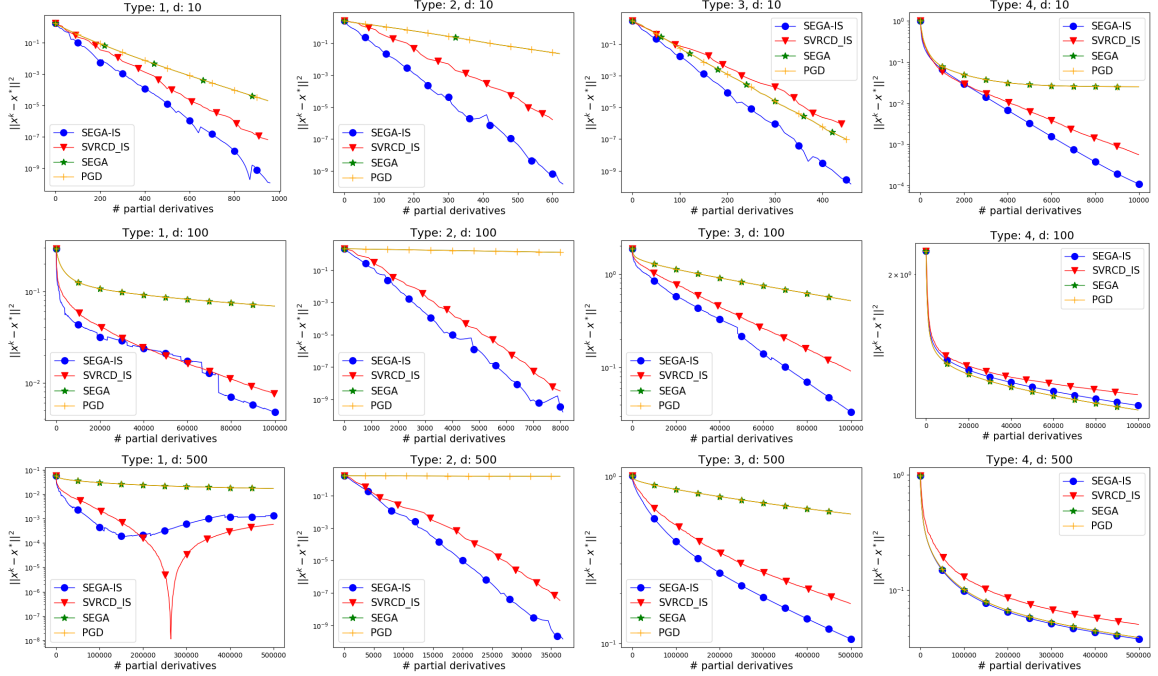


Figure 5.1: Comparison of SEGA-AS, SVRCD-AS, SEGA and proximal gradient on 4 quadratic problems given by Table 5.2. SEGA-AS, SVRCD-AS and SEGA compute single partial derivative each iteration (SVRCD computes all of them with probability ρ), SEGA-AS, SVRCD-AS with probabilities proportional to diagonal of \mathbf{M} .

However, too small ρ leads to significantly slower convergence. Note that those findings are in accord with Corollary E.5.3. Similar results were shown in [106] for LSVRG.

5.6.3 ISAEGA

In this section we test a simple version of ISAEGA (Algorithm 41)⁶. As mentioned, ISAEGA is an algorithm for distributed optimization which, at each iteration, computes a subset of partial derivatives of stochastic gradient on each machine, and constructs corresponding Jacobian estimate and stochastic gradient.

For simplicity, we consider only the simple version which assumes $\mathbf{M}_j = m\mathbf{I}_d$ for all j (i.e. we do not do importance sampling), and we suppose that $|R_t| = 1$ always for all t (i.e. each machine always looks at a single function from the local finite sum). Further, we consider $\psi(x) = 0$. Corollary E.10.3 shows that, if the condition number of the problem is not too small, ISAEGA with $|L_t| \approx \frac{1}{T}$ (where T is a number of parallel units) enjoys, up to small constant factor, same rate as SAGA (which is, under a convenient smoothness, the same rate as the convergence rate of gradient descent). Thus, ISAEGA scales linearly in terms of partial derivative complexity in parallel setup. In other words, given that we have twice more workers, each of them can afford to evaluate twice less partial derivatives⁷. The experiments we propose aim to verify this claim.

⁶The full description of ISAEGA, together with convergence guarantees are provided in Section E.10.3

⁷Practical implications of the method are further explained in [137].

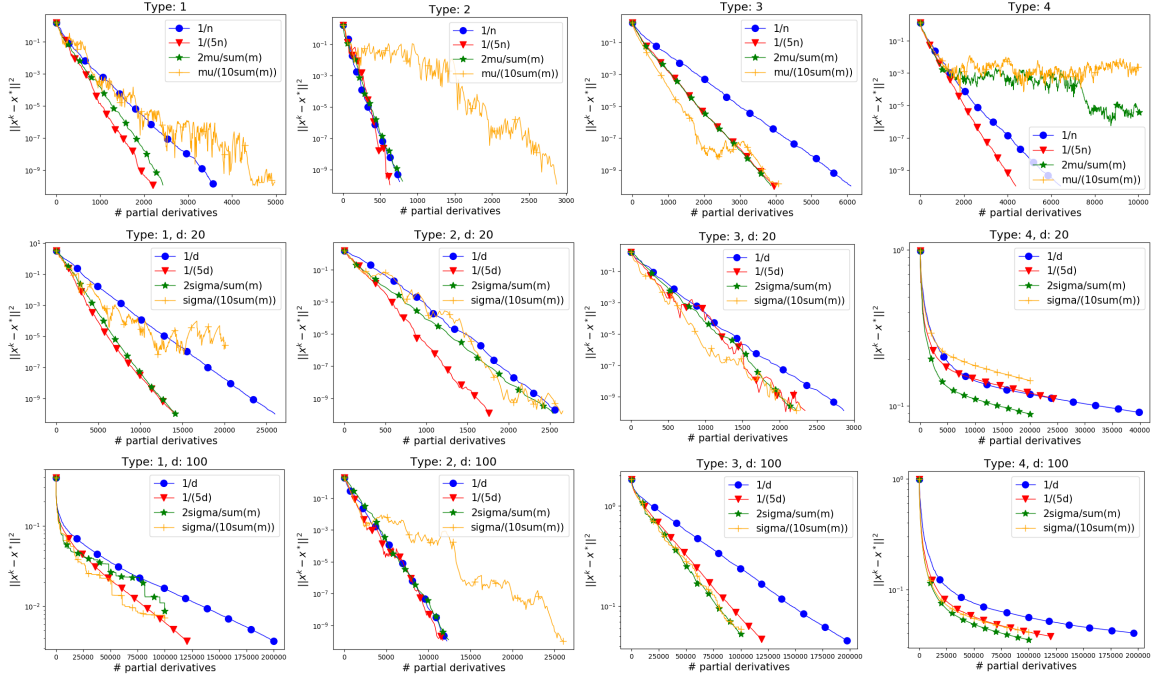


Figure 5.2: The effect of ρ on convergence rate of SVRCD on quadratic problems from Table 5.2. In every case, probabilities were chosen proportionally to the diagonal of \mathbf{M} and only a single partial derivative is evaluated in \mathcal{S} .

We consider ℓ_2 regularized logistic regression (for the binary classification). In particular,

$$\forall j : \quad f_j(x) \stackrel{\text{def}}{=} \log(1 + \exp(\mathbf{A}_{j,:}x \cdot y_i)) + \frac{\lambda}{2}\|x\|^2,$$

where $\mathbf{A} \in \mathbb{R}^{n \times d}$ is a data matrix, $y \in \{-1, 1\}^n$ is a vector of labels and $\lambda \geq 0$ is the regularization parameter. Both \mathbf{A}, y are provided from LibSVM [23] datasets: a1a, a9a, w1a, w8a, gisette, madelon, phishing and mushrooms. Further, \mathbf{A} was normalized such that $\|\mathbf{A}_{j,:}\|^2 = 1$. Next, it is known that f_j is $(\frac{1}{4} + \lambda)$ -smooth, convex, while f is λ -strongly convex. Therefore, as a stepsize for all versions of ISAEAG, we set $\gamma = \frac{1}{6\lambda + \frac{3}{2}}$ (this is an approximation of theoretical stepsize).

In each experiment, we compare 4 different setups for ISEAGA – given by 4 different values of T . Given a value of T , we set $|L_t| = \frac{1}{T}$ for all t . Further, we always sample L_t uniformly. The results are presented in Figure 5.3. Indeed, we observe the almost perfect parallel linear scaling.

For completeness, we provide dataset sized in Table 5.3.

5.6.4 LSVRG with importance sampling

As mentioned, one of the contributions of this work is LSVRG with arbitrary sampling. In this section, we demonstrate that designing a good sampling can yield a significant speedup in practice. We consider logistic regression problem on LibSVM [23] data, as described in Section 5.6.3. However, since LibSVM data are normalized, we pre-multiply

Name	n	d
a1a	1605	123
a9a	32561	123
w1a	2477	300
w8a	49749	300
gisette	6000	5000
madelon	2000	500
phishing	11055	68
mushrooms	8124	112

Table 5.3: Table of LibSVM data used for our experiments.

each row of the data matrix by a random scaling factor. In particular, the scaling factors are proportional to l^2 where l is sampled uniformly from $[1000]$ such that the Frobenius norm of the data matrix is n . For the sake of simplicity, consider case $\lambda = 0$.

Choice vector v . Note that since $\mathbf{M}_j = \mathbf{A}_{j:}^\top \mathbf{A}_{j:}$, the following claim must hold: *Consider fixed v . Then if (E.18) holds for any set of vector $\{h_j\}_{j=1}^n$ such that h_j is parallel to $\mathbf{A}_{j:}$, then (E.18) holds for any set of vector $\{h_j\}_{j=1}^n$.* Thus, we can set $h_j = c_j \mathbf{A}_{j:}^\top / \|\mathbf{A}_{j:}\|$ without loss of generality. Thus, $\mathbf{M}_j^{\frac{1}{2}} h_j = c_j \mathbf{A}_{j:}^\top$, and (E.18) becomes equivalent to $\mathbf{P} \circ (\mathbf{A}^\top \mathbf{A}) \preceq \mathbf{Diag}(p \circ v)$ where $\mathbf{P}_{jj'} = \mathbb{P}(j \in R, j' \in R)$. Note that this is exactly expected separable overapproximation (ESO) for coordinate descent [167]. Thus we choose vector v to be proportional to p such that $\mathbf{P} \circ (\mathbf{A}^\top \mathbf{A}) \preceq \mathbf{Diag}(p \circ v)$ holds (as proposed in Chapter 2). In order to compute the scaling constant, one needs to evaluate maximum eigenvalue of PSD $n \times n$ matrix, which is of $\mathcal{O}(n^2)$ cost. We do so in the experiments. Note that there is a suboptimal, but cheaper way to obtain v described in [165]. Lastly, if $\lambda > 0$, we set v such that $\mathbf{P} \circ (\mathbf{A}^\top \mathbf{A} + \lambda \mathbf{I}) \preceq \mathbf{Diag}(p \circ v)$.

Choice of probabilities. In order to be fair, we only compare methods where $\mathbb{E}[|R|] = \tau$. For the case $\tau = 1$, we consider a sampling such that $|R| = 1$ according to a given probability vector p . For uniform sampling, we have $p = n^{-1} \mathbf{e}$, while for importance sampling, we set $p_j = \frac{\lambda_{\max}(\mathbf{M}_j)}{\sum_{j'=1}^n \lambda_{\max}(\mathbf{M}_{j'})}$. In the case $\tau > 1$, we consider independent sampling from Chapter 2. In particular, $\mathbb{P}(j \in R) = p_j$ with $\sum p_j = \tau$ and binary random variables ($j \in R$) are jointly independent. For uniform sampling we have $p = \tau n^{-1} \mathbf{e}$. For importance sampling, probability vector p is chosen such that $p_j = \frac{\lambda_{\max}(\mathbf{M}_j)}{\varrho + \lambda_{\max}(\mathbf{M}_j)}$, where ϱ is such that $\sum p_j = \tau$. The mentioned sampling was proven to be superior over uniform minibatching in Chapter 2. Next, stepsize $\gamma = \frac{1}{6} \min_j \frac{np_j}{v_j}$ was chosen for all methods.

Lastly, $\rho = \frac{1}{2n}$ was chosen for LSVRG. The results are presented in Figures 5.4 and 5.5.

In all cases, LSVRG with importance sampling was the fastest method. As provided theory suggests, it outperformed methods with importance sampling especially significantly for small τ ; and the larger τ , the smaller the effect of importance sampling is. However, our experiments indicate the superiority of LSVRG to SAGA in the importance sampling

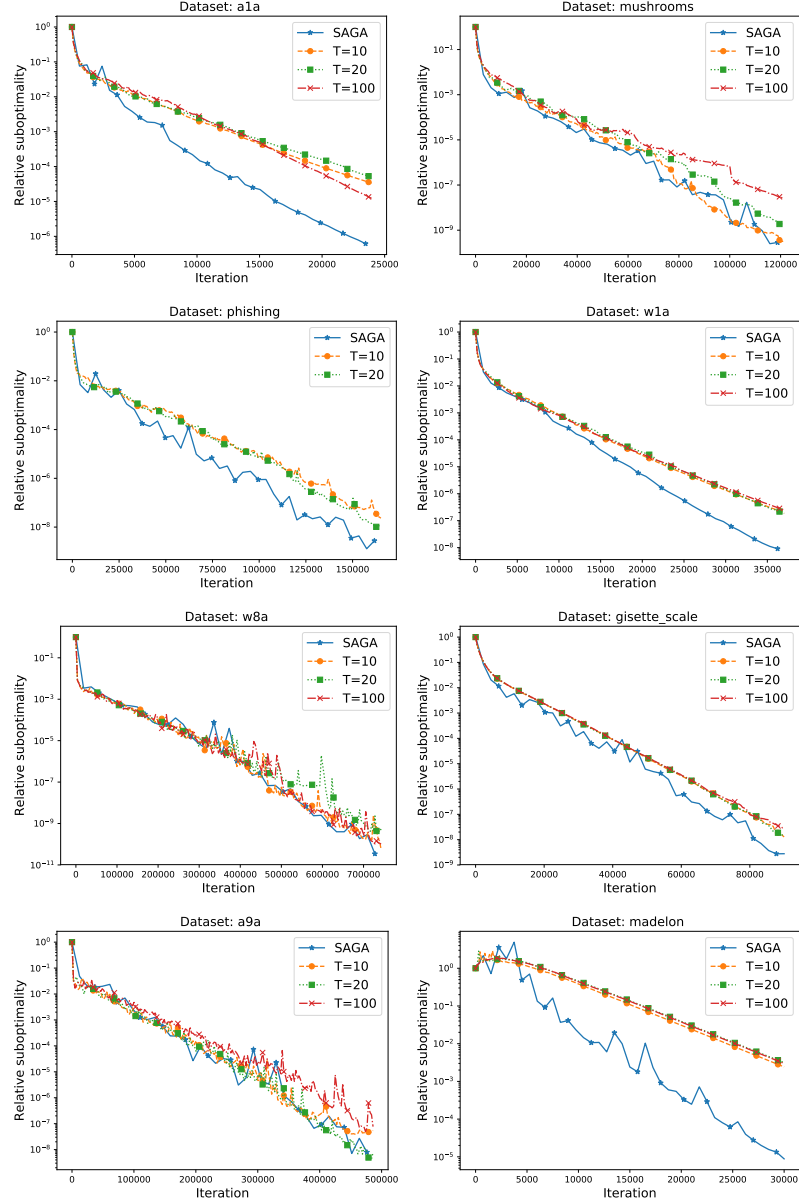


Figure 5.3: ISAEGA applied on LIBSVM [23] datasets with $\lambda = 4 \cdot 10^{-5}$. Axis y stands for relative suboptimality, i.e. $\frac{f(x^k) - f(x^*)}{f(x^k) - f(x^0)}$.

setup. In particular, stepsize $\gamma = \frac{1}{6} \min_j \frac{np_j}{v_j}$ is often too large for SAGA. Note that both optimal stepsize and optimal probabilities require the prior knowledge of the quasi strong convexity constant μ^8 which is, in our case unknown (see the importance serial sampling proposed in [65], and SAGA is more sensitive to that choice. One can still estimate it as λ , however, this would yield suboptimal performance as well.

⁸Or more generally, strong growth constant, see Appendix E.13

5.7 Conclusion

In this chapter we proposed a fairly general algorithm—GJS—capable of inserting the variance reduction mechanism under arbitrary random first-order oracle. Each special case either recovers a known algorithm with its tight rate, or improves a known algorithm or is a new algorithm. In the next chapter we go even further: we introduce a general technique to analyze unbiased stochastic gradient algorithms that are not necessarily variance reduced.

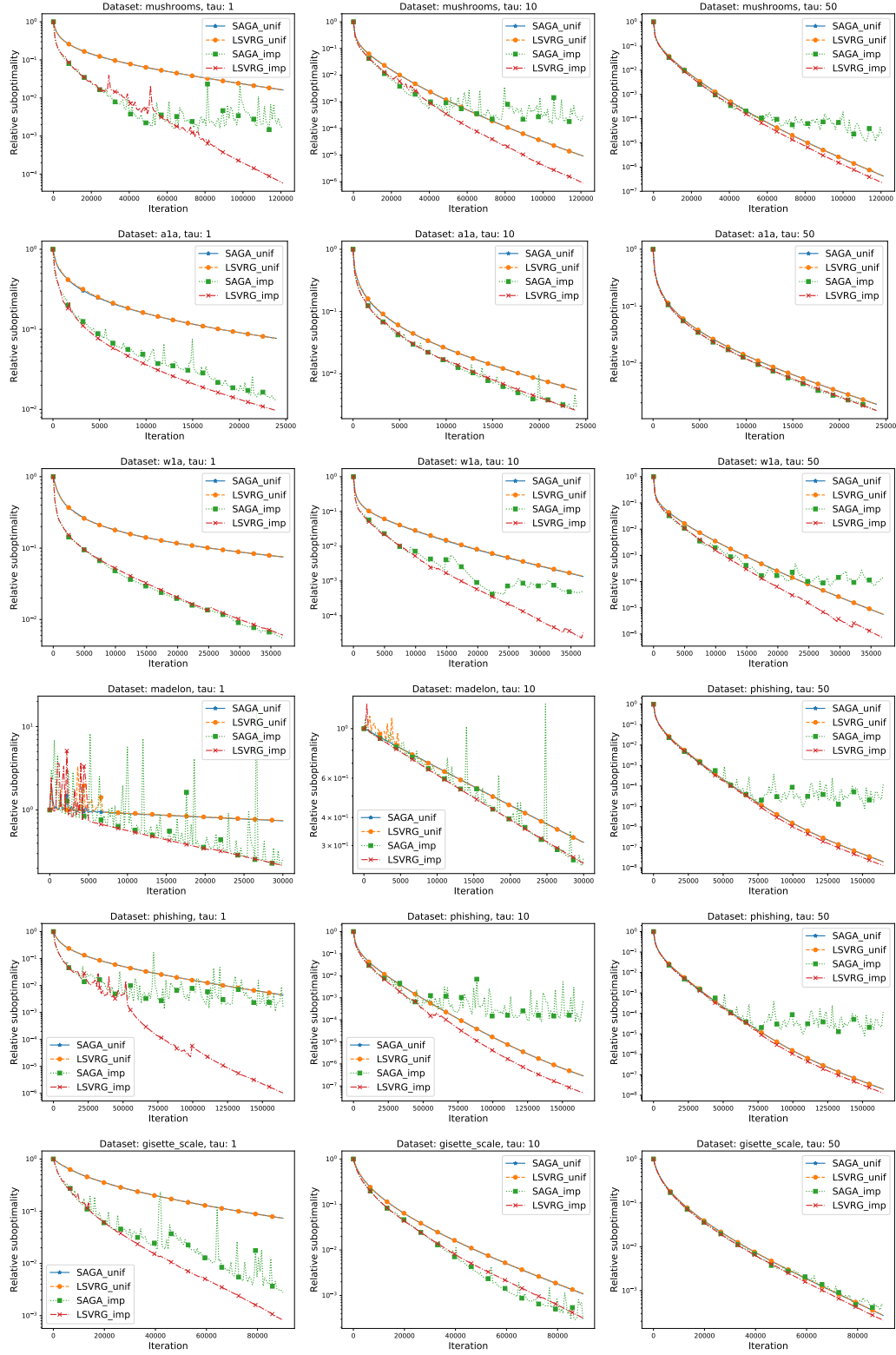


Figure 5.4: LSVRG applied on LIBSVM [23] datasets with $\lambda = 10^{-5}$. Axis y stands for relative suboptimality, i.e. $\frac{f(x^k) - f(x^*)}{f(x^k) - f(x^0)}$.

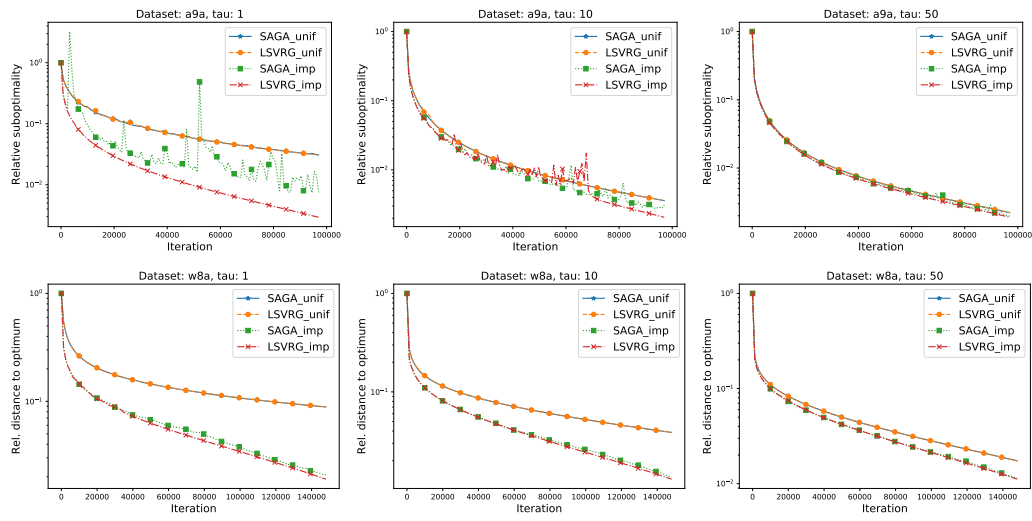


Figure 5.5: LSVRG applied on LIBSVM [23] datasets. For a9a, $\lambda = 0$ and $\rho = \frac{1}{n}$ was chosen; for w8a, $\lambda = 10^{-8}$ and $\rho = \frac{3}{n}$ was chosen. Axis y stands for relative suboptimality, i.e. $\frac{f(x^k) - f(x^*)}{f(x^k) - f(x^0)}$.

Chapter 6

A Unified Theory of SGD: Variance Reduction, Sampling, Quantization and Coordinate Descent

In Chapter 5, we have proposed a general variance-reduced algorithm applicable in many different scenarios. In this chapter, we go a step further. In particular, we propose a new generic analysis technique capable of providing complexity bounds for a significantly broader class of stochastic gradient algorithms.

Stochastic optimization. In this chapter we are primarily concerned with regularized stochastic optimization problems of the form

$$\min_{x \in \mathbb{R}^d} \mathbb{E}_{\xi \sim \mathcal{D}} [f_{\xi}(x)] + \psi(x), \quad (6.1)$$

and let

$$f(x) = \mathbb{E}_{\xi \sim \mathcal{D}} [f_{\xi}(x)]. \quad (6.2)$$

As usual, function f is assumed to be convex, differentiable with Lipschitz gradient, and $\psi : \mathbb{R}^d \rightarrow \mathbb{R} \cup \{+\infty\}$ is a proximable (proper closed convex) regularizer. Specifically for this section, we assume that ξ is a random variable, and $f_{\xi} : \mathbb{R}^d \rightarrow \mathbb{R}$ is smooth function for all ξ .

Stochastic optimization problems are of key importance in statistical supervised learning theory. In this setup, x represents a machine learning model described by d parameters (e.g., logistic regression or a deep neural network), \mathcal{D} is an unknown distribution of labelled examples, $f_{\xi}(x)$ represents the loss of model x on datapoint ξ , and f is the generalization error. Problem (6.1) seeks to find the model x minimizing the generalization error. In statistical learning theory one assumes that while \mathcal{D} is not known, samples $\xi \sim \mathcal{D}$ are available. In such a case, $\nabla f(x)$ is not computable, while $\nabla f_{\xi}(x)$, which is an unbiased estimator of the gradient of f at x , is easily computable.

Finite-sum problems. Another prominent example, one of special interest in this work, are functions f which arise as averages of a very large number of smooth functions:

$$f(x) = \frac{1}{n} \sum_{i=1}^n f_i(x). \quad (6.3)$$

This problem often arises by approximation of the stochastic optimization loss function (6.2) via Monte Carlo integration, and is in this context known as the empirical risk minimization (ERM) problem. ERM is currently the dominant paradigm for solving supervised

learning problems [188]. If index i is chosen uniformly at random from $[n] \stackrel{\text{def}}{=} \{1, 2, \dots, n\}$, $\nabla f_i(x)$ is an unbiased estimator of $\nabla f(x)$. Typically, $\nabla f(x)$ is about n times more expensive to compute than $\nabla f_i(x)$.

Distributed optimization. Lastly, in some applications, especially in distributed training of supervised models, one considers problem (6.3), with n being the number of machines, and each f_i also having a finite sum structure, i.e.,

$$f_i(x) = \frac{1}{m} \sum_{j=1}^m f_{ij}(x), \quad (6.4)$$

where m corresponds to the number of training examples stored on machine i .

6.1 The many faces of stochastic gradient descent

Stochastic gradient descent (SGD) [179, 148, 208] is a state-of-the-art algorithmic paradigm for solving optimization problems (6.1) in situations when f is either of structure (6.2) or (6.3). In its generic form, (proximal) SGD defines the new iterate by subtracting a multiple of a stochastic gradient from the current iterate, and subsequently applying the proximal operator of ψ :

$$x^{k+1} = \text{prox}_{\alpha\psi}(x^k - \alpha g^k). \quad (6.5)$$

Here, g^k is an unbiased estimator of the gradient (i.e., a stochastic gradient),

$$\mathbb{E}[g^k | x^k] = \nabla f(x^k), \quad (6.6)$$

and $\text{prox}_{\alpha\psi}(x) = \arg \min_y \{\alpha\psi(y) + \frac{1}{2} \|y - x\|^2\}$. However, and this is the starting point of our journey in this chapter, there are *infinitely many* ways of obtaining a random vector g^k satisfying (6.6). On the one hand, this gives algorithm designers the flexibility to *construct* stochastic gradients in various ways in order to target desirable properties such as convergence speed, iteration cost, parallelizability and generalization. On the other hand, this poses considerable challenges in terms of convergence analysis. Indeed, if one aims to, as one should, obtain the sharpest bounds possible, dedicated analyses are needed to handle each of the particular variants of SGD.

Vanilla SGD. The flexibility in the design of efficient strategies for constructing g^k has led to a creative renaissance in the optimization and machine learning communities, yielding a large number of immensely powerful new variants¹ of SGD, such as those employing *importance sampling* [223, 146], and *mini-batching* [102]. These efforts are subsumed by the recently developed and remarkably sharp analysis of SGD under *arbitrary sampling* paradigm [60], first introduced in the study of randomized coordinate descent methods by [175]. The arbitrary sampling paradigm covers virtually all stationary mini-batch and

¹In this chapter, by *vanilla* SGD we refer to SGD variants with or without importance sampling and mini-batching, but *excluding* variance-reduced variants, such as SAGA [37] and SVRG [88].

importance sampling strategies in a unified way, thus making headway towards theoretical unification of two separate strategies for constructing stochastic gradients. For strongly convex f , the SGD methods analyzed in [60] converge linearly to a neighbourhood of the solution $x^* = \arg \min_x f(x)$ for a fixed stepsize $\alpha^k = \alpha$. The size of the neighbourhood is proportional to the second moment of the stochastic gradient at the optimum ($\sigma^2 \stackrel{\text{def}}{=} \frac{1}{n} \sum_{i=1}^n \|\nabla f_i(x^*)\|^2$), to the stepsize (α), and inversely proportional to the modulus of strong convexity. The effect of various sampling strategies, such as importance sampling and mini-batching, is twofold: i) improvement of the linear convergence rate by enabling larger stepsizes, and ii) modification of σ^2 . However, none of these strategies² is able to completely eliminate the adverse effect of σ^2 . That is, SGD with a fixed stepsize does not reach the optimum, unless one happens to be in the overparameterized case characterized by the identity $\sigma^2 = 0$.

Variance reduced SGD. While sampling strategies such as importance sampling and mini-batching reduce the variance of the stochastic gradient, in the finite-sum case (6.3) a new type of *variance reduction* strategies has been developed over the last few years [182, 37, 88, 191, 169, 160, 106, 86] (see also Chapter 5). These variance-reduced SGD methods differ from the sampling strategies discussed before in a significant way: they can iteratively *learn* the stochastic gradients at the optimum, and in so doing are able to eliminate the adverse effect of the gradient noise $\sigma^2 > 0$ which, as mentioned above, prevents the iterates of vanilla SGD from converging to the optimum. As a result, for strongly convex f , these new variance-reduced SGD methods converge linearly to x^* , with a fixed stepsize. At the moment, these variance-reduced variants require a markedly different convergence theory from the vanilla variants of SGD. An exception to this is the situation when $\sigma^2 = 0$ as then variance reduction is not needed; indeed, vanilla SGD already converges to the optimum, and with a fixed stepsize. We end the discussion here by remarking that this *hints* at a possible existence of a more unified theory, one that would include both vanilla and variance-reduced SGD.

Distributed SGD, quantization and variance reduction. When SGD is implemented in a distributed fashion, the problem is often expressed in the form (6.3), where n is the number of workers/nodes, and f_i corresponds to the loss based on data stored on node i . Depending on the number of data points stored on each node, it may or may not be efficient to compute the gradient of f_i in each iteration. In general, SGD is implemented in this way: each node i first computes a stochastic gradient g_i^k of f_i at the current point x^k (maintained individually by each node). These gradients are then aggregated by a master node [193, 105], in-network by a switch [184], or a different technique best suited to the architecture used. To alleviate the communication bottleneck, various lossy update compression strategies such as quantization [187, 71, 222], sparsification [105, 3, 212] and dithering [2] were proposed. The basic idea is for each worker to apply a randomized transformation $Q : \mathbb{R}^d \rightarrow \mathbb{R}^d$ to g_i^k , resulting in a vector which is still an unbiased estimator of the gradient, but one that can be communicated with fewer bits. Mathematically, this amounts to injecting additional noise into the already noisy stochastic gradient g_i^k .

²Except for the full batch strategy, which is prohibitively expensive.

The field of quantized SGD is still young, and even some basic questions remained open until recently. For instance, there was no distributed quantized SGD capable of provably solving (6.1) until the DIANA algorithm [136] was introduced. DIANA applies quantization to *gradient differences*, and in so doing is able to learn the gradients at the optimum, which makes it able to work for any regularizer ψ . DIANA has some structural similarities with SEGA [77]—the first coordinate descent type method which works for non-separable regularizers—but a more precise relationship remains elusive. When the functions of f_i are of a finite-sum structure as in (6.4), one can apply variance reduction to reduce the variance of the stochastic gradients g_i^k together with quantization, resulting in the VR-DIANA method [85]. This is the first distributed quantized SGD method which provably converges to the solution of (6.1)+(6.4) with a fixed stepsize.

Randomized coordinate descent (RCD). Lastly, in a distinctly separate strain, there are SGD methods for the coordinate/subspace descent variety [152]. While it is possible to see *some* RCD methods as special cases of (6.5)+(6.6), most of them do not follow this algorithmic template. First, standard RCD methods use different stepsizes for updating different coordinates [166], and this seems to be crucial to their success. Second, until the recent discovery of the SEGA method, RCD methods were not able to converge with non-separable regularizers. Third, RCD methods are naturally variance-reduced in the $\psi \equiv 0$ case as partial derivatives at the optimum are all zero. As a consequence, attempts at creating variance-reduced RCD methods seem to be futile. Lastly, RCD methods are typically analyzed using different techniques. While there are deep links between standard SGD and RCD methods, these are often indirect and rely on duality [191, 30, 62].

6.2 Contributions

As outlined in the previous section, the world of SGD is vast and beautiful. It is formed by many largely disconnected islands populated by elegant and efficient methods, with their own applications, intuitions, and convergence analysis techniques. While some links already exist (e.g., the unification of importance sampling and mini-batching variants under the arbitrary sampling umbrella), there is no comprehensive general theory. It is becoming increasingly difficult for the community to understand the relationships between these variants, both in theory and practice. New variants are yet to be discovered, but it is not clear what tangible principles one should adopt beyond intuition to aid the discovery. This situation is exacerbated by the fact that a number of different assumptions on the stochastic gradient, of various levels of strength, is being used in the literature.

The main contributions of this work include:

- **Unified analysis.** In this work we propose a *unifying theoretical framework* which covers all of the variants of SGD outlined in Section 6.1. As a by-product, we obtain the *first unified analysis* of vanilla and variance-reduced SGD methods. For instance, our analysis covers as special cases vanilla SGD methods from [159] and [60], variance-reduced SGD methods such as SAGA [37], LSVRG [83, 106] and JacSketch [65]. Another by-product is *the unified analysis of SGD methods which include RCD*. For instance, our theory covers the subspace descent method SEGA [77] as a special case.

Lastly, our framework is general enough to capture the phenomenon of *quantization*. For instance, we obtain the DIANA and VR-DIANA methods in special cases.

- **Generalization of existing methods.** An important yet *relatively* minor contribution of our work is that it enables *generalization* of known methods. For instance, some particular methods we consider, such as LSVRG (Algorithm 53) [106], were not analyzed in the proximal ($\psi \neq 0$) case before. To illustrate how this can be done within our framework, we do it here for LSVRG. Further, most of the methods we analyze can be extended to the *arbitrary sampling* paradigm.
- **Sharp rates.** In all known special cases, the rates obtained from our general theorem (Theorem 6.3.4) are the *best known rates* for these methods.
- **New methods.** Our general analysis provides estimates for a possibly infinite array of new and yet-to-be-developed variants of SGD. One only needs to verify that Assumption 6.3.1 holds, and a complexity estimate is readily furnished by Theorem 6.3.4. Selected existing and new methods that fit our framework are summarized in Table 6.1. This list is for illustration only, we believe that future work by us and others will lead to its rapid expansion.
- **Experiments.** We show through extensive experimentation that some of the *new* and *generalized* methods proposed here and analyzed via our framework have some intriguing practical properties when compared against appropriately selected existing methods.

6.3 Main result

We first introduce the key assumption on the stochastic gradients g^k enabling our general analysis (Assumption 6.3.1), then state our assumptions on f (Assumption 6.3.2), and finally state and comment on our unified convergence result (Theorem 6.3.4).

Notation. Consistently with the rest of the thesis, we use the following notation: $\langle x, y \rangle \stackrel{\text{def}}{=} \sum_i x_i y_i$ is the standard Euclidean inner product, and $\|x\| \stackrel{\text{def}}{=} \langle x, x \rangle^{1/2}$ is the induced ℓ_2 norm. For simplicity we assume that (6.1) has a unique minimizer, which we denote x^* . Let $D_f(x, y)$ denote the *Bregman divergence* associated with f : $D_f(x, y) \stackrel{\text{def}}{=} f(x) - f(y) - \langle \nabla f(y), x - y \rangle$. We often write $[n] \stackrel{\text{def}}{=} \{1, 2, \dots, n\}$.

6.3.1 Key assumption

Our first assumption is of key importance. It is mainly an assumption on the sequence of stochastic gradients $\{g^k\}$ generated by an arbitrary randomized algorithm. Besides unbiasedness (see (6.7)), we require two recursions to hold for the iterates x^k and the stochastic gradients g^k of a randomized method. We allow for flexibility by casting these inequalities in a parametric manner.

Assumption 6.3.1. Let $\{x^k\}$ be the random iterates produced by proximal SGD (Algorithm in Eq (6.5)). We first assume that the stochastic gradients g^k are unbiased

$$\mathbb{E}[g^k | x^k] = \nabla f(x^k), \quad (6.7)$$

for all $k \geq 0$. Further, we assume that there exist non-negative constants A, B, C, D_1, D_2, ρ and a (possibly) random sequence $\{\sigma_k^2\}_{k \geq 0}$ such that the following two relations hold³

$$\mathbb{E}[\|g^k - \nabla f(x^*)\|^2 | x^k, \sigma_k^2] \leq 2AD_f(x^k, x^*) + B\sigma_k^2 + D_1, \quad (6.8)$$

$$\mathbb{E}[\sigma_{k+1}^2 | x^k, \sigma_k^2] \leq (1 - \rho)\sigma_k^2 + 2CD_f(x^k, x^*) + D_2, \quad (6.9)$$

The expectation above is with respect to the randomness of the algorithm.

The unbiasedness assumption (6.7) is standard. The key innovation we bring is inequality (6.8) coupled with (6.9). We argue, and justify this statement by furnishing many examples in Section 6.4, that these inequalities capture the essence of a wide array of existing and some new SGD methods, including vanilla, variance reduced, arbitrary sampling, quantized and coordinate descent variants. Note that in the case when $\nabla f(x^*) = 0$ (e.g., when $\psi \equiv 0$), the inequalities in Assumption 6.3.1 reduce to

$$\mathbb{E}[\|g^k\|^2 | x^k, \sigma_k^2] \leq 2A(f(x^k) - f(x^*)) + B\sigma_k^2 + D_1, \quad (6.10)$$

$$\mathbb{E}[\sigma_{k+1}^2 | x^k, \sigma_k^2] \leq (1 - \rho)\sigma_k^2 + 2C(f(x^k) - f(x^*)) + D_2. \quad (6.11)$$

Similar inequalities can be found in the analysis of stochastic first-order methods. However, this is the first time that such inequalities are generalized, equipped with parameters, and elevated to the status of an assumption that can be used on its own, independently from any other details defining the underlying method that generated them.

To give a further intuition about inequalities (6.8) and (6.9), we shall note that sequence σ_k usually represents the portion of noise that can gradually decrease over the course of optimization while constants D_1, D_2 represent a static noise. On the other hand, constants A, C are usually related to some measure of smoothness of the objective. For instance, the parameters for (deterministic) gradient descent can be chosen as $A = L, B = C = D_1 = D_2 = \sigma_k^2 = \rho = 0$. For an overview of parameter choices for specific instances of (6.5), see Table 6.2. Note also that the choice of parameters of (6.8) and (6.9) is not unique, however this has no impact on convergence rates we provide.

6.3.2 Main theorem

For simplicity, we shall assume throughout that f is μ -strongly quasi-convex, which is a generalization of μ -strong convexity. We leave an analysis under different assumptions on f to future work.

³For convex and L -smooth f , one can show that $\|\nabla f(x) - \nabla f(y)\|^2 \leq 2LD_f(x, y)$. Hence, D_f can be used as a measure of proximity for the gradients.

Assumption 6.3.2 (μ -strong quasi-convexity). *There exists $\mu > 0$ such that $f : \mathbb{R}^d \rightarrow \mathbb{R}$ is μ -strongly quasi-convex. That is, the following inequality holds for all $x \in \mathbb{R}^d$:*

$$f(x^*) \geq f(x) + \langle \nabla f(x), x^* - x \rangle + \frac{\mu}{2} \|x^* - x\|^2. \quad (6.12)$$

We are now ready to present the key lemma of this chapter which states per iteration recurrence to analyze (6.5). Due to space limitations, we present the proof in Section 6.3 of the Appendix.

Lemma 6.3.3. *Let Assumptions 6.3.1 and 6.3.2 be satisfied. Then the following inequality holds for all $k \geq 0$:*

$$\begin{aligned} \mathbb{E} \left[\|x^{k+1} - x^*\|^2 \right] + M\alpha^2 \mathbb{E} [\sigma_{k+1}^2] + 2\alpha(1 - \alpha(A + CM)) \mathbb{E} [D_f(x^k, x^*)] \\ \leq (1 - \alpha\mu) \mathbb{E} \left[\|x^k - x^*\|^2 \right] + (1 - \rho) M\alpha^2 \mathbb{E} [\sigma_k^2] + B\alpha^2 \mathbb{E} [\sigma_k^2] + (D_1 + MD_2)\alpha^2. \end{aligned}$$

Using recursively Lemma 6.3.3, we obtain the convergence rate of proximal SGD, which we state as Theorem 6.3.4.

Theorem 6.3.4. *Let Assumptions 6.3.1 and 6.3.2 be satisfied. Choose constant M such that $M > \frac{B}{\rho}$. Choose a stepsize satisfying*

$$0 < \alpha \leq \min \left\{ \frac{1}{\mu}, \frac{1}{A + CM} \right\}. \quad (6.13)$$

Then the iterates $\{x^k\}_{k \geq 0}$ of proximal SGD (Algorithm (6.5)) satisfy

$$\mathbb{E} [V^k] \leq \max \left\{ (1 - \alpha\mu)^k, \left(1 + \frac{B}{M} - \rho \right)^k \right\} V^0 + \frac{(D_1 + MD_2)\alpha^2}{\min \left\{ \alpha\mu, \rho - \frac{B}{M} \right\}}, \quad (6.14)$$

where the Lyapunov function V^k is defined by $V^k \stackrel{\text{def}}{=} \|x^k - x^\|^2 + M\alpha^2 \sigma_k^2$.*

This theorem establishes a linear rate for a wide range of proximal SGD methods up to a certain oscillation radius, controlled by the additive term in (6.14), and namely, by parameters D_1 and D_2 . As we shall see in Section F.1 (refer to Table 6.2), the main difference between the vanilla and variance-reduced SGD methods is that while the former satisfy inequality (6.9) with $D_1 > 0$ or $D_2 > 0$, which in view of (6.14) prevents them from reaching the optimum x^* (using a fixed stepsize), the latter methods satisfy inequality (6.9) with $D_1 = D_2 = 0$, which in view of (6.14) enables them to reach the optimum.

6.4 The classic, the recent and the brand new

In this section we deliver on the promise from the introduction and show how many existing and some new variants of SGD fit our general framework (see Table 6.1).

Problem	Method	Alg	Citation	VR	AS	Quant	RCD	Sec	Cor
(6.1)+(6.2)	SGD	44	[159]	✗	✗	✗	✗	F.1.1	F.1.2
(6.1)+(6.3)	SGD-SR	45	[60]	✗	✓	✗	✗	F.1.2	F.1.5
(6.1)+(6.3)	SGD-MB	46	NEW	✗	✗	✗	✗	F.1.3	F.1.9
(6.1)+(6.3)	SGD-star	47	NEW	✓	✓	✗	✗	F.1.4	F.1.11
(6.1)+(6.3)	SAGA	48	[37]	✓	✗	✗	✗	F.1.5	F.1.13
(6.1)+(6.3)	N-SAGA	49	NEW	✗	✗	✗	✗	F.1.6	F.1.15
(6.1)	SEGA	50	[77]	✓	✗	✗	✓	F.1.7	F.1.17
(6.1)	N-SEGA	51	NEW	✗	✗	✗	✓	F.1.8	F.1.19
(6.1)+(6.3)	SVRG ^a	52	[88]	✓	✗	✗	✗	F.1.9	F.1.21
(6.1)+(6.3)	LSVRG	53	[83]	✓	✗	✗	✗	F.1.10	F.1.23
(6.1)+(6.3)	DIANA	54	[136]	✗	✗	✓	✗	F.1.11	F.1.26
(6.1)+(6.3)	DIANA ^b	55	[136]	✓	✗	✓	✗	F.1.11	F.1.27
(6.1)+(6.3)	Q-SGD-SR	56	NEW	✗	✓	✓	✗	F.1.12	F.1.29
(6.1)+(6.3)+(6.4)	VR-DIANA	57	[85]	✓	✗	✓	✗	F.1.13	F.1.32
(6.1)+(6.3)	JacSketch	58	[65]	✓	✓ ✗	✗	✗	F.1.14	F.1.34

Table 6.1: List of specific existing (in some cases generalized) and new methods which fit our general analysis framework. VR = variance reduced method, AS = arbitrary sampling, Quant = supports gradient quantization, RCD = randomized coordinate descent type method. ^a Special case of SVRG with 1 outer loop only; ^b Special case of DIANA with 1 node and quantization of exact gradient.

An overview. As claimed, our framework is powerful enough to include vanilla methods (✗ in the “VR” column) as well as variance-reduced methods (✓ in the “VR” column), methods which generalize to arbitrary sampling (✓ in the “AS” column), methods supporting gradient quantization (✓ in the “Quant” column) and finally, also RCD type methods (✓ in the “RCD” column).

For existing methods we provide a citation; new methods developed in this chapter are marked accordingly. Due to space restrictions, all algorithms are described (in detail) in the Appendix; we provide a link to the appropriate section for easy navigation. While these details are important, the main message of this chapter, i.e., the generality of our approach, is captured by Table 6.1. The “Result” column of Table 6.1 points to a corollary of Theorem 6.3.4; these corollaries state in detail the convergence statements for the various methods. In all cases where known methods are recovered, these corollaries of Theorem 6.3.4 recover the best known rates.

Parameters. From the point of view of Assumption 6.3.1, the methods listed in Table 6.1 exhibit certain patterns. To shed some light on this, in Table 6.2 we summarize the values of these parameters.

Note, for example, that for all methods the parameter A is non-zero. Typically, this a multiple of an appropriately defined smoothness parameter (e.g., L is the Lipschitz constant of the gradient of f , \mathcal{L} and \mathcal{L}_1 in SGD-SR⁴, SGD-star and JacSketch are

⁴SGD-SR is first SGD method analyzed in the *arbitrary sampling* paradigm. It was developed using the *stochastic reformulation* approach (whence the “SR”) pioneered in [176] in a numerical linear algebra setting, and later extended to develop the JacSketch variance-reduction technique for finite-sum

Table 6.2: The parameters for which the methods from Table 6.1 (special cases of (6.5)) satisfy Assumption 6.3.1. The meaning of the expressions appearing in the table, as well as their justification is defined in detail in the Appendix (Section F.1).

Method	A	B	ρ	C	D_1	D_2
SGD	$2L$	0	1	0	$2\sigma^2$	0
SGD-SR	$2\mathcal{L}$	0	1	0	$2\sigma^2$	0
SGD-MB	$\frac{A'+L(\tau-1)}{\tau}$	0	1	0	$\frac{D'}{\tau}$	0
SGD-star	$2\mathcal{L}$	0	1	0	0	0
SAGA	$2L$	2	$1/n$	L/n	0	0
N-SAGA	$2L$	2	$1/n$	L/n	$2\sigma^2$	$\frac{\sigma^2}{n}$
SEGA	$2dL$	$2d$	$1/d$	L/d	0	0
N-SEGA	$2dL$	$2d$	$1/d$	L/d	$2d\sigma^2$	$\frac{\sigma^2}{d}$
SVRG ^a	$2L$	2	0	0	0	0
LSVRG	$2L$	2	p	Lp	0	0
DIANA	$(1 + \frac{2\omega}{n}) L$	$\frac{2\omega}{n}$	γ	$L\gamma$	$\frac{(1+\omega)\sigma^2}{n}$	$\gamma\sigma^2$
DIANA ^b	$(1 + 2\omega) L$	2ω	γ	$L\gamma$	0	0
Q-SGD-SR	$2(1 + \omega)\mathcal{L}$	0	1	0	$2(1 + \omega)\sigma^2$	0
VR-DIANA	$(1 + \frac{4\omega+2}{n}) L$	$\frac{2(\omega+1)}{n}$	γ	$(\frac{1}{m} + 4\gamma) L$	0	0
JacSketch	$2\mathcal{L}_1$	$\frac{2\lambda_{\max}^n}{n}$	λ_{\min}	$\frac{\mathcal{L}_2}{n}$	0	0

expected smoothness parameters). In the three variants of the DIANA method, ω captures the variance of the quantization operator Q . That is, one assumes that $\mathbb{E}[Q(x)] = x$ and $\mathbb{E}[\|Q(x) - x\|^2] \leq \omega \|x\|^2$ for all $x \in \mathbb{R}^d$. In view of (6.13), large A means a smaller stepsize, which slows down the rate. Likewise, the variance ω also affects the parameter B , which in view of (6.14) also has an adverse effect on the rate. Further, as predicted by Theorem 6.3.4, whenever either $D_1 > 0$ or $D_2 > 0$, the corresponding method converges to an oscillation region only. These methods are not variance-reduced. All symbols used in Table 6.2 are defined in the appendix, in the same place where the methods are described and analyzed.

Five new methods. To illustrate the usefulness of our general framework, we develop 5 new variants of SGD never explicitly considered in the literature before (see Table 6.1). Here we briefly motivate them; details can be found in the Appendix.

- **SGD-MB** (Algorithm 46). This method is specifically designed for functions of the finite-sum structure (6.4). As we show through experiments, this is a powerful mini-batch SGD method, with mini-batches formed with replacement as follows: in each iteration, we repeatedly (τ times) and independently pick $i \in [n]$ with probability $p_i > 0$. Stochastic gradient g^k is then formed by averaging the stochastic gradients $\nabla f_i(x^k)$ for all selected indices i (including each i as many times as this index

optimization [65].

was selected). This allows for a more practical importance mini-batch sampling implementation than what was until now possible (see Remark 29 in the Appendix for more details and experiment in Figure 6.1).

- **SGD-star (Algorithm 47).** This new method forms a bridge between vanilla and variance-reduced SGD methods. While not practical, it sheds light on the role of variance reduction. Again, we consider functions of the finite-sum form (6.4). This method answers the following question: assuming that the gradients $\nabla f_i(x^*)$, $i \in [n]$ are *known*, can they be used to design a more powerful SGD variant? The answer is yes, and SGD-star is the method. In its most basic form, SGD-star constructs the stochastic gradient via $g^k = \nabla f_i(x^k) - \nabla f_i(x^*) + \nabla f(x^*)$, where $i \in [n]$ is chosen uniformly at random. Inferring from Table 6.2, where $D_1 = D_2 = 0$, this method converges to x^* , and not merely to some oscillation region. Variance-reduced methods essentially work by iteratively constructing increasingly more accurate *estimates* of $\nabla f_i(x^*)$. Typically, the term σ_k^2 in the Lyapunov function of variance reduced methods will contain a term of the form $\sum_i \|h_i^k - \nabla f_i(x^*)\|^2$, with h_i^k being the estimators maintained by the method. Remarkably, SGD-star was never explicitly considered in the literature before.
- **N-SAGA (Algorithm 49).** This is a novel variant of SAGA [37], one in which one does not have access to the gradients of f_i , but instead only has access to *noisy* stochastic estimators thereof (with noise σ^2). Like SAGA, N-SAGA is able to reduce the variance inherent in the finite sum structure (6.4) of the problem. However, it necessarily pays the price of noisy estimates of ∇f_i , and hence, just like vanilla SGD methods, is ultimately unable to converge to x^* . The oscillation region is governed by the noise level σ^2 (refer to D_1 and D_2 in Table 6.2). This method will be of practical importance for problems where each f_i is of the form (6.2), i.e., for problems of the “average of expectations” structure. Batch versions of N-SAGA would be well suited for distributed optimization, where each f_i is owned by a different worker, as in such a case one wants the workers to work in parallel.
- **N-SEGA (Algorithm 51).** This is a *noisy* extension of the RCD-type method SEGA, in complete analogy with the relationship between SAGA and N-SAGA. Here we assume that we only have noisy estimates of partial derivatives (with noise σ^2). This situation is common in derivative-free optimization, where such a noisy estimate can be obtained by taking (a random) finite difference approximation [153]. Unlike SEGA, N-SEGA only converges to an oscillation region the size of which is governed by σ^2 .
- **Q-SGD-SR (Algorithm 56).** This is a quantized version of SGD-SR, which is the first SGD method analyzed in the arbitrary sampling paradigm. As such, Q-SGD-SR is a vast generalization of the celebrated QSGD method [2].

6.5 Experiments

In this section we numerically verify the claims from the chapter. We perform three different experiments: we verify the usefulness of SGD-MB alongside with testing both

SGD-star and N-SEGA.

6.5.1 SGD-MB: remaining experiments and exact problem setup.

In Section F.1.3, we describe in detail the SGD-MB method already outlined before. The main advantage of SGD-MB is that the sampling procedure it employs can be implemented in just $\mathcal{O}(\tau \log n)$ time. In contrast, even the simplest without-replacement sampling which selects each function into the minibatch with a prescribed probability independently (we will refer to it as independent SGD) requires n calls of a uniform random generator. We demonstrate numerically that SGD-MB has essentially identical iteration complexity to independent SGD in practice. We consider logistic regression with Tikhonov regularization of order λ :

$$\frac{1}{n} \sum_{i=1}^n \log(1 + \exp(a_i^\top x \cdot b_i)) + \frac{\lambda}{2} \|x\|^2, \quad (6.15)$$

where $a_i \in \mathbb{R}^n$, $b_i \in \{-1, 1\}$ is i th data-label pair is a vector of labels and $\lambda \geq 0$ is the regularization parameter. The data and labels were obtained from LibSVM datasets a1a, a9a, w1a, w8a, gisette, madelon, phishing and mushrooms. Further, the data were rescaled by a random variable cu_i^2 where u_i is random integer from $1, 2, \dots, 1000$ and c is such that the mean norm of a_i is 1. Note that we have now an infinite array of possibilities on how to write (6.15) as (6.3). For simplicity, distribute l_2 term evenly among the finite sum.

For a fixed expected sampling size τ , we consider two options for the probability of sampling the i th function:

- (i) $\frac{\tau}{n}$, or
- (ii) $\frac{\|a_i\|^2 + \lambda}{\delta + \|a_i\|^2 + \lambda}$, where δ is such that⁵ $\sum_{i=1}^n \frac{\|a_i\|^2 + \lambda}{\delta + \|a_i\|^2 + \lambda} = 1$.

The results can be found in Figure 6.1, where we also report the choice of stepsize α and the choice of τ in the legend and title of the plot, respectively.

Indeed, iteration complexity of SGD-MB and independent SGD is almost identical. Since the cost of each iteration of SGD-MB is cheaper⁶, we conclude superiority of SGD-MB to independent SGD.

6.5.2 Experiments on SGD-star

In this section, we study SGD-star and numerically verify claims from Section F.1.4. In particular, Corollary F.1.11 shows that SGD-star enjoys linear convergence rate which is constant times better to the rate of SAGA (given that problem condition number is high enough). We compare 3 methods – SGD-star, SGD and SAGA. We consider simple

⁵An RCD version of this sampling was proposed in [78]; it was shown to be superior to uniform sampling both in theory and practice.

⁶The relative difference between iteration costs of SGD-MB and independent SGD can be arbitrary, especially for the case when cost of evaluating $\nabla f_i(x)$ is cheap, n is huge and $n \gg \tau$. In such case, cost of one iteration of SGD-MB is $\tau \text{Cost}(\nabla f_i) + \tau \log(n)$ while the cost of one iteration of independent SGD is $\tau \text{Cost}(\nabla f_i) + n$.

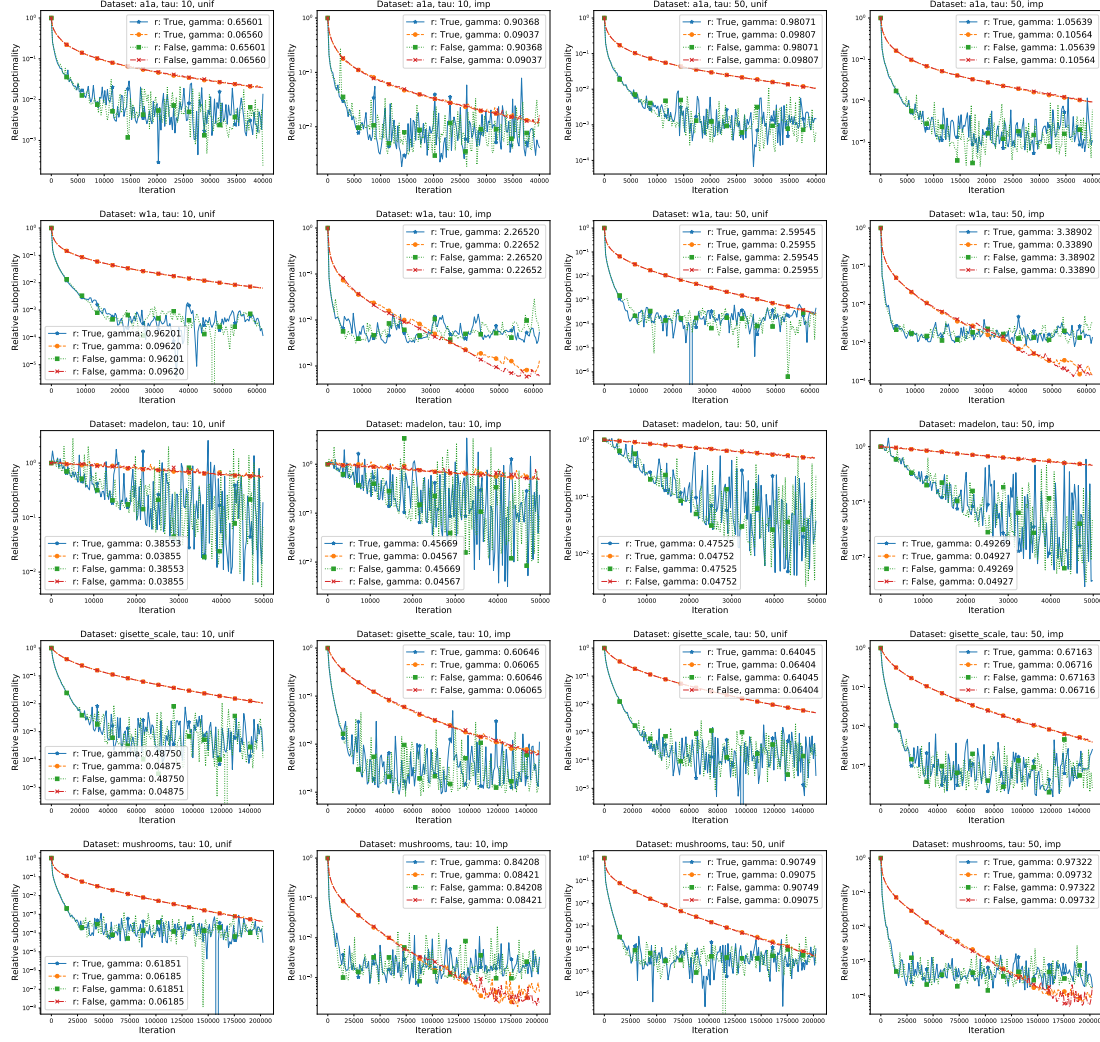


Figure 6.1: SGD-MB and independent SGD applied on LIBSVM [23] datasets with regularization parameter $\lambda = 10^{-5}$. Axis y stands for relative suboptimality, i.e. $\frac{f(x^k) - f(x^*)}{f(x^k) - f(x^0)}$. Title label “unif” corresponds to probabilities chosen by (i) while label “imp” corresponds to probabilities chosen by (ii). Lastly, legend label “r” corresponds to “replacement” with value “True” for SGD-MB and value “False” for independent SGD.

and well-understood least squares problem $\min_x \frac{1}{2} \|\mathbf{A}x - b\|^2$ where elements of \mathbf{A} , b were generated (independently) from standard normal distribution. Further, rows of \mathbf{A} were normalized so that $\|\mathbf{A}_i\| = 1$. Thus, denoting $f_i(x) = \frac{1}{2}(\mathbf{A}_i^\top x - b_i)^2$, f_i is 1-smooth. For simplicity, we consider SGD-star with uniform serial sampling, i.e. $\mathcal{L} = 1$.

Next, for both SGD-star and SGD we use stepsize $\alpha = \frac{1}{2}$ (theory supported stepsize for SGD-star), while for SAGA we set $\alpha = \frac{1}{5}$ (almost theory supported stepsize). Figure 6.2 shows the results.

Note that, as theory predicts, SGD-star is always faster to SAGA, although only constant times. Further, in the cases where $d \geq n$, performance of SGD seems identical to the performance of SGD-shift. This is due to a simple reason: if $d \geq n$, we must have

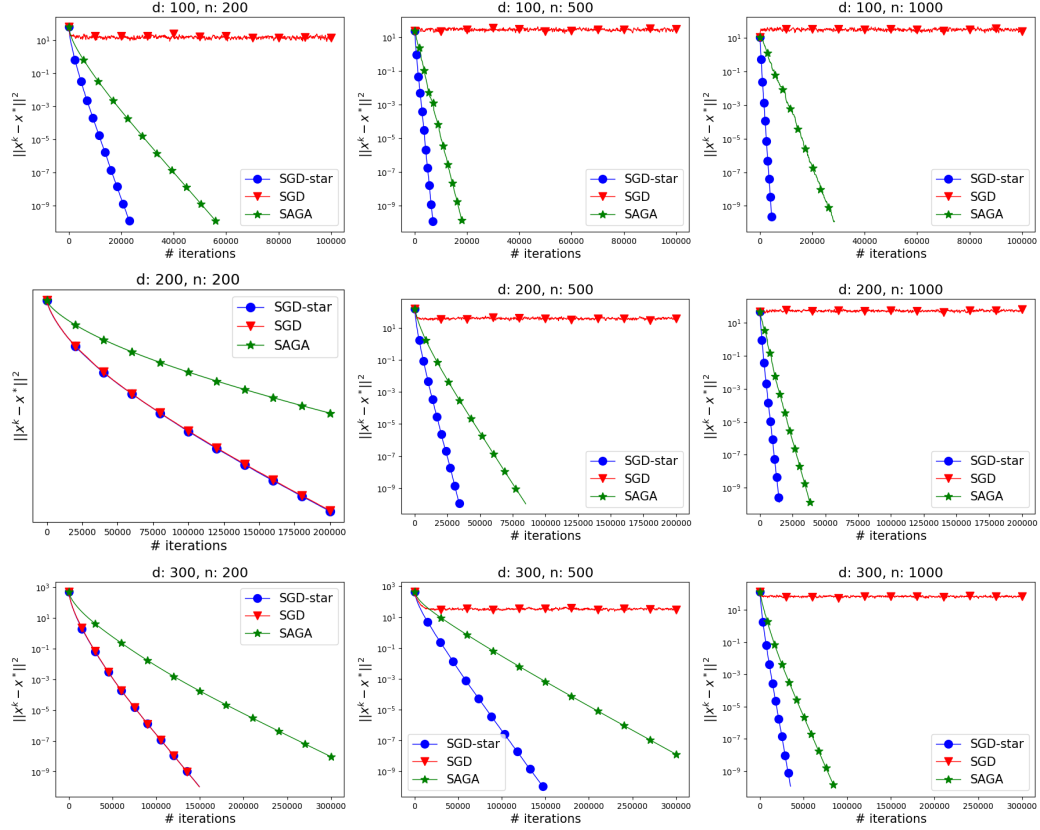


Figure 6.2: Comparison of SGD-star, SGD and SAGA on least squares problem.

$\nabla f_i(x^*) = 0$ for all i , and thus SGD and SGD-shift are in fact identical algorithms.

6.5.3 Experiments on N-SEGA

In this experiment we study the effect of noise on N-SEGA. We consider unit ball constrained least squares problem: $\min_{\|x\| \leq 1} f(x)$ where $f(x) = \|\mathbf{A}x - b\|^2$. and we suppose that there is an oracle providing us with noised partial derivative $g_i(x, \zeta) = \nabla_i f(x) + \zeta$, where $\zeta \sim N(0, \sigma^2)$. For each problem instance (i.e. pair \mathbf{A}, b), we compare performance of N-SEGA under various noise magnitudes σ^2 .

The specific problem instances are presented in Table 6.3. Figure 6.3 shows the results.

Type	\mathbf{A}	b
1	$\mathbf{A}_{ij} \sim N(0, 1)$ (independently)	vector of ones
2	Same as 1, but scaled so that $\lambda_{\max}(\mathbf{A}^\top \mathbf{A}) = 1$	vector of ones
3	$\mathbf{A}_{ij} = \varrho_{ij} \varpi_j \forall i, j : \varrho_{ij}, \varpi_j \sim N(0, 1)$ (independently)	vector of ones
4	Same as 3, but scaled so that $\lambda_{\max}(\mathbf{A}^\top \mathbf{A}) = 1$	vector of ones

Table 6.3: Four types of least squares.

We shall mention that this experiment serves to support and give a better intuition

about the results from Section F.1.8 and is by no means practical. The results show, as predicted by theory, linear convergence to a specific neighborhood of the objective. The effect of the noise varies, however, as a general rule, the larger strong convexity μ is (i.e. problems 1,3 where scaling was not applied), the smaller the effect of noise is.

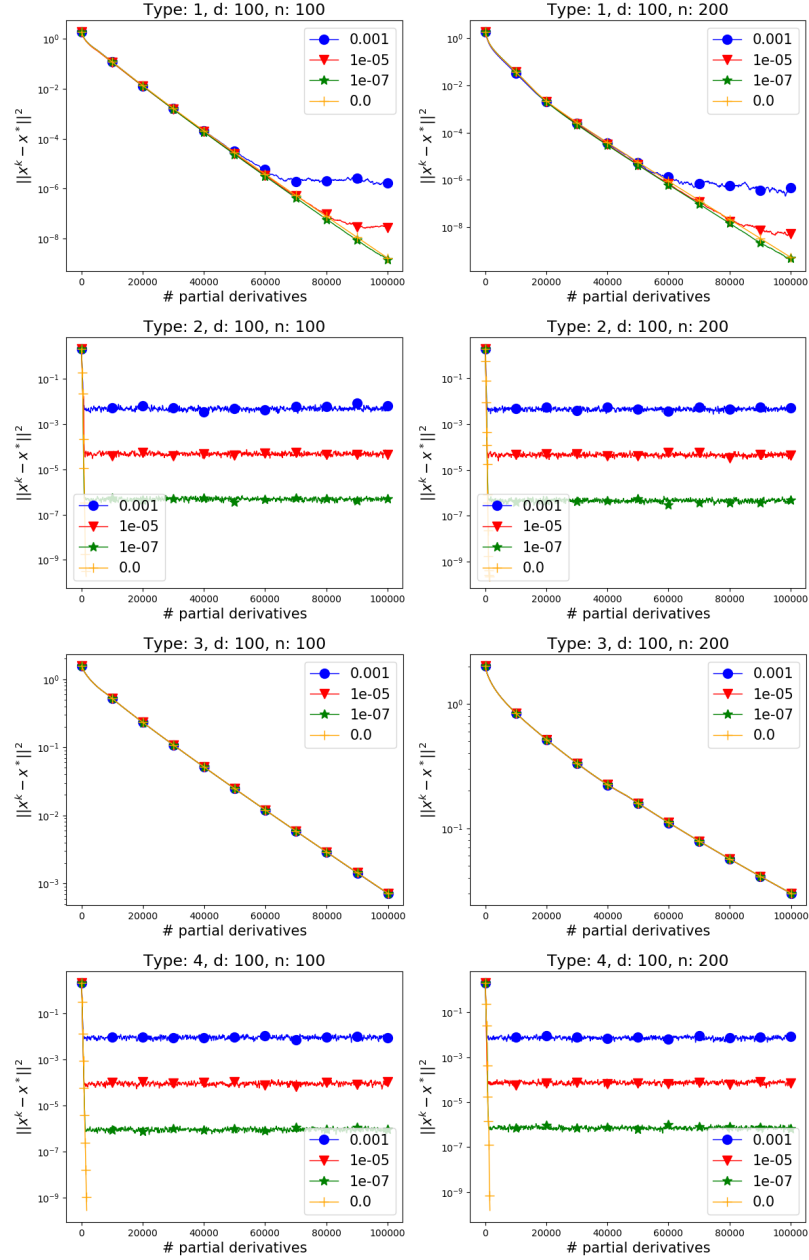


Figure 6.3: N-SEGA applied on constrained least squares problem with noised partial derivative oracle. Legend labels stand for the magnitude σ^2 of the oracle noise.

6.6 Conclusion

In this chapter we have introduced a general scheme to analyze stochastic gradient algorithms with many different applications. Although the presented approach is rather general, we still see several possible directions for future extensions, including:

- We believe our results can be extended to *weakly convex* functions. However, producing a comparable result in the *nonconvex* case remains a major open problem.
- It would be further interesting to unify our theory with *biased* gradient estimators. If this was possible, one could recover methods as SAG [182] in special cases, or obtain rates for the zero-order optimization. We have some preliminary results in this direction already.
- Although our theory allows for non-uniform stochasticity, it does not recover the best known rates for RCD type methods with *importance sampling*. It would be thus interesting to provide a more refined analysis capable of capturing importance sampling phenomena more accurately.
- An extension of Assumption 6.3.1 to *iteration dependent* parameters A, B, C, D_1, D_2, ρ would enable an array of new methods, such as SGD with decreasing stepsizes. Such an extension is rather very straightforward.
- It would be interesting to provide a unified analysis of stochastic methods with *acceleration* and *momentum*. In fact, [110] provide (separately) a unification of some methods with and without variance reduction. The next chapter provides another step towards the unified accelerated analysis – we introduce an accelerated SVRCD algorithm.

Chapter 7

Variance Reduced Coordinate Descent with Acceleration: New Method With a Surprising Application to Finite-Sum Problems

In this chapter, we aim to solve the regularized optimization problem

$$\min_{x \in \mathbb{R}^d} \{F(x) = f(x) + \psi(x)\}, \quad (7.1)$$

where function f is convex and differentiable (not necessarily of a finite-sum structure), while the regularizer ψ is convex and non-smooth. Furthermore, we assume that the dimensionality d is large.

The most standard approach to deal with the huge d is to decompose the space, i.e., use coordinate descent, or, more generally, subspace descent methods [152, 215, 109]. Those methods are especially popular as they achieve a linear convergence rate on strongly convex problems while enjoying a relatively cheap cost of performing each iteration.

However, coordinate descent methods are only feasible if the regularizer ψ is separable [173]. In contrast, if ψ is not separable, the corresponding stochastic gradient estimator has an inherent (non-zero) variance at the optimum, and thus the linear convergence rate is not achievable.

This phenomenon is, to some extent, similar when applying Stochastic Gradient Descent (SGD) [179, 148] on finite sum objective – the corresponding stochastic gradient estimator has a (non-zero) variance at the optimum, which prevents SGD from converging linearly. Recently, the issue of sublinear convergence of SGD has been resolved using the idea of control variates [82], resulting in famous variance reduced methods such as SVRG [88] and SAGA [37].

Motivated by the massive success of variance reduced methods for finite sums, control variates have been proposed to “fix” coordinate descent methods to minimize problem (7.1) with non-separable ψ . To best of our knowledge, there are two such algorithms in the literature—SEGA (proposed in Chapter 3) and SVRCD (proposed in Chapter 5)—which we now quickly describe.¹

Let x^k be the current iterate of SEGA (or SVRCD) and suppose that the oracle reveals $\nabla_i f(x^k)$ (for i chosen uniformly at random). The simplest unbiased gradient estimator of $\nabla f(x^k)$ can be constructed as $\tilde{g}^k = d \nabla_i f(x^k) e_i$ (where $e_i \in \mathbb{R}^d$ is the i th standard basis vector). The idea behind these methods is to enrich \tilde{g}^k using a control variate $h^k \in \mathbb{R}^d$, resulting in a new (still unbiased) gradient estimator g^k :

$$g^k = d \nabla_i f(x^k) e_i - d h_i^k e_i + h^k.$$

¹VRSSD [109] is yet another stochastic subspace descent algorithm aided by control variates; however, it was proposed to minimize f only (i.e., considers $\psi \equiv 0$).

How to choose the sequence of control variates $\{h^k\}$? Intuitively, we wish for both sequences $\{h^k\}$ and $\{\nabla f(x^k)\}$ to have an identical limit point. In such case, we have $\lim_{k \rightarrow \infty} \text{Var}(g^k) = 0$, and thus one shall expect faster convergence. There is no unique way of setting $\{h^k\}$ to have the mentioned property satisfied – this is where SEGA and SVRCD differ. See Algorithm 15 for details.

Algorithm 15 SEGA and SVRCD

Require: Stepsize $\alpha > 0$, starting point $x^0 \in \mathbb{R}^d$, probability vector p : $p_i \stackrel{\text{def}}{=} \mathbb{P}(i \in S)$
Set $h^0 = 0 \in \mathbb{R}^d$
for $k = 0, 1, 2, \dots$ **do**
 Sample random $S \subseteq \{1, 2, \dots, d\}$
 $g^k = \sum_{i \in S} \frac{1}{p_i} (\nabla_i f(x^k) - h_i^k) e_i + h^k$
 $x^{k+1} = \text{prox}_{\alpha\psi}(x^k - \alpha g^k)$
 $h^{k+1} = \begin{cases} h^k + \sum_{i \in S} (\nabla_i f(x^k) - h_i^k) e_i & \text{for SEGA} \\ \begin{cases} \nabla f(x^k) & \text{with probability } \rho \\ h^k & \text{with probability } 1 - \rho \end{cases} & \text{for SVRCD} \end{cases}$
end for

In this work, we continue the above research along the lines of variance reduced coordinate descent algorithms, with surprising consequences.

7.1 Contributions

Here we list the main contributions of this chapter.

- **Exploiting prox in SEGA/SVRCD.** Assume that the regularizer ψ includes an indicator function of some affine subspace of \mathbb{R}^d . We show that both SEGA and SVRCD might exploit this fact, resulting in a faster convergence rate. As a byproduct, we establish the same result in the more general GJS framework from Chapter 5 (presented in the appendix).
- **Accelerated SVRCD.** We propose an accelerated version of SVRCD - ASVRCD. ASVRCD is the first accelerated variance reduced coordinate descent to minimize objectives with non-separable, proximable regularizer.²
- **SEGA/SVRCD/ASVRCD generalizes SAGA/LSVRG/L-Katyusha.** We show a surprising link between SEGA and SAGA. In particular, SAGA is a special case of SEGA; and the new rate we obtain for SEGA recovers the tight complexity of SAGA [165, 52]. Similarly, we recover loopless SVRG (LSVRG) [83, 106] along with its best-known rate [79, 164] using a result for SVRCD. Lastly, as a particular case of ASVRCD,

²We shall note that an accelerated version of SEGA was already proposed in [77] for $\psi = 0$ – this was rather an impractical result demonstrating that SEGA can match state-of-the-art convergence rate of accelerated coordinate descent from [7, 158, 78]. In contrast, our results cover any convex ψ .

we recover an algorithm which is marginally preferable to loopless Katyusha (L-Katyusha) [164]: while we recover their iteration complexity result, our proof is more straightforward, and at the same time, the stepsize for the proximal operator is smaller.³

7.2 Preliminaries

As mentioned in Section 7.1, the new results we provide are particularly interesting if the regularizer ψ contains an indicator function of some affine subspace of \mathbb{R}^d .

Assumption 7.2.1. Assume that \mathbf{W} is a projection matrix such that

$$\psi(x) = \begin{cases} \psi'(x) & \text{if } x \in \{x^0 + \text{Range}(\mathbf{W})\} \\ \infty & \text{if } x \notin \{x^0 + \text{Range}(\mathbf{W})\} \end{cases} \quad (7.2)$$

for some convex function $\psi'(x)$. Furthermore, suppose that the proximal operator of ψ is cheap to compute.

Remark 6. If ψ is convex, there is always some \mathbf{W} such that (7.2) holds as one might choose $\mathbf{W} = \mathbf{I}$.

Next, we require smoothness of the objective, as well as the strong convexity over the affine subspace given by $\text{Range}(\mathbf{W})$.

Assumption 7.2.2. Function f is \mathbf{M} -smooth, i.e., for all $x, y \in \mathbb{R}^d$:⁴

$$f(x) \leq f(y) + \langle \nabla f(y), x - y \rangle + \frac{1}{2} \|x - y\|_{\mathbf{M}}^2.$$

Function f is μ -strongly convex over $\{x^0 + \text{Range}(\mathbf{W})\}$, i.e., for all $x, y \in \{x^0 + \text{Range}(\mathbf{W})\}$:

$$f(x) \geq f(y) + \langle \nabla f(y), x - y \rangle + \frac{\mu}{2} \|x - y\|^2. \quad (7.3)$$

Remark 7. Smoothness with respect to matrix \mathbf{M} arises naturally in various applications. For example, if $f(x) = f'(\mathbf{A}x)$, where f' is L' -smooth (for scalar $L' > 0$), we can derive that f is $\mathbf{M} = L' \mathbf{A}^\top \mathbf{A}$ -smooth.

In order to stress the distinction between the finite sum setup and the setup from the rest of the chapter, we are denoting the finite-sum variables that differ from the non-finite sum case in **red**. We thus, *recommend printing this chapter in color*.

7.3 Better rates for SEGA and SVRCD

In this section, we show that a specific structure of nonsmooth function ψ might lead to faster convergence of SEGA and SVRCD.

³This is preferable especially if the proximal operator has to be estimated numerically.

⁴We define $\|x\|^2 \stackrel{\text{def}}{=} \langle x, x \rangle$ and $\|x\|_{\mathbf{M}}^2 \stackrel{\text{def}}{=} \langle \mathbf{M}x, x \rangle$.

The next lemma is a direct consequence of Assumption 7.2.1 – it shows that proximal operator of ψ is contractive under \mathbf{W} -norm.

Lemma 7.3.1. *Let $\{x^k\}_{k \geq 0}$ be a sequence of iterates of Algorithm 15 and let x^* be optimal solution of (7.1). Then*

$$x^k \in \{x^0 + \mathbf{Range}(\mathbf{W})\}, \quad x^* \in \{x^0 + \mathbf{Range}(\mathbf{W})\}. \quad (7.4)$$

for all k . Furthermore, for any $x, y \in \mathbb{R}^d$ and $\alpha > 0$ we have

$$\|\text{prox}_{\alpha\psi}(x) - \text{prox}_{\alpha\psi}(y)\|^2 \leq \|x - y\|_{\mathbf{W}}^2. \quad (7.5)$$

Next, we state the convergence rate of both SEGA and SVRCD under Assumption 7.2.1 as Theorem 7.3.2. We also generalize the main theorem from Chapter 5 (fairly general algorithm which covers SAGA, SVRG, SEGA, SVRCD, and more as a special case; see Section G.4 of the appendix); from which the convergence rate of SEGA/SVRCD follows as a special case.

Theorem 7.3.2. *Let Assumptions 7.2.1, 7.2.2 hold and denote $p_i \stackrel{\text{def}}{=} \mathbb{P}(i \in S)$. Consider vector $v = \sum_{i=1}^d e_i v_i, v_i \geq 0$ such that*

$$\mathbf{M}^{\frac{1}{2}} \mathbb{E} \left[\sum_{i \in S} \frac{1}{p_i} e_i e_i^\top \mathbf{W} \sum_{i \in S} \frac{1}{p_i} e_i e_i^\top \right] \mathbf{M}^{\frac{1}{2}} \preceq \mathbf{Diag}(p^{-1} \circ v), \quad (7.6)$$

where $\mathbf{Diag}(\cdot)$ is a diagonal operator.⁵ Then, iteration complexity of SEGA with $\alpha = \min_i \frac{p_i}{4v_i + \mu}$ is $\max_i \left(\frac{4v_i + \mu}{p_i \mu} \right) \log \frac{1}{\epsilon}$. At the same time, iteration complexity of SVRCD with $\alpha = \min_i \frac{1}{4v_i p_i^{-1} + \mu \rho^{-1}}$ is $\left(\frac{4 \max_i (v_i p_i^{-1}) + \mu \rho^{-1}}{\mu} \right) \log \frac{1}{\epsilon}$.

Let us look closer to convergence rate of SVRCD from Theorem 7.3.2. The optimal vector v is a solution to the following optimization problem

$$\min_{v \in \mathbb{R}^d} \left(\frac{4 \max_i \{v_i p_i^{-1}\} + \mu \rho^{-1}}{\mu} \right) \log \frac{1}{\epsilon} \quad \text{such that} \quad (7.6) \text{ holds.}$$

Clearly, there exists a solution of the form $v \propto p$; let us thus choose $v \stackrel{\text{def}}{=} \mathcal{L}p$ with $\mathcal{L} > 0$. In this case, to satisfy (7.6) we must have

$$\mathcal{L} = \lambda_{\max} \left(\mathbf{M}^{\frac{1}{2}} \mathbb{E} \left[\sum_{i \in S} \frac{1}{p_i} e_i e_i^\top \mathbf{W} \sum_{i \in S} \frac{1}{p_i} e_i e_i^\top \right] \mathbf{M}^{\frac{1}{2}} \right) \quad (7.7)$$

and the iteration complexity of SVRCD becomes $\left(\frac{4\mathcal{L} + \mu \rho^{-1}}{\mu} \right) \log \frac{1}{\epsilon}$.⁶

⁵Returns matrix with the input on the diagonal, zeros everywhere else.

⁶We decided to not present this, simplified rate in Theorem 7.3.2 for the following two reasons: 1) it would yields a slightly suboptimal rate of SEGA and 2) the connection of to the convergence rate of SAGA from [165] is more direct via (7.6).

How does \mathbf{W} influence the rate? As mentioned, one can always consider $\mathbf{W} = \mathbf{I}$. In such a case, we recover the convergence rate of SEGA and SVRCD from Chapter 5. However, the smaller rank of \mathbf{W} is, the faster rate is Theorem 7.3.2 providing. To see this, it suffices to realize that if \mathcal{L} is increasing in \mathbf{W} (in terms of Loewner ordering).

Example 4. Let $\mathbf{M} = \mathbf{I}$ and $S = \{i\}$ with probability d^{-1} for all $1 \leq i \leq d$. Given that $\mathbf{W} = \mathbf{I}$, it is easy to see that $\mathcal{L} = d$. In such case, the iteration complexity of SVRCD is $\left(\frac{4d+\mu\rho^{-1}}{\mu}\right) \log \frac{1}{\epsilon}$. In the other extreme, if $\mathbf{W} = \frac{1}{d}ee^\top$, we have $\mathcal{L} = 1$, which yields complexity (of SVRCD) $\left(\frac{4+\mu\rho^{-1}}{\mu}\right) \log \frac{1}{\epsilon}$. Therefore, given that $\mu = \mathcal{O}(\rho)$, the low rank of \mathbf{W} caused the speedup of order $\Theta(d)$.

We shall also note that the tight rate of SAGA and LSVRG might be recovered from Theorem 7.3.2 only using a non-trivial \mathbf{W} (see Section 7.4), while the original theory of SEGA and SVRCD only yield a suboptimal rate for both SAGA and LSVRG.

Connection with Subspace SEGA (from Section C.3). Assume that function f is of structure $f(x) = h(\mathbf{A}x)$. As a consequence, we have $\nabla f(x) = \mathbf{A}^\top \nabla h(\mathbf{A}x)$ and thus $\nabla f(x) \in \mathbf{Range}(\mathbf{A}^\top)$. This fact was exploited by Subspace SEGA in order to achieve a faster convergence rate. Our results can mimic Subspace SEGA by setting ψ to be an indicator function of $x^0 + \mathbf{Range}(\mathbf{A}^\top)$, given that there is no extra non-smooth term in the objective.

Remark 8. Throughout all proofs of this section, we have used a weaker conditions than Assumption 7.2.2. In particular, instead of \mathbf{M} -smoothness, it is sufficient to have⁷ $D_f(x, x^*) \geq \frac{1}{2} \|\nabla f(x) - \nabla f(x^*)\|_{\mathbf{M}^{-1}}^2$ for all $x \in \mathbb{R}^d$ (Lemma G.4.3 shows that it is indeed a consequence of \mathbf{M} smoothness and convexity). At the same time, instead of μ -strong convexity, it is sufficient to have μ -quasi strong convexity, i.e., for all $x \in \{x^0 + \mathbf{Range}(\mathbf{W})\}$: $f(x^*) \geq f(x) + \langle \nabla f(x), x^* - x \rangle + \frac{\mu}{2} \|x - x^*\|^2$. However, the accelerated method (presented in Section 7.5) requires the fully general version of Assumption 7.2.2.

7.4 Connection between SEGA (SVRCD) and SAGA (LSVRG)

In this section, we show that SAGA and LSVRG are special cases of SEGA and SVRCD, respectively. At the same time, the previously tightest convergence rate of SAGA [52, 165] and LSVRG [79, 164] follow from Theorem 7.3.2 (convergence rate of SEGA and SVRCD).

⁷By $D_f(x, y)$ we denote Bregman distance between x, y , i.e., $D_f(x, y) \stackrel{\text{def}}{=} f(x) - f(y) - \langle \nabla f(x), y - x \rangle$

7.4.1 Convergence rate of SAGA and LSVRG

We quickly state the best-known convergence rate for both SAGA and LSVRG to minimize the following objective:

$$\min_{\tilde{x} \in \mathbb{R}^{\tilde{d}}} \left\{ \tilde{F}(\tilde{x}) \stackrel{\text{def}}{=} \frac{1}{n} \sum_{j=1}^n \underbrace{\tilde{f}_j(\tilde{x})}_{\stackrel{\text{def}}{=} \tilde{f}(\tilde{x})} + \tilde{\psi}(\tilde{x}) \right\}. \quad (7.8)$$

Assumption 7.4.1. Each \tilde{f}_j is convex, $\tilde{\mathbf{M}}_j$ -smooth and \tilde{f} is $\tilde{\mu}$ -strongly convex.

Assuming the oracle access to $\nabla \tilde{f}_i(\tilde{x}^k)$ for $i \in \tilde{S}$ (where \tilde{S} is a random subset of $\{1, \dots, n\}$), the minibatch SGD [60] uses moves in the direction of the “plain” unbiased stochastic gradient $\frac{1}{n} \sum_{i \in \tilde{S}} \frac{1}{\tilde{p}_i} \nabla \tilde{f}_i(\tilde{x}^k)$ (where $\tilde{p}_i \stackrel{\text{def}}{=} \mathbb{P}(i \in \tilde{S})$).

In contrast, variance reduced methods such as SAGA and LSVRG enrich the “plain” unbiased stochastic gradient with control variates:

$$\tilde{g}^k = \frac{1}{n} \sum_{i \in \tilde{S}} \frac{1}{\tilde{p}_i} \left(\nabla \tilde{f}_i(\tilde{x}^k) - \mathbf{J}_{:,i}^k \right) + \frac{1}{n} \mathbf{J}^k \tilde{\mathbf{e}}. \quad (7.9)$$

where $\mathbf{J}^k \in \mathbb{R}^{\tilde{d} \times n}$ is the control matrix and $\tilde{\mathbf{e}} \in \mathbb{R}^n$ is vector of ones. The difference between SAGA and LSVRG lies in the procedure to update \mathbf{J}^k ; SAGA uses the freshest gradient information to replace corresponding columns in \mathbf{J}^k ; i.e.

$$\mathbf{J}_{:,i}^{k+1} = \begin{cases} \nabla \tilde{f}_i(\tilde{x}^k) & \text{if } i \in \tilde{S} \\ \mathbf{J}_{:,i}^k & \text{if } i \notin \tilde{S}. \end{cases} \quad (7.10)$$

On the other hand, LSVRG sets \mathbf{J}^k to the true Jacobian of f upon a successful, unfair coin toss:

$$\mathbf{J}^{k+1} = \begin{cases} \left[\nabla \tilde{f}_1(\tilde{x}^k), \dots, \nabla \tilde{f}_n(\tilde{x}^k) \right] & \text{w. p. } \rho \\ \mathbf{J}^k & \text{w. p. } 1 - \rho. \end{cases} \quad (7.11)$$

The formal statement of SAGA and LSVRG is provided in as Algorithm 16, while Proposition 7.4.2 states their convergence rate.

Proposition 7.4.2. ([79]) Suppose that Assumption 7.4.1 holds and let $\tilde{\mathbf{v}}$ be a nonnegative vector such that for all $h_1, \dots, h_n \in \mathbb{R}^{\tilde{d}}$ we have

$$\mathbb{E} \left[\left\| \sum_{j \in \tilde{S}} \tilde{\mathbf{M}}_j^{\frac{1}{2}} h_j \right\|^2 \right] \leq \sum_{j=1}^n \tilde{p}_j \tilde{v}_j \|h_j\|^2. \quad (7.12)$$

Then the iteration complexity of SAGA with $\tilde{\alpha} = \min_j \frac{n \tilde{p}_j}{4 \tilde{v}_j + n \tilde{\mu}}$ is $\max_j \left(\frac{4 \tilde{v}_j + n \tilde{\mu}}{n \tilde{\mu} \tilde{p}_j} \right) \log \frac{1}{\epsilon}$. At

Algorithm 16 SAGA/LSVRG

Require: $\alpha > 0$, $\rho \in (0, 1)$
 $\tilde{x}^0 \in \mathbb{R}^{\tilde{d}}$, $\mathbf{J}^0 = 0 \in \mathbb{R}^{\tilde{d} \times n}$
for $k = 0, 1, 2, \dots$ **do**
 Sample random $\tilde{S} \subseteq \{1, \dots, n\}$
 $\tilde{g}^k = \frac{1}{n} \mathbf{J}^k \tilde{e} + \frac{1}{n} \sum_{i \in \tilde{S}} \frac{1}{\tilde{p}_i} (\nabla \tilde{f}_i(\tilde{x}^k) - \mathbf{J}_{:,i}^k)$
 $\tilde{x}^{k+1} = \text{prox}_{\tilde{\alpha}\psi}(\tilde{x}^k - \tilde{\alpha}\tilde{g}^k)$
 Update \mathbf{J}^{k+1} according to (7.10) or (7.11)
end for

the same time, iteration complexity of LSVRG with $\tilde{\alpha} = \min_j \frac{n}{4\frac{\tilde{v}_j}{\tilde{p}_j} + \frac{\tilde{\mu}n}{\rho}}$ is $\max_j \left(4\frac{\tilde{v}_j}{n\tilde{\mu}\tilde{p}_j} + \frac{1}{\rho} \right) \log \frac{1}{\epsilon}$.

7.4.2 SAGA is a special case of SEGA

Consider setup from Section 7.4.1; i.e., problem (7.8) along with Assumption 7.4.1 and \tilde{v} defined according to (7.12). We will construct an instance of (7.1) (i.e., specific f , ψ), which is equivalent to (7.8), such that applying SEGA on (7.1) is equivalent applying SAGA on (7.8).

Let $d \stackrel{\text{def}}{=} \tilde{d}n$.

For convenience, define $R_j \stackrel{\text{def}}{=} \{\tilde{d}(j-1) + 1, \tilde{d}(j-1) + 1, \dots, \tilde{d}j\}$ (i.e., $|R_j| = \tilde{d}$) and

lifting operator $U(\cdot) : \mathbb{R}^{\tilde{d}} \rightarrow \mathbb{R}^d$ defined as $U(\tilde{x}) \stackrel{\text{def}}{=} \begin{bmatrix} \tilde{x}^\top, \dots, \tilde{x}^\top \end{bmatrix}^\top$.
 $\underbrace{\hspace{1.5cm}}_{n \text{ times}}$

Construction of f , ψ . Let I be indicator function of the set⁸ $x_{R_1} = \dots = x_{R_n}$ and choose

$$f(x) \stackrel{\text{def}}{=} \frac{1}{n} \sum_{j=1}^n \tilde{f}_j(x_{R_j}), \quad \psi(x) \stackrel{\text{def}}{=} I(x) + \tilde{\psi}(x_{R_1}) \quad (7.13)$$

Now, it is easy to see that problem (7.8) and problem (7.1) with the choice (7.13) are equivalent; each $x \in \mathbb{R}^d$ such that $F(x) < \infty$ must be of the form $x = U(\tilde{x})$ for some $\tilde{x} \in \mathbb{R}^{\tilde{d}}$. In such case, we have $F(x) = \tilde{F}(\tilde{x})$. The next lemma goes further, and derives the values \mathbf{M} , μ , \mathbf{W} and v based on $\tilde{\mathbf{M}}_i$ ($1 \leq i \leq n$), $\tilde{\mu}$, \tilde{v} .

Lemma 7.4.3. Consider f, ψ defined by (7.13). Function f satisfies Assumption 7.2.2 with $\mu \stackrel{\text{def}}{=} \frac{\tilde{\mu}}{n}$ and $\mathbf{M} \stackrel{\text{def}}{=} \frac{1}{n} \text{BlockDiag}(\tilde{\mathbf{M}}_1, \dots, \tilde{\mathbf{M}}_n)$. Function ψ and $x^0 = U(\tilde{x}^0)$ satisfy Assumption with $\mathbf{W} \stackrel{\text{def}}{=} \frac{1}{n} \tilde{e} \tilde{e}^\top \otimes \mathbf{I}$. At the same time, given that \tilde{v} satisfies (7.12), inequality (7.6) holds with $v = \tilde{v}n^{-1}$.

⁸Indicator function of a set returns 0 for each point inside of the set and ∞ for each point outside of the set.

Next, we show that running Algorithm 15 in this particular setup is equivalent to running Algorithm 16 for the finite sum objective.

Lemma 7.4.4. *Consider f, ψ from (7.13), S as described in the last paragraph and $x^0 = U(\tilde{x}^0)$. Running SEGA (SVRCD) on (7.1) with $S \stackrel{\text{def}}{=} \cup_{j \in \tilde{S}} R_j$ and $\alpha \stackrel{\text{def}}{=} n\tilde{\alpha}$ is equivalent to running SAGA (LSVRG) on (7.8); i.e., we have for all k*

$$x^k = U(\tilde{x}^k). \quad (7.14)$$

As a consequence of Lemmas 7.4.3 and 7.4.4, we get the next result.

Corollary 7.4.5. *Let f, ψ, S be as described above. Convergence rate of SAGA (LSVRG) given by Proposition 7.4.2 to solve (7.1) is identical to convergence rate of SEGA (SVRCD) given by Theorem 7.3.2.*

7.5 The ASVRCD algorithm

In this section we present SVRCD with Nesterov's momentum [149] – ASVRCD. The development of ASVRCD along with the theory (Theorem 7.5.1) was motivated by Katyusha [4], ASVRG [194] and their loopless variants [106, 164]. In Section 7.6.2, we show that a variant of L-Katyusha (Algorithm 18) is a special case of ASVRCD, and argue that it is slightly superior to the methods mentioned above.

The main component of ASVRCD is the gradient estimator g^k constructed analogously to SVRCD. In particular, g^k is an unbiased estimator of $\nabla f(x^k)$ controlled by $\nabla f(w^k)$:⁹

$$g^k = \nabla f(w^k) + \sum_{i \in S} \frac{1}{p_i} (\nabla_i f(x^k) - \nabla_i f(w^k)) e_i. \quad (7.15)$$

Next, ASVRCD requires two more sequences of iterates $\{y^k\}_{k \geq 0}, \{z^k\}_{k \geq 0}$ in order to incorporate Nesterov's momentum. The update rules of those sequences consist of subtracting g^k alongside with convex combinations or interpolations of the iterates. See Algorithm 17 for specific formulas.

We are now ready to present ASVRCD along with its convergence guarantees.

Theorem 7.5.1. *Let Assumption 7.2.1, 7.2.2 hold and denote $L \stackrel{\text{def}}{=} \lambda_{\max} \left(\mathbf{M}^{\frac{1}{2}} \mathbf{W} \mathbf{M}^{\frac{1}{2}} \right)$. Further, let \mathcal{L}' be such that for all k we have*

$$\mathbb{E} \left[\|g^k - \nabla f(x^k)\|_{\mathbf{W}}^2 \right] \leq 2\mathcal{L}' D_f(w^k, x^k). \quad (7.16)$$

Define the following Lyapunov function:

$$\Psi^k \stackrel{\text{def}}{=} \|z^k - x^*\|^2 + \frac{2\gamma\beta}{\theta_1} [F(y^k) - F(x^*)] + \frac{(2\theta_2 + \theta_1)\gamma\beta}{\theta_1\rho} [F(w^k) - F(x^*)],$$

⁹This is efficient to implement as sequence of iterates $\{w^k\}$ is updated rarely.

Algorithm 17 Accelerated SVRCD (ASVRCD)

Require: $0 < \theta_1, \theta_2 < 1$, $\eta, \beta, \gamma > 0$, $\rho \in (0, 1)$, $y^0 = z^0 = x^0 \in \mathbb{R}^d$

for $k = 0, 1, 2, \dots$ **do**

$$x^k = \theta_1 z^k + \theta_2 w^k + (1 - \theta_1 - \theta_2) y^k$$

Sample random $S \subseteq \{1, 2, \dots, d\}$

$$g^k = \nabla f(w^k) + \sum_{i \in S} \frac{1}{p_i} (\nabla_i f(x^k) - \nabla_i f(w^k)) e_i$$

$$y^{k+1} = \text{prox}_{\eta\psi}(x^k - \eta g^k)$$

$$z^{k+1} = \beta z^k + (1 - \beta) x^k + \frac{\gamma}{\eta} (y^{k+1} - x^k)$$

$$w^{k+1} = \begin{cases} y^k, & \text{with probability } \rho \\ w^k, & \text{with probability } 1 - \rho \end{cases}$$

end for

and let

$$\begin{aligned} \eta &= \frac{1}{4} \max\{\mathcal{L}', L\}^{-1}, \\ \theta_2 &= \frac{\mathcal{L}'}{2 \max\{L, \mathcal{L}'\}}, \\ \gamma &= \frac{1}{\max\{2\mu, 4\theta_1/\eta\}}, \\ \beta &= 1 - \gamma\mu \text{ and} \\ \theta_1 &= \min \left\{ \frac{1}{2}, \sqrt{\eta\mu \max \left\{ \frac{1}{2}, \frac{\theta_2}{\rho} \right\}} \right\}. \end{aligned}$$

Then the following inequality holds:

$$\mathbb{E} [\Psi^{k+1}] \leq \left[1 - \frac{1}{4} \min \left\{ \rho, \sqrt{\frac{\mu}{2 \max \left\{ L, \frac{\mathcal{L}'}{\rho} \right\}}} \right\} \right] \Psi^0.$$

As a consequence, iteration complexity of Algorithm 17 is $\mathcal{O} \left(\left(\frac{1}{\rho} + \sqrt{\frac{L}{\mu}} + \sqrt{\frac{\mathcal{L}'}{\rho\mu}} \right) \log \frac{1}{\epsilon} \right)$.

Convergence rate of ASVRCD depends on constant \mathcal{L}' such that (7.16) holds. The next lemma shows that \mathcal{L}' can be obtained indirectly from \mathcal{M} -smoothness (via \mathcal{L}), in which case the convergence rate provided by Theorem 7.5.1 significantly simplifies.

Lemma 7.5.2. *Inequality 7.16 holds for $\mathcal{L}' = \mathcal{L}$ (defined in (7.7)). Further, we have $L \leq \mathcal{L}$. Therefore, setting $\rho \geq \sqrt{\frac{\mu}{L}}$ yields the following complexity of ASVRCD:*

$$\mathcal{O} \left(\sqrt{\frac{\mathcal{L}}{\rho\mu}} \log \frac{1}{\epsilon} \right). \quad (7.17)$$

Setting $\mathcal{L}' = \mathcal{L}$ might be, however, loose in some cases. In particular, inequality (7.16)

is slightly weaker than (7.6) and consequently, the bound from Theorem 7.5.1 is slightly better than (7.17). To see this, notice that the proof of Lemma 7.5.2 bounds variance of $g^k + \nabla f(w^k)$ by its second moment. Admittedly, this bound might not worsen the rate by more than a constant factor when $\frac{\mathbb{E}[|S|]}{d}$ is not close to 1. Therefore, bound (7.17) is good in essentially all practical cases. The next reason why we keep inequality (7.16) is that an analogous assumption was required for the analysis of L-Katyusha in [164] (see Section 7.6.1) – and so we can now recover L-Katyusha results directly.

Let us give a quick taste how the rate of ASVRCD behaves depending on \mathbf{W} . In particular, Lemma 7.5.3 shows that nontrivial \mathbf{W} might lead to speedup of order $\Theta(\sqrt{d})$ for ASVRCD.

Lemma 7.5.3. *Let $S = i$ for each $1 \leq i \leq d$ with probability $\frac{1}{d}$ and $\rho = \frac{1}{d}$. Then, if $\mathbf{W} = \mathbf{I}$, iteration complexity of ASVRCD is $\mathcal{O}\left(d\sqrt{\frac{\lambda_{\max}\mathbf{M}}{\mu}} \log \frac{1}{\epsilon}\right)$. If, however, $\mathbf{W} = \frac{1}{d}ee^\top$, iteration complexity of ASVRCD is $\mathcal{O}\left(\sqrt{\frac{d\lambda_{\max}\mathbf{M}}{\mu}} \log \frac{1}{\epsilon}\right)$.*

7.6 Connection between ASVRCD and L-Katyusha

Next, we show that L-Katyusha can be seen as a particular case of ASVRCD.

7.6.1 Convergence rate of L-Katyusha

In this section, we quickly introduce the loopless Katyusha (L-Katyusha) from [164] along with its convergence guarantees. In the next section, we show that an improved version of L-Katyusha can be seen as a special case of ASVRCD, and at the same time, the tight convergence guarantees from [164] can be obtained as a special case of Theorem 7.5.1.

Consider problem (7.8) and suppose that \tilde{f} is \tilde{L} -smooth and $\tilde{\mu}$ -strongly convex. Let \tilde{S} be a random subset of $\{1, \dots, n\}$ (sampled from arbitrary fixed distribution) such that $\tilde{p}_i \stackrel{\text{def}}{=} \mathbb{P}(i \in \tilde{S})$. For each k let \tilde{g}^k be the following unbiased, variance reduced estimator of $\nabla \tilde{f}(x^k)$:

$$\tilde{g}^k = \frac{1}{n} \left(\sum_{i \in \tilde{S}} \tilde{p}_i^{-1} \left(\nabla \tilde{f}_i(\tilde{x}^k) - \nabla \tilde{f}_i(\tilde{w}^k) \right) \right) + \nabla \tilde{f}(\tilde{w}^k).$$

Next, L-Katyusha requires the variance of \tilde{g}^k to be bounded by Bregman distance between \tilde{w}^k and \tilde{x}^k with constant $\tilde{\mathcal{L}}$, as the next assumption states.

Assumption 7.6.1. *For all k we have*

$$\mathbb{E} \left[\|\tilde{g}^k - \nabla \tilde{f}(\tilde{x}^k)\|^2 \right] \leq 2\tilde{\mathcal{L}}D_f(\tilde{w}^k, \tilde{x}^k). \quad (7.18)$$

Proposition 7.6.2 provides a convergence rate of L-Katyusha.

Proposition 7.6.2. *([164]) Let \tilde{f} be \tilde{L} -smooth and $\tilde{\mu}$ -strongly convex while Assumption 7.6.1 holds. Iteration complexity of L-Katyusha is $\mathcal{O}\left(\left(\frac{1}{\tilde{p}} + \sqrt{\frac{\tilde{L}}{\tilde{\mu}}} + \sqrt{\frac{\tilde{\mathcal{L}}}{\tilde{\mu}\tilde{p}}}\right) \log \frac{1}{\epsilon}\right)$.*

7.6.2 L-Katyusha is a special case of ASVRCD

In this section, we show that a modified version of L-Katyusha (Algorithm 18) is a special case of ASVRCD. Furthermore, we show that the tight convergence rate of L-Katyusha [164] follows from Theorem 7.5.1 (convergence rate of ASVRCD).

Consider again f, ψ chosen according to (7.13). With this choice, problem (7.1) and (7.8) are equivalent. At the same time, Lemma 7.4.4 establishes that f satisfies Assumption 7.2.2 with $\mu = \frac{\tilde{\mu}}{n}$ and $\mathbf{M} = \frac{1}{n} \text{BlockDiag}(\tilde{\mathbf{M}}_1, \dots, \tilde{\mathbf{M}}_n)$ while ψ and x^0 satisfy Assumption with $\mathbf{W} \stackrel{\text{def}}{=} \frac{1}{n} \tilde{\mathbf{e}} \tilde{\mathbf{e}}^\top \otimes \mathbf{I}$.

Note that the update rule of sequences x^k, z^k, w^k are identical for both algorithms; we shall thus verify that the update rule on y^k is identical as well. The last remaining thing is to relate \mathcal{L}' and $\tilde{\mathcal{L}}$. The next lemma establishes both results.

Lemma 7.6.3. *Running ASVRCD on (7.1) with $S \stackrel{\text{def}}{=} \cup_{j \in \tilde{S}} R_j$ and $\eta \stackrel{\text{def}}{=} n\tilde{\eta}$, $\gamma \stackrel{\text{def}}{=} n\tilde{\gamma}$ is equivalent to running Algorithm 18 on (7.8). At the same time, inequality 7.16 holds with $\mathcal{L}' = n^{-1}\tilde{\mathcal{L}}$, while we have $L = n^{-1}\tilde{L}$.*

As a direct consequence of Lemma 7.6.3 and Theorem 7.5.1, we obtain the next corollary.

Corollary 7.6.4. *Let f, ψ, S be as described above. Iteration complexity of Algorithm 18 is*

$$\mathcal{O} \left(\left(\frac{1}{\tilde{p}} + \sqrt{\frac{\tilde{L}}{\tilde{\mu}}} + \sqrt{\frac{\tilde{\mathcal{L}}}{\tilde{\mu}\tilde{p}}} \right) \log \frac{1}{\epsilon} \right).$$

As promised, the convergence rate of Algorithm 18 matches the convergence rate of L-Katyusha from Proposition 7.6.2 and thus matches the lower bound for finite sum minimization by [214]. Let us now argue that Algorithm 18 is slightly superior to other accelerated SVRG variants.

First, Algorithm 18 is loopless; thus has a simpler analysis and slightly better properties (as shown by [106]) over Katyusha [4] and ASVRG [194]. Next, the analysis is simpler than [164] (i.e., we do not require one page of going through special cases). At the same time, Algorithm 18 uses a smaller stepsize for the proximal operator than L-Katyusha, which is useful if the proximal operator does is estimated numerically. However, Algorithm 18 is almost indistinguishable from L-Katyusha if $\tilde{\psi} = 0$.

Remark 9. The convergence rate of L-Katyusha from [164] allows exploiting the strong convexity of regularizer ψ (given that it is strongly convex). While such a result is possible to obtain in our case, we have omitted it for simplicity.

7.7 Experiments

In this section, we numerically verify the performance of ASVRCD, as well as the improved performance of SVRCD under Assumption 7.2.1. In order to better understand and control

Algorithm 18 Variant of L-Katyusha (special case of Algorithm 17)**Require:** $0 < \theta_1, \theta_2 < 1$, $\tilde{\eta}, \beta, \tilde{\gamma} > 0$, $\rho \in (0, 1)$

$$\tilde{y}^0 = \tilde{z}^0 = \tilde{x}^0 \in \mathbb{R}^{\tilde{d}}$$

for $k = 0, 1, 2, \dots$ **do**

$$\tilde{x}^k = \theta_1 \tilde{z}^k + \theta_2 \tilde{w}^k + (1 - \theta_1 - \theta_2) \tilde{y}^k$$

Sample random $\tilde{S} \subseteq \{1, 2, \dots, n\}$

$$g^k = \nabla \tilde{f}(\tilde{w}^k) + \sum_{i \in \tilde{S}} \frac{1}{\tilde{p}_i} (\nabla \tilde{f}_i(\tilde{x}^k) - \nabla \tilde{f}_i(\tilde{w}^k))$$

$$\tilde{y}^{k+1} = \text{prox}_{\tilde{\eta}\psi}(\tilde{x}^k - \tilde{\eta}g^k)$$

$$\tilde{z}^{k+1} = \beta \tilde{z}^k + (1 - \beta) \tilde{x}^k + \frac{\tilde{\gamma}}{\tilde{\eta}} (\tilde{y}^{k+1} - \tilde{x}^k)$$

$$\tilde{w}^{k+1} = \begin{cases} \tilde{y}^k, & \text{with probability } \rho \\ \tilde{w}^k, & \text{with probability } 1 - \rho \end{cases}$$

end for

the experimental setup, we consider a quadratic minimization (four different types) over the unit ball intersected with a linear subspace.¹⁰

In all experiments, we have chosen

$$f(x) = \frac{1}{2} x^\top \mathbf{M} x - b^\top x,$$

where $x \in \mathbb{R}^{1000}$, while ψ is an indicator function of the unit ball intersected with $\text{Range}(\mathbf{W})$. First, matrix \mathbf{M} was chosen according to Table 7.1. Next, vector b was chosen as follows: first we generate $\tilde{x} \in \mathbb{R}^d$ with independent normal entries, then compute $\tilde{b} = \mathbf{M}^{-1} \tilde{x}$ and set $b = \frac{3}{2\|\tilde{b}\|} \tilde{b}$. Lastly, for Figure 7.2, the projection matrix \mathbf{W} of rank r was chosen as a block diagonal matrix with r blocks, each of them being the matrix of ones multiplied by $\frac{r}{d}$.

Table 7.1: Choice of \mathbf{M} . O_{dd} is set of all odd positive integers smaller than $d + 1$, while matrix \mathbf{U} was set as random orthonormal matrix (generated by QR decomposition from a matrix with independent standard normal entries).

Type	\mathbf{M}	Fig. 7.1: L	Fig. 7.2: L
1	$\mathbf{U} \left(\mathbf{I} + \mathbf{I}_{:,O_{dd}} \mathbf{Diag} \left(((L-1)^{\frac{1}{500}})^{(1:500)} \right) \mathbf{I}_{O_{dd},:} \right) \mathbf{U}^\top$	100	1000
2	$\mathbf{U} \left(\mathbf{I} + \sum_{i=1}^{100} (L-1) e_i e_i^\top \right) \mathbf{U}^\top$	100	1000
3	$\mathbf{U} \left(\kappa \mathbf{I} - \sum_{i=1}^{100} (L-1) e_i e_i^\top \right) \mathbf{U}^\top$	100	1000
4	$\left(\mathbf{I} + \frac{L}{500} \mathbf{I}_{:,O_{dd}} \mathbf{Diag} (1 : 500) \mathbf{I}_{O_{dd},:} \right)$	100	1000

¹⁰Note that the practicality of ASVRCD immediately follows as it recovers Algorithm 18 as a special case, which is (especially for $\psi \equiv 0$) almost indistinguishable to L-Katyusha – state-of-the-art method for smooth finite sum minimization. For this reason, we decided to focus on less practical, but better-understood experiments.

7.7.1 The effect of acceleration and importance sampling

In the first experiment we demonstrate the superiority of ASVRCD to SVRCD for problems with $\mathbf{W} = \mathbf{I}$. We consider four different methods – ASVRCD and SVRCD, both with uniform and importance sampling such that $|S| = 1$ with probability 1. The importance sampling is the same as one from Chapter 5. In short, the goal is to have \mathcal{L} from (7.7) as small as possible. Using $\mathbf{W} = \mathbf{I}$, it is easy to see that $\mathcal{L} = \lambda_{\max} \left(\text{Diag}(p)^{-\frac{1}{2}} \mathbf{M} \text{Diag}(p)^{-\frac{1}{2}} \right)$. While the optimal p is still hard to find, we set $p_i \propto \mathbf{M}_{i,i}$ (i.e., the effect of importance sampling is the same as the effect of Jacobi preconditioner). Figure 7.1 shows the result. As expected, accelerated SVRCD always outperforms non-accelerated variant, while at the same time, the importance sampling improves the performance too.

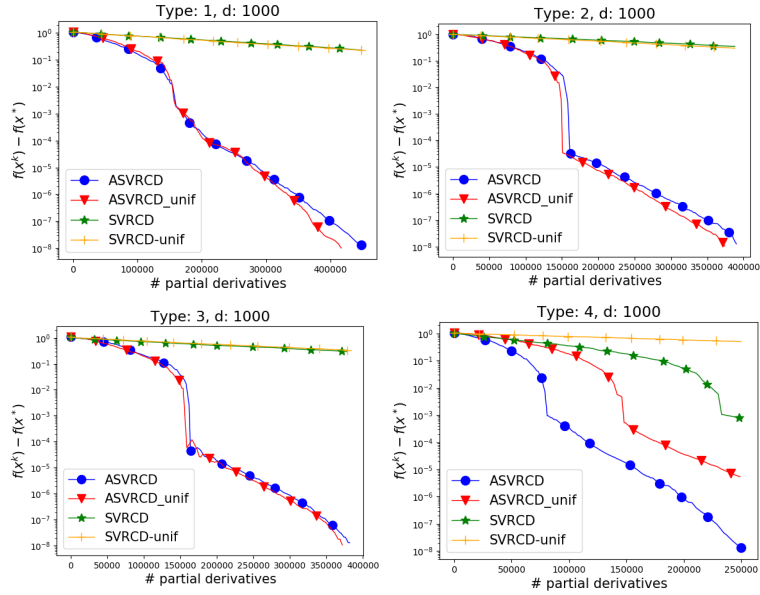


Figure 7.1: Comparison of both ASVRCD and SVRCD with importance and uniform sampling.

7.7.2 The effect of \mathbf{W}

The second experiment compares the performance of both ASVRCD and SVRCD for various \mathbf{W} . We only consider methods with importance sampling ($p_i \propto \mathbf{M}_{i,i} \mathbf{W}_{i,i}$) and theory supported stepsize. Figure 7.2 presents the result. We see that the smaller $\text{Range}(\mathbf{W})$ is, the faster the convergence is. This observation is well-aligned with our theory: \mathcal{L} is increasing as a function of \mathbf{W} (in terms of Loewner ordering).

7.8 Conclusion

In this chapter we have introduced ASVRCD – an accelerated SVRCD algorithm. Besides that, we have shown that SAGA/tt L-Katyusha are a special case of SEGA/ASVRCD, while their convergence guarantees can be recovered. This rationale can be further generalized: it is possible to show that essentially any finite-sum stochastic algorithm is a special case

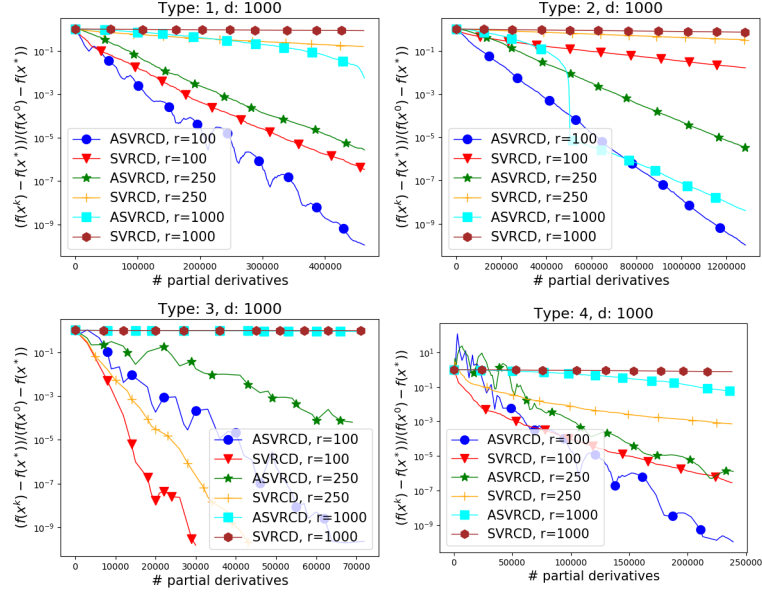


Figure 7.2: Comparison of ASVRCD and SVRCD for various \mathbf{W} . Label 'r' indicates the dimension of $\text{Range}(\mathbf{W})$.

of analogous method with partial derivative oracle (those are yet to be discovered/analyzed) in a given setting (i.e., strongly convex, convex, non-convex). Those include, but are not limited to SGD [179, 148], over-parametrized SGD [208], SAG [182], SVRG [88], S2GD [101], SARAH [160], incremental methods such as Finito [38], MISO [133] or accelerated algorithms such as point-SAGA [36], Katyusha [4], MiG [227], SAGA-SSNM [226], Catalyst [121, 111], non-convex variance reduced algorithms [171, 5, 47] and others. In particular, SGD can be seen as a special case of block coordinate descent, while SAG is a special case of bias-SEGA from [77] (neither of CD with non-separable prox, nor bias-SEGA were analyzed yet).

Chapter 8

Federated Learning of a Mixture of Global and Local Models

With the proliferation of mobile phones, wearable devices, tablets, and smart home devices comes an increase in the volume of data captured and stored on them. This data contains a wealth of potentially useful information to the owners of these devices, and more so if appropriate machine learning models could be trained on the heterogeneous data stored across the network of such devices. The traditional approach involves moving the relevant data to a data center where centralized machine learning techniques can be efficiently applied [35, 172]. However, this approach is not without issues. First, many device users are increasingly sensitive to privacy concerns and prefer their data to never leave their devices. Second, moving data from their place of origin to a centralized location is very inefficient in terms of energy and time.

8.1 Federated learning

Federated learning (FL) [134, 104, 103, 135] has emerged as an interdisciplinary field focused on addressing these issues by training machine learning models directly on edge devices. The currently prevalent paradigm [119, 90] casts supervised FL as an empirical risk minimization problem of the form

$$\min_{x \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^n f_i(x), \quad (8.1)$$

where n is the number of devices participating in training, $x \in \mathbb{R}^d$ encodes the d parameters of a *global model* (e.g., weights of a neural network) and $f_i(x) \stackrel{\text{def}}{=} \mathbb{E}_{\xi \sim \mathcal{D}_i} f(x, \xi)$ represents the aggregate loss of model x on the local data represented by distribution \mathcal{D}_i stored on device i . One of the defining characteristics of FL is that the data distributions \mathcal{D}_i may possess very different properties across the devices. Hence, any potential FL method is explicitly required to be able to work under the *heterogeneous* data setting.

The most popular method for solving (8.1) in the context of FL is the FedAvg algorithm [134]. In its most simple form, when one does not employ partial participation, model compression, or stochastic approximation, FedAvg reduces to Local Gradient Descent (LGD) [95, 96], which is an extension of GD performing more than a single gradient step on each device before aggregation. FedAvg has been shown to work well empirically, particularly for non-convex problems, but comes without convergence guarantees and can diverge in practical settings when data are heterogeneous.

8.1.1 Some issues with current approaches to FL

The first motivation for our research comes from the appreciation that data heterogeneity does not merely present challenges to the design of new provably efficient training methods for solving (8.1), but *also inevitably raises questions about the utility of such a global solution to individual users*. Indeed, a global model trained across all the data from all devices might be so removed from the typical data and usage patterns experienced by an individual user as to render it virtually useless. This issue has been observed before, and various approaches have been proposed to address it. For instance, the MOCHA [196] framework uses a multi-task learning approach to allow for personalization. [97] propose a generic online algorithm for gradient-based parameter-transfer meta-learning and demonstrate improved practical performance over FedAvg [135]. Approaches based on variational inference [28], cyclic patterns in practical FL data sampling [46] and transfer learning [225] have been proposed.

The second motivation for our work is the realization that even very simple variants of FedAvg, such as LGD, which should be easier to analyze, fail to provide theoretical improvements in communication complexity over their non-local cousins, in this case, GD [95, 96]. This observation is at odds with the practical success of local methods in FL. This leads us to ask the question:

If LGD does not theoretically improve upon GD as a solver for the traditional global problem (8.1), perhaps LGD should not be seen as a method for solving (8.1) at all. In such a case, what problem does LGD solve?

A good answer to this question would shed light on the workings of LGD, and by analogy, on the role local steps play in more elaborate FL methods such as local SGD [198, 96] and FedAvg.

8.2 Contributions

In our work we argue that the two motivations mentioned in the introduction point in the same direction, i.e., we show that *a single solution can be devised addressing both problems at the same time*.

Our main contributions are:

- **New formulation of FL which seeks a mixture of global and local models.**

We propose a *new optimization formulation of FL*. Instead of learning a single global model by solving (8.1), we propose to learn a mixture of the global model and the purely local models which can be trained by each device i on its own, using its data \mathcal{D}_i only. Our formulation (see (8.2) in Section 8.3) lifts the problem from \mathbb{R}^d to \mathbb{R}^{nd} , allowing each device i to learn a personalized model $x_i \in \mathbb{R}^d$. However, these personalized models are explicitly encouraged to not depart too much from their mean by the inclusion of a quadratic penalty Φ multiplied by a penalty parameter $\lambda \geq 0$.¹

¹The idea of softly-enforced similarity of the local models was already introduced in the domain of

- Theoretical properties of the new formulation.** We study the properties of the optimal solution of our formulation, thus developing an algorithmic-free theory. When the penalty parameter is set to zero, then obviously, each device is allowed to train their own model without any dependence on the data stored on other devices. Such purely local models are rarely useful. We prove that the optimal local models converge to the traditional global model characterized by (8.1) at the rate $\mathcal{O}(1/\lambda)$. We also show that the total loss evaluated at the local models is always not higher than the total loss evaluated at the global model (see Theorem 8.3.2). Moreover, we prove an insightful structural result for the optimal local models: the optimal model learned by device i arises by subtracting the gradient of the loss function stored on that device evaluated at the same point (i.e., a local model) from the average of the optimal local models (see Theorem 8.3.3). As a byproduct, this theoretical result sheds new light on the key update step in the model agnostic meta-learning (MAML) method of [50], which has a similar but subtly different structure. The subtle difference is that the MAML update obtains the local model by subtracting the gradient evaluated at the *global* model. While MAML is a heuristic, we provide rigorous theoretical guarantees.
- Loopless LGD: non-uniform SGD applied to our formulation.** We then propose a randomized gradient-based method—*Loopless Local Gradient Descent* (LLGD)—for solving our new formulation (Algorithm 19). This method is, in fact, a non-standard application of SGD to our problem, and can be seen as an instance of SGD with non-uniform sampling applied to the problem of minimizing the sum of two convex functions [223, 60]: the average loss, and the penalty. When the loss function is selected by the randomness in our SGD method, the resultant stochastic gradient step can be *interpreted* as the execution of a single local GD step on each device. Since we set the probability of the loss being sampled to be high, this step is typically repeated multiple times, and this has the effect of taking multiple local GD steps. In contrast to standard LGD, the number of local steps is not fixed, but random, and follows a geometric distribution. This mechanism is similar in spirit to how the recently proposed loopless variants of SVRG [83, 106] work in comparison with the original SVRG [88, 217]. Once the penalty is sampled by our method, the resultant SGD step can be interpreted as the execution of an aggregation step. In contrast with standard aggregation, which performs full averaging of the local models, our method is more sophisticated and merely takes a *step towards averaging*. However, the step is relatively large. This suggests that perhaps full averaging in modern FL methods such as FedAvg or LGD and LSGD is too aggressive, and should be re-examined.
- Convergence theory.** By adapting the general theory from [60] to our setting, we obtain theoretical convergence guarantees assuming that each f_i is L -smooth and μ -strongly convex (see Theorem 8.4.3). Interestingly, by optimizing the sampling

decentralized optimization by [112, 54]. However, their motivation is vastly different to ours (besides not considering FL or local algorithms) – the mentioned methods still aim to find the global model by having the penalty parameter inversely proportional to the target accuracy ε .

probability (we get $p^* = \frac{\lambda}{\lambda+L}$) which is an indirect way of fixing the *expected number of local steps* to $1 + \frac{L}{\lambda}$, we prove the communication complexity result (i.e., bound on the expected number of communication rounds; see Corollary 8.4.4)

$$\frac{2\lambda}{\lambda+L} \frac{L}{\mu} \log \frac{1}{\varepsilon}.$$

We believe that this is remarkable in several ways. By choosing λ small, we tilt our goal towards pure local models, and the number of communication rounds is very small, tending to 0 as $\lambda \rightarrow 0$. If $\lambda \rightarrow \infty$, our the solution to our formulation converges to the optimal global model, and L2GD obtains the communication bound $\mathcal{O}\left(\frac{L}{\mu} \log \frac{1}{\varepsilon}\right)$, which matches the efficiency of GD. Our results can be extended to convex and non-convex regimes, but we do not explore such generalizations here.

- **Generalizations: partial participation, local SGD and variance reduction.** We further generalize and improve our method and convergence results by allowing for
 - (i) stochastic *partial participation* of devices in each communication round,
 - (ii) *subsampling* on each device which means we can perform local SGD steps instead of local GD steps, and
 - (iii) *total variance reduction mechanism* to tackle the variance coming from three sources: locality of the updates induced by non-uniform sampling (already present in L2GD), partial participation and subsampling from local data.

Due to its level of generality, this method, which we call L2SGD++, is presented in the Appendix only, alongside the associated complexity results. In the main body of this chapter, we instead present a simplified version thereof, one that does not include partial participation. We call this method L2SGD+ (Algorithm 20). The convergence theory for it is presented in Theorem 8.5.2 and Corollary 8.5.3.

- **Allowing for heterogeneous data.** All our methods and convergence results allow for fully heterogeneous data and do not depend on any assumptions on data similarity across the devices.
- **Superior empirical performance.** We show through ample numerical experiments that our theoretical predictions can be observed in practice.

8.3 New formulation of FL

We now introduce our new formulation for training supervised FL models:

$$\begin{aligned} \min_{x_1, \dots, x_n \in \mathbb{R}^d} \left\{ F(x) \stackrel{\text{def}}{=} f(x) + \lambda \Phi(x) \right\} \\ f(x) \stackrel{\text{def}}{=} \frac{1}{n} \sum_{i=1}^n f_i(x_i), \quad \Phi(x) \stackrel{\text{def}}{=} \frac{1}{2n} \sum_{i=1}^n \|x_i - \bar{x}\|^2, \end{aligned} \tag{8.2}$$

where $\lambda \geq 0$ is a penalty parameter, $x_1, \dots, x_n \in \mathbb{R}^d$ are local models, $x \stackrel{\text{def}}{=} (x_1, x_2, \dots, x_n) \in \mathbb{R}^{nd}$ and $\bar{x} \stackrel{\text{def}}{=} \frac{1}{n} \sum_{i=1}^n x_i$ is the average of the local models.

Due to the assumptions on f_i we will make in Section 8.3.1, F is strongly convex and hence (8.2) has a unique solution, which we denote

$$x(\lambda) \stackrel{\text{def}}{=} (x_1(\lambda), \dots, x_n(\lambda)) \in \mathbb{R}^{nd}.$$

We further let

$$\bar{x}(\lambda) \stackrel{\text{def}}{=} \frac{1}{n} \sum_{i=1}^n x_i(\lambda).$$

We now comment on the rationale behind the new formulation.

Local models ($\lambda = 0$). Note that for each i , $x_i(0)$ solves the *local problem*

$$\min_{x_i \in \mathbb{R}^d} f_i(x_i).$$

That is, $x_i(0)$ is the local model based on data \mathcal{D}_i stored on device i only. This model can be computed by device i without any communication whatsoever. Typically, \mathcal{D}_i is not rich enough for this local model to be useful. In order to learn a better model, one has to take into account the data from other clients as well. This, however, requires communication.

Mixed models ($\lambda \in (0, \infty)$). As λ increases, the penalty $\lambda\Phi(x)$ has an increasingly more substantial effect, and communication is needed to ensure that the models are not too dissimilar, as otherwise Φ would be too large.

Global model ($\lambda = \infty$). Let us now look at the limit case $\lambda \rightarrow \infty$. Intuitively, this limit case should force the optimal local models to be mutually identical, while minimizing the loss f . In particular, this limit case will solve²

$$\min_{x_1, \dots, x_n \in \mathbb{R}^d} \{f(x) : x_1 = x_2 = \dots = x_n\},$$

which is equivalent to the global formulation (8.2). Because of this, let us define $x_i(\infty)$ for each i to be the optimal global solution of (8.1), and let $x(\infty) \stackrel{\text{def}}{=} (x_1(\infty), \dots, x_n(\infty))$.

8.3.1 Technical preliminaries

Similarly to the rest of the thesis, we make the following assumption on the functions f_i :

Assumption 8.3.1. *For each i , the function $f_i : \mathbb{R}^d \rightarrow \mathbb{R}$ is L -smooth and μ -strongly convex.*

²If $\lambda = \infty$ and $x_1 = x_2 = \dots = x_n$ does not hold, we have $F(x) = \infty$. Therefore, we can restrict ourselves on set $x_1 = x_2 = \dots = x_n$ without loss of generality.

Note that the separable structure of f implies that $(\nabla f(x))_i = \frac{1}{n} \nabla f_i(x_i)$, i.e.,

$$\nabla f(x) = \frac{1}{n} (\nabla f_1(x_1), \nabla f_2(x_2), \dots, \nabla f_n(x_n)). \quad (8.3)$$

Hence, the norm of $\nabla f(x) \in \mathbb{R}^{nd}$ decomposes as $\|\nabla f(x)\|^2 = \frac{1}{n} \sum_{i=1}^n \|\nabla f_i(x_i)\|^2$.

Note that Assumption 8.3.1 implies that f is L_f -smooth with $L_f \stackrel{\text{def}}{=} \frac{L}{n}$ and μ_f -strongly convex with $\mu_f \stackrel{\text{def}}{=} \frac{\mu}{n}$. Clearly, Φ is convex by construction. It can be shown that Φ is L_Φ -smooth with $L_\Phi = \frac{1}{n}$ (see Appendix). We can also easily see that

$$(\nabla \Phi(x))_i = \frac{1}{n} (x_i - \bar{x}) \quad (8.4)$$

(see Appendix), which implies

$$\Phi(x) \stackrel{(8.2)+(8.4)}{=} \frac{n}{2} \sum_{i=1}^n \|(\nabla \Phi(x))_i\|^2 = \frac{n}{2} \|\nabla \Phi(x)\|^2.$$

8.3.2 Characterization of optimal solutions

Our first result describes the behavior of $f(x(\lambda))$ and $\Phi(x(\lambda))$ as a function of λ .

Theorem 8.3.2. *The function $\lambda \rightarrow \Phi(x(\lambda))$ is non-increasing, and for all $\lambda > 0$ we have*

$$\Phi(x(\lambda)) \leq \frac{f(x(\infty)) - f(x(0))}{\lambda}. \quad (8.5)$$

Moreover, the function $\lambda \rightarrow f(x(\lambda))$ is non-decreasing, and for all $\lambda \geq 0$ we have

$$f(x(\lambda)) \leq f(x(\infty)). \quad (8.6)$$

Inequality (8.5) says that the penalty decreases to zero as λ grows, and hence the optimal local models $x_i(\lambda)$ are increasingly similar as λ grows. The second statement suggest that the loss $f(x(\lambda))$ increases with λ , but never exceeds the optimal global loss $f(x(\infty))$ of the standard FL formulation (8.1).

We now characterize the optimal local models which connect our model to the MAML framework [50], as mentioned in the introduction.

Theorem 8.3.3. *For each $\lambda > 0$ and $1 \leq i \leq n$ we have*

$$x_i(\lambda) = \bar{x}(\lambda) - \frac{1}{\lambda} \nabla f_i(x_i(\lambda)). \quad (8.7)$$

Further, we have $\sum_{i=1}^n \nabla f_i(x_i(\lambda)) = 0$ and $\Phi(x(\lambda)) = \frac{1}{2\lambda^2} \|\nabla f(x(\lambda))\|^2$.

The optimal local models (8.7) are obtained from the average model by subtracting a multiple of the local gradient. Moreover, observe that the local gradients always sum up

to zero at optimality. This is obviously true for $\lambda = \infty$, but it is a bit less obvious that this holds for any $\lambda > 0$.

8.4 The L2GD algorithm

In this section we describe a new randomized gradient-type method for solving our new formulation (8.2). Our method is a non-uniform SGD for (8.2) seen as a 2-sum problem, sampling either ∇f or $\nabla \Phi$ to estimate ∇F . Letting $0 < p < 1$, we define a stochastic gradient of F at $x \in \mathbb{R}^{nd}$ as follows

$$g(x) \stackrel{\text{def}}{=} \begin{cases} \frac{\nabla f(x)}{1-p} & \text{with probability } 1-p \\ \frac{\lambda \nabla \Phi(x)}{p} & \text{with probability } p \end{cases}. \quad (8.8)$$

Since

$$\mathbb{E}[g(x)] = (1-p) \frac{\nabla f(x)}{1-p} + p \frac{\lambda \nabla \Phi(x)}{p} = \nabla F(x),$$

the vector $g(x)$ is an unbiased estimator of $\nabla F(x)$. This leads to the following method for minimizing F , which we call L2GD:

$$x^{k+1} = x^k - \alpha G(x^k). \quad (8.9)$$

Plugging formulas (8.3) and (8.4) for $\nabla f(x)$ and $\nabla \Phi(x)$ into (8.8) and subsequently into (8.9), and writing the resulting method in a distributed manner, we arrive at Algorithm 19. In each iteration, a coin ξ is tossed and lands 1 with probability p and 0 with probability $1-p$. If $\xi = 0$, all **Devices** perform one local GD step (8.10), and if $\xi = 1$, **Master** shifts each local model towards the average via (8.11). As we shall see in Section 8.4.3, our convergence theory limits the value of the stepsize α , which has the effect that the ratio $\frac{\alpha\lambda}{np}$ cannot exceed $\frac{1}{2}$. Hence, (8.11) is a convex combination of x_i^k and \bar{x}^k , which justifies the statement we have made above: x_i^{k+1} shifts towards \bar{x}^k along the line joining these two points.

8.4.1 Understanding communication

Example 5. In order to better understand when communication takes place in Algorithm 19, consider the following possible sequence of coin tosses: 0, 0, 1, 0, 1, 1, 1, 0. The first two coin tosses lead to two local GD steps (8.10) on all devices. The third coin toss lands 1, at which point all local models x_i^k are communicated to the master, averaged to form \bar{x}^k , and the step (8.11) towards averaging is taken. The fourth coin toss is 0, and at this point, the master communicates the updated local models back to the devices, which subsequently perform a single local GD step (8.10). Then come three consecutive coin tosses landing 1, which means that the local models are again communicated to the master, which performs three averaging steps (8.11). Finally, the eighth coin toss lands 0, which makes the master send the updated local models back to the devices, which subsequently perform a single local GD step.

Algorithm 19 L2GD: Loophes Local Gradient Descent

Input: $x_1^0 = \dots = x_n^0 \in \mathbb{R}^d$, stepsize α , probability p
for $k = 0, 1, 2, \dots$ **do**
 $\xi = 1$ with probability p and 0 with probability $1 - p$
if $\xi = 0$ **then**
All **Devices** $i = 1, \dots, n$ perform a local GD step:

$$x_i^{k+1} = x_i^k - \frac{\alpha}{n(1-p)} \nabla f_i(x_i^k) \quad (8.10)$$

else

Master computes the average $\bar{x}^k = \frac{1}{n} \sum_{i=1}^n x_i^k$
Master for each i computes step towards aggregation

$$x_i^{k+1} = \left(1 - \frac{\alpha\lambda}{np}\right) x_i^k + \frac{\alpha\lambda}{np} \bar{x}^k \quad (8.11)$$

end if
end for

This example illustrates that communication needs to take place whenever two consecutive coin tosses land a different value. If 0 is followed by a 1, all devices communicate to the master, and if 1 is followed by a 0, the master communicates back to the devices. It is standard to count each pair of communications, **Device**→**Master** and the subsequent **Master**→**Device**, as a single communication round.

Lemma 8.4.1. *The expected number of communication rounds in k iterations of L2GD is $p(1-p)k$.*

8.4.2 The dynamics of local GD and averaging steps

Further, notice that *the average of the local models does not change* during an aggregation step. Indeed, \bar{x}^{k+1} is equal to

$$\frac{1}{n} \sum_{i=1}^n x_i^{k+1} \stackrel{(8.11)}{=} \frac{1}{n} \sum_{i=1}^n \left[\left(1 - \frac{\alpha\lambda}{np}\right) x_i^k + \frac{\alpha\lambda}{np} \bar{x}^k \right] = \bar{x}^k.$$

If several averaging steps take place in a sequence, the point $a = \bar{x}^k$ in (8.11) remains unchanged, and each local model x_i^k merely moves along the line joining the initial value of the local model at the start of the sequence and a , with each step pushing x_i^k closer to the average a .

In summary, the more local GD steps are taken, the closer the local models get to the pure local models, and the more averaging steps are taken, the closer the local models get to their average value. The relative number of local GD vs. averaging steps is controlled by the parameter p : the expected number of local GD steps is $\frac{1}{p}$, and the expected number of consecutive aggregation steps is $\frac{1}{1-p}$.

8.4.3 Convergence theory

We first show that our gradient estimator $g(x)$ satisfies the expected smoothness property [65, 60].

Lemma 8.4.2. *Let $\mathcal{L} \stackrel{\text{def}}{=} \frac{1}{n} \max \left\{ \frac{L}{1-p}, \frac{\lambda}{p} \right\}$ and*

$$\sigma^2 \stackrel{\text{def}}{=} \frac{1}{n^2} \sum_{i=1}^n \left(\frac{1}{1-p} \|\nabla f_i(x_i(\lambda))\|^2 + \frac{\lambda^2}{p} \|x_i(\lambda) - \bar{x}(\lambda)\|^2 \right).$$

Then for all $x \in \mathbb{R}^d$ we have the inequalities

$$\mathbb{E} [\|g(x) - G(x(\lambda))\|^2] \leq 2\mathcal{L} (F(x) - F(x(\lambda)))$$

and

$$\mathbb{E} [\|g(x)\|^2] \leq 4\mathcal{L}(F(x) - F(x(\lambda))) + 2\sigma^2.$$

We now present our convergence result for L2GD.

Theorem 8.4.3. *Let Assumption 8.3.1 hold. If $\alpha \leq \frac{1}{2\mathcal{L}}$, then*

$$\mathbb{E} [\|x^k - x(\lambda)\|^2] \leq \left(1 - \frac{\alpha\mu}{n}\right)^k \|x^0 - x(\lambda)\|^2 + \frac{2n\alpha\sigma^2}{\mu}.$$

If we choose $\alpha = \frac{1}{2\mathcal{L}}$, then $\frac{\alpha\mu}{n} = \frac{\mu}{2 \max \left\{ \frac{L}{1-p}, \frac{\lambda}{p} \right\}}$ and

$$\frac{2n\alpha\sigma^2}{\mu} = \frac{\sum_{i=1}^n \left(\frac{1}{1-p} \|\nabla f_i(x_i(\lambda))\|^2 + \frac{\lambda^2}{p} \|x_i(\lambda) - \bar{x}(\lambda)\|^2 \right)}{\max \left\{ \frac{L}{1-p}, \frac{\lambda}{p} \right\} \mu}.$$

Remark 10 (Full averaging not supported). Is a setup such that conditions of Theorem 8.4.3 are satisfied and the aggregation update (8.11) is identical to full averaging? This is equivalent requiring $0 < p < 1$ such that $\alpha\lambda = np$. However, we have $\alpha\lambda \leq \frac{\lambda}{2\mathcal{L}} \leq np$, which means that full averaging is not supported by our theory.

8.4.4 Optimizing the rate and communication

Let us find the parameters p and α which lead to the fastest rate, in terms of either iterations or communication rounds, to push the error within ε of the neighborhood³ from Theorem 8.4.3, i.e., to achieve

$$\mathbb{E} [\|x^k - x(\lambda)\|^2] \leq \varepsilon \|x^0 - x(\lambda)\|^2 + \frac{2n\alpha\sigma^2}{\mu}. \quad (8.12)$$

³In Section 8.5 we propose a variance reduced algorithm which is able to get rid of the neighborhood in the convergence result completely. In that setting, our goal will be to achieve $\mathbb{E} [\|x^k - x(\lambda)\|^2] \leq \varepsilon \|x^0 - x(\lambda)\|^2$.

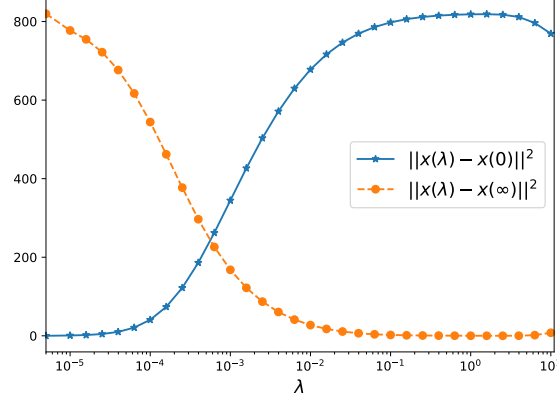


Figure 8.1: Distance of solution $x(\lambda)$ of (8.2) to pure local solution $x(0)$ and global solution $x(\infty)$ as a function of λ . Logistic regression on a1a dataset. See Appendix for experimental setup.

Corollary 8.4.4. *The value $p^* = \frac{\lambda}{L+\lambda}$ minimizes both the number of iterations and the expected number of communications for achieving (8.12). In particular, the optimal number of iterations is $2\frac{L+\lambda}{\mu} \log \frac{1}{\varepsilon}$, and the optimal expected number of communications is $\frac{2\lambda}{\lambda+L} \frac{L}{\mu} \log \frac{1}{\varepsilon}$.*

If we choose $p = p^*$, then $\frac{\alpha\lambda}{np} = \frac{1}{2}$, and the aggregation rule (8.11) in Algorithm 19 becomes

$$x_i^{k+1} = \frac{1}{2} (x_i^k + \bar{x}^k) \quad (8.13)$$

while the local GD step (8.10) becomes $x_i^{k+1} = x_i^k - \frac{1}{2L} \nabla f_i(x_i^k)$. Notice that while our method does not support full averaging as that is too unstable, (8.13) suggests that one should take a large step *towards* averaging.

As λ get smaller, the solution to the optimization problem (8.2) will increasingly favour pure local models, i.e., $x_i(\lambda) \rightarrow x_i(0) \stackrel{\text{def}}{=} \arg \min f_i$ for all i as $\lambda \rightarrow 0$. Pure local models can be computed without any communication whatsoever and Corollary 8.4.4 confirms this intuition: the optimal number of communication round decreases to zero as $\lambda \rightarrow 0$. On the other hand, as $\lambda \rightarrow \infty$, the optimal number of communication rounds converges to $2\frac{L}{\mu} \log \frac{1}{\varepsilon}$, which recovers the performance of GD for finding the globally optimal model (see Figure 8.1).

In summary, we recover the communication efficiency of GD for finding the globally optimal model as $\lambda \rightarrow \infty$. However, for other values of λ , the communication complexity of L2GD is better and decreases to 0 as $\lambda \rightarrow 0$. Hence, our communication complexity result interpolates between the communication complexity of GD for finding the global model and the zero communication complexity for finding the pure local models.

8.5 The L2SGD+ algorithm

As we have seen in Section 8.4.3, L2GD is a specific instance of SGD, thus only converges linearly to the neighborhood of the optimum. In this section, we resolve the mentioned

issue by incorporating control variates to the stochastic gradient [88, 37].

We also go further. We assume that each local objective has a finite-sum structure and propose an algorithm—L2SGD+—which takes *local stochastic* gradient steps, while maintaining (global) linear convergence rate. As a consequence, L2SGD+ is the first local SGD with linear convergence.⁴ For the reader's convenience, we present a variance reduced local gradient descent (i.e., no subsampling) in the Appendix.

8.5.1 Setup

Consider $f_i(x_i) = \frac{1}{m} \sum_{j=1}^m \textcolor{red}{f}_{i,j}(x_i)$. Therefore, the objective function (8.2) becomes

$$F(x) = \frac{1}{n} \sum_{i=1}^n \underbrace{\left(\frac{1}{m} \sum_{j=1}^m \textcolor{red}{f}_{i,j}(x_i) \right)}_{=f_i(x)} + \lambda \underbrace{\frac{1}{2n} \sum_{i=1}^n \|x_i - \bar{x}\|^2}_{=\Phi(x)}.$$

Assumption 8.5.1. Function $\textcolor{red}{f}_{i,j}$ is convex, \tilde{L} smooth while f_i is μ -strongly convex (for each $1 \leq j \leq m, 1 \leq i \leq n$).

Denote $\mathbf{1} \in \mathbb{R}^m$ to be vector of ones. We are now ready to state L2SGD+ as Algorithm 20.

Remark 11. L2SGD+ is the simplest local SGD method with variance reduction. In the Appendix, we present general L2SGD++ which allows for 1) an arbitrary number of data points per client and arbitrary local subsampling strategy, 2) partial participation of clients, and 3) local SVRG-like updates of control variates (thus potentially better memory). Lastly, L2SGD++ is able exploit the smoothness structure of the local objectives, resulting in tighter rates.

L2SGD+ only communicates when a two consecutive coin tosses land a different value, thus, on average $p(1-p)k$ times per k iterations. However, L2SGD+ requires communication of control variates $\mathbf{J}_i \mathbf{1}, \Psi_i$ as well – each communication round is thus three times more expensive. In the Appendix, we provide an implementation of L2SGD+ that does not require the communication of $\mathbf{J}_i \mathbf{1}, \Psi_i$.

8.5.2 Theory

We are now ready to present a convergence rate of L2SGD+.

Theorem 8.5.2. Let Assumption 8.5.1 hold and choose $\alpha = n \min \left\{ \frac{(1-p)}{4\tilde{L}+\mu m}, \frac{p}{4\lambda+\mu} \right\}$. Then the iteration complexity of Algorithm 20 is $\max \left\{ \frac{4\tilde{L}+\mu m}{(1-p)\mu}, \frac{4\lambda+\mu}{p\mu} \right\} \log \frac{1}{\varepsilon}$.

Next, we find a probability p that yields both the best iteration and communication complexity.

⁴We are aware that a linearly converging local SGD (with $\lambda = \infty$) might be obtained as a particular instance of the decoupling method from [139], although this was not stated in the mentioned paper. Other variance reduced local SGD algorithms [120, 92, 216] are not capable of achieving linear convergence.

Algorithm 20 L2SGD+: Loopless Local SGD with Variance Reduction

Input: $x_1^0 = \dots = x_n^0 \in \mathbb{R}^d$, stepsize α , probability p
 $\mathbf{J}_i^0 = 0 \in \mathbb{R}^{d \times m}$, $\Psi_i^0 = 0 \in \mathbb{R}^d$ (for $i = 1, \dots, n$)
for $k = 0, 1, 2, \dots$ **do**
 $\xi = 1$ with probability p and 0 with probability $1 - p$
 if $\xi = 0$ **then**
 All **Devices** $i = 1, \dots, n$:
 Sample $j \in \{1, \dots, m\}$ (uniformly at random)
 $g_i^k = \frac{1}{n(1-p)} \left(\nabla \textcolor{red}{f}_{i,j}(x_i^k) - (\mathbf{J}_i^k)_{:,j} \right) + \frac{\mathbf{J}_i^k \mathbf{1}}{nm} + \frac{\Psi_i^k}{n}$
 $x_i^{k+1} = x_i^k - \alpha g_i^k$
 Set $(\mathbf{J}_i^{k+1})_{:,j} = \nabla \textcolor{red}{f}_{i,j}(x_i^k)$, $\Psi_i^{k+1} = \Psi_i^k$,
 $(\mathbf{J}_i^{k+1})_{:,l} = (\mathbf{J}_i^k)_{:,l}$ for all $l \neq j$
 else
 Master computes the average $\bar{x}^k = \frac{1}{n} \sum_{i=1}^n x_i^k$
 Master does for all $i = 1, \dots, n$:
 $g_i^k = \frac{\lambda}{np} (x_i^k - \bar{x}^k) - \frac{p^{-1}-1}{n} \Psi_i^k + \frac{1}{nm} \mathbf{J}_i^k \mathbf{1}$
 Set $x_i^{k+1} = x_i^k - \alpha g_i^k$
 Set $\Psi_i^{k+1} = \lambda(x_i^k - \bar{x}^k)$, $\mathbf{J}_i^{k+1} = \mathbf{J}_i^k$
 end if
end for

Corollary 8.5.3. *Both communication and iteration complexity of L2SGD+ are minimized for $p = \frac{4\lambda + \mu}{4\lambda + 4\bar{L} + (m+1)\mu}$. The resulting iteration complexity is $\left(4\frac{\lambda}{\mu} + 4\frac{\bar{L}}{\mu} + m + 1\right) \log \frac{1}{\epsilon}$, while the communication complexity is $\frac{4\lambda + \mu}{4\bar{L} + 4\lambda + (m+1)\mu} \left(4\frac{\bar{L}}{\mu} + m\right) \log \frac{1}{\epsilon}$.*

Note that with $\lambda \rightarrow \infty$, the communication complexity of L2SGD+ tends to $\left(4\frac{\bar{L}}{\mu} + m\right) \log \frac{1}{\epsilon}$, which is communication complexity of minibatch SAGA to find the globally optimal model (see Chapter 5). On the other hand, in the pure local setting ($\lambda = 0$), the communication complexity becomes $\log \frac{1}{\epsilon}$ – this is because the Lyapunov function involves a term that measures the distance of local models, which requires communication to be estimated.

8.6 Experiments

In this section, we numerically verify the theoretical claims from this chapter. In all experiments in this chapter, we consider a simple binary classification model – logistic regression. In particular, suppose that device i owns data matrix $\mathbf{A}_i \in \mathbb{R}^{m \times d}$ along with corresponding labels $b_i \in \{-1, 1\}^m$. The local objective for client i is then given as follows

$$f_i(x) \stackrel{\text{def}}{=} \frac{1}{m} \sum_{j=1}^m \textcolor{red}{f}_{i,j}(x) + \frac{\mu}{2} \|x\|^2, \quad \text{where } \textcolor{red}{f}_{im+j}(x) \stackrel{\text{def}}{=} \log(1 + \exp((\mathbf{A}_i)_{j,:} x \cdot b_i)).$$

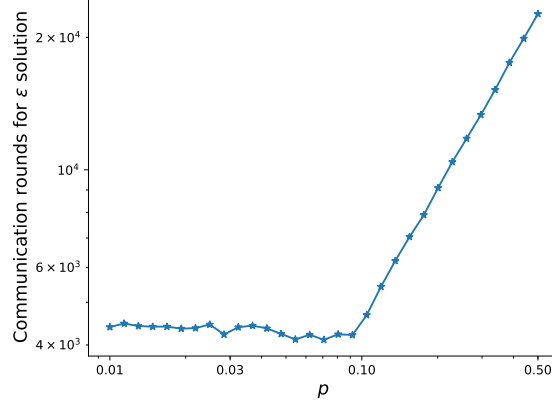


Figure 8.2: Communication rounds to get $\frac{F(x^k) - F(x^*)}{F(x^0) - F(x^*)} \leq 10^{-5}$ as a function of p with $p^* \approx 0.09$ (for L2SGD+). Logistic regression on a1a dataset with $\lambda = 0.1$; details in the Appendix.

The rows of data matrix \mathbf{A} were normalized to have length 4 so that each $f_{i,j}$ is 1-smooth for each j . At the same time, the local objective on each device is 10^{-4} -strongly convex. Next, datasets are from LibSVM [23].

In each case, we consider the simplest locally stochastic algorithm. In particular, each dataset is evenly split among the clients, while the local stochastic method samples a single data point each iteration.

We have chosen a different number of clients for each dataset – so that we cover different possible scenarios. See Table 8.1 for details (it also includes sizes of the datasets). Lastly, the stepsize was always chosen according to Theorem 8.5.2.

Table 8.1: Setup for the experiments.

Dataset	N $= nm$	d	n	m	μ	L	p 8.6.1	λ 8.6.2	p 8.6.3
a1a	1 605	123	5	321	10^{-4}	1	0.1	0.1	0.1
mushrooms	8 124	112	12	677	10^{-4}	1	0.1	0.05	0.3
phishing	11 055	68	11	1 005	10^{-4}	1	0.1	0.1	0.001
madelon	2 000	500	50	40	10^{-4}	1	0.1	0.02	0.05
duke	44	7 129	4	11	10^{-4}	1	0.1	0.4	0.1
gisette_scale	6 000	5 000	100	60	10^{-4}	1	0.1	0.2	0.003
a8a	22 696	123	8	109	10^{-4}	1	0.1	0.1	0.1

8.6.1 Comparison of the methods

In our first experiment, we verify two phenomena:

- Effect of variance reduction on the convergence speed of local methods. We compare 3 different methods: local SGD with full variance reduction (Algorithm 20), shifted local SGD (Algorithm 64) and local SGD (Algorithm 63). Our theory predicts that a fully variance reduced algorithm converges to the global optimum linearly, while both shifted local SGD and local SGD converge to a neighborhood of the optimum. At the same time, the neighborhood should be smaller for shifted local SGD.
- The claim that heterogeneity of the data does not influence the convergence rate. We consider two splits of the data heterogeneous and homogenous. For the homogenous split, we first randomly reshuffle the data and then construct the local objectives according to the current order (i.e., the first client owns the first m indices, etc.). For heterogeneous split, we first sort the data based on the labels and then construct the local objectives accordingly (thus achieving the worst-case heterogeneity). Note that the overall objective to solve is different in homogenous and heterogeneous case – we thus plot relative suboptimality of the objective (i.e., $\frac{F(x^k) - F(x^*)}{F(x^0) - F(x^*)}$) to directly compare the convergence speed.

In all cases, we choose $p = 0.1$ and $\lambda = \frac{1}{9}$ – such choice mean that p is very close to optimal. The other parameters (i.e. number of clients) are provided in Table 8.1. Figure 8.3 presents the result.

As expected, Figure 8.3 clearly demonstrates the following:

- Full variance reduction always converges to the global optima, methods with partial variance reduction only converge to a neighborhood of the optimum.
- Partial variance reduction (i.e., shifting the local SGD) is better than not using control variates at all. Although the improvement in the performance is rather negligible.
- Data heterogeneity does not affect the convergence speed of the proposed methods. Therefore, unlike standard local SGD, mixing the local and global models does not suffer the problems with heterogeneity.

8.6.2 Effect of p

In the second experiment, we study the effect of p on the convergence rate of variance reduced local SGD. Note that p immediately influences the number of communication rounds – on average, the clients take $(p^{-1} - 1)$ local steps in between two consecutive rounds of communication (aggregation).

In Section 8.5, we argue that, it is optimal (in terms of the convergence rate) to choose p of order $p^* \stackrel{\text{def}}{=} \frac{\lambda}{L + \lambda}$. Figure 8.4 compares $p = p^*$ against other values of p and confirms its optimality (in terms of optimizing the convergence rate).

While the slower convergence of Algorithm 20 with $p < p^*$ is expected (i.e., communicating more frequently yields a faster convergence), slower convergence for $p > p^*$ is rather surprising; in fact, it means that communicating less frequently yields faster convergence. This effect takes place due to the specific structure of problem (8.2); it would be lost when enforcing $x_1 = \dots = x_n$ (corresponding to $\lambda = \infty$).

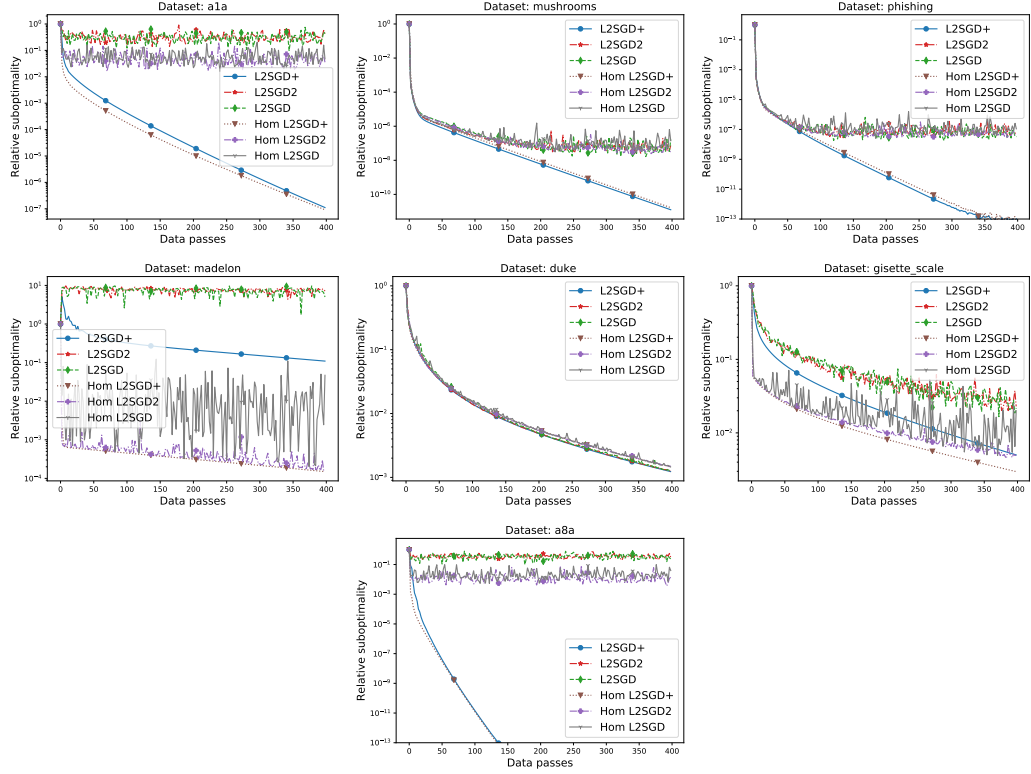


Figure 8.3: Variance reduced local SGD (Algorithm 20), shifted local SGD (Algorithm 64) and local SGD (Algorithm 63) applied on LibSVM problems for both homogenous split of data and Heterogenous split of the data. Stepsize for non-variance reduced method was chosen the same as for the analogous variance reduced method.

8.6.3 Effect of λ

In this experiment we study how different values of λ influence the convergence rate of Algorithm 20, given that everything else (i.e. p) is fixed. Note that for each value of λ we get a different instance of problem (8.2); thus the optimal solution is different as well. Therefore, in order to make a fair comparison between convergence speeds, we plot the relative suboptimality (i.e. $\frac{F(x^k) - F(x^*)}{F(x^0) - F(x^*)}$) against the data passes. Figure 8.5 presents the results.

The complexity of Algorithm 20 is⁵ $\mathcal{O}\left(\frac{\bar{L}}{(1-p)\mu}\right) \log \frac{1}{\varepsilon}$ as soon as $\lambda < \lambda^* \stackrel{\text{def}}{=} \frac{Lp}{(1-p)}$; otherwise the complexity is $\mathcal{O}\left(\frac{\lambda}{p\mu}\right) \log \frac{1}{\varepsilon}$. This perfectly consistent with what Figure 8.5 shows – the choice $\lambda < \lambda^*$ resulted in comparable convergence speed than $\lambda = \lambda^*$; while the choice $\lambda > \lambda^*$ yields noticeably worse rate than $\lambda = \lambda^*$.

⁵Given that μ is small.

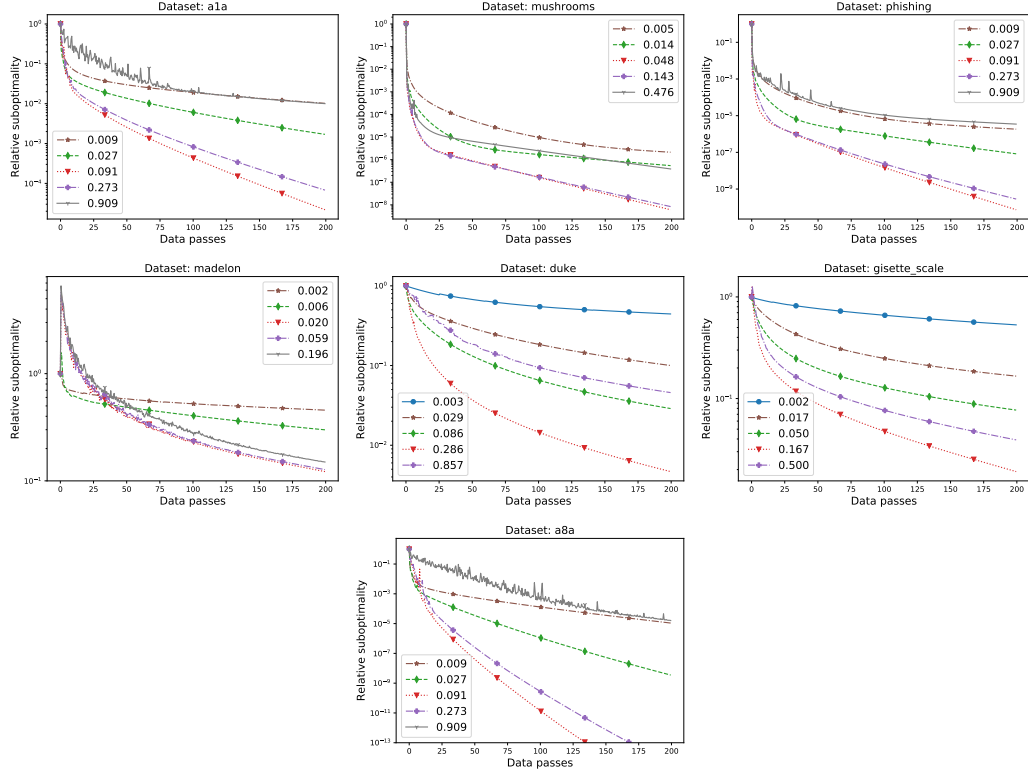


Figure 8.4: Effect of the aggregation probability p (legend of the plots) on the convergence rate of Algorithm 20. Choice $p = p^*$ corresponds to red dotted line with triangle marker. Parameter λ was chosen in each case as Table 8.1 indicates.

8.7 Conclusion

In this chapter we have proposed a new optimization formulation for federated learning. The algorithms (i.e., L2GD) we propose to solve the new formulation are similar the classical local SGD, however, the rates we have provided are superior to classical local SGD analysis.

Our analysis of L2GD can be extended to cover smooth convex and non-convex loss functions f_i (we do not explore these directions). Further, our methods can be extended to a decentralized regime where the devices correspond to devices of a connected network, and communication is allowed along the edges of the graph only. This can be achieved by introducing an additional randomization over the penalty Φ . Further, our approach can be accelerated in the sense of Nesterov [154] by adapting the results from [4, 164] to our setting, thus further reducing the number of communication rounds.

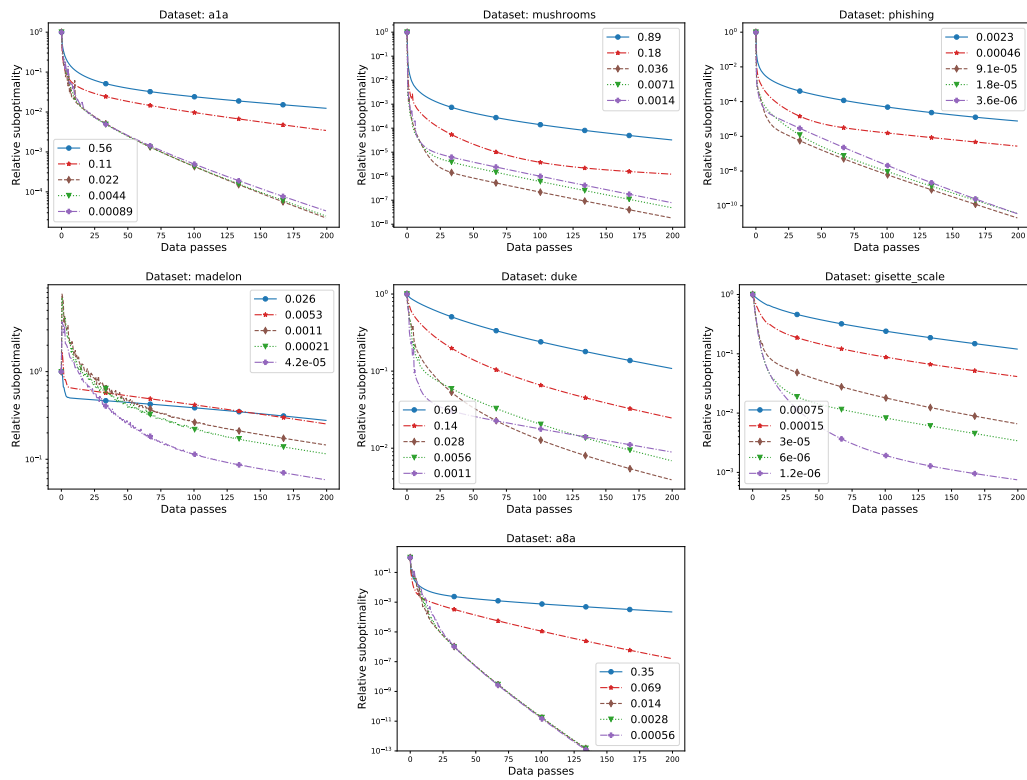


Figure 8.5: Effect of parameter λ (legend of the plot) on the convergence rate of Algorithm 20. The choice $\lambda = \lambda^*$ corresponds to brown dash-dotted line with diamond marker (the third one from the legend). Aggregation probability p was chosen in each case as Table 8.1 indicates.

Chapter 9

Stochastic Subspace Cubic Newton Method

In this chapter we consider a regularized not necessarily finite-sum optimization problem

$$\min_{x \in \mathbb{R}^d} \left\{ F(x) \stackrel{\text{def}}{=} f(x) + \psi(x) \right\}, \quad (9.1)$$

where $f : \mathbb{R}^d \rightarrow \mathbb{R}$ is convex and twice differentiable and $\psi : \mathbb{R}^d \rightarrow \mathbb{R} \cup \{+\infty\}$ is a proximable convex function. We are interested in the regime where the dimension d is very large, which arises in many contexts, such as the training of modern over-parameterized machine learning models. In this regime, coordinate descent (CD) methods, or more generally subspace descent methods, are the methods of choice.

9.1 Subspace descent methods

Subspace descent methods rely on update rules of the form

$$x^+ = x + \mathbf{S}h, \quad \mathbf{S} \in \mathbb{R}^{d \times \tau(\mathbf{S})}, \quad h \in \mathbb{R}^{\tau(\mathbf{S})}, \quad (9.2)$$

where \mathbf{S} is a thin matrix, typically with a negligible number of columns compared to the dimension (i.e., $\tau(\mathbf{S}) \ll d$). That is, they move from x to x^+ along the subspace spanned by the columns of \mathbf{S} .

In these methods, the subspace matrix \mathbf{S} is typically chosen first, followed by the determination of the parameters h which define the linear combination of the columns determining the update direction. Several different rules have been proposed in the literature for choosing the matrix \mathbf{S} , including greedy, cyclic and randomized rules. In this work we consider a *randomized* rule. In particular, we assume that \mathbf{S} is sampled from an arbitrary but fixed distribution \mathcal{D} restricted to requiring that \mathbf{S} be of full column rank¹ with probability one.

Once $\mathbf{S} \sim \mathcal{D}$ is sampled, a rule for deciding the stepsize h varies from algorithm to algorithm, but is mostly determined by the underlying *oracle model* for information access to function f . For instance, first-order methods require access to the subspace gradient $\nabla_{\mathbf{S}} f(x) \stackrel{\text{def}}{=} \mathbf{S}^\top \nabla f(x)$, and are relatively well studied [152, 199, 173, 215, 109]. At the other extreme are variants performing a full subspace minimization, i.e., f is minimized over the affine subspace given by $\{x + \mathbf{S}h \mid h \in \mathbb{R}^{\tau(\mathbf{S})}\}$ [24]. In particular, in this chapter

¹It is rather simple to extend our results to matrices \mathbf{S} which are column-rank deficient. However, this would introduce a rather heavy notation burden which we decided to avoid for the sake of clarity and readability.

we are interested in the *second-order* oracle model; i.e. we claim access both to the subspace gradient $\nabla_S f(x)$ and the subspace Hessian $\nabla_S^2 f(x) \stackrel{\text{def}}{=} \mathbf{S}^\top \nabla^2 f(x) \mathbf{S}$.

9.2 Contributions

We now summarize our contributions:

- **New 2nd order subspace method.** We propose a new stochastic subspace method—Stochastic Subspace Cubic Newton (SSCN)—constructed by minimizing an oracle-consistent global upper bound on the objective f in each iteration (Section 9.4). This bound is formed using both the subspace gradient and the subspace Hessian at the current iterate and relies on Lipschitzness of the subspace Hessian.
- **Interpolating global rate.** We prove (Section 9.6) that SSCN enjoys a global convergence rate that interpolates between the rate of stochastic CD and the rate of cubic regularized Newton as one varies the expected dimension of the subspace, $\mathbb{E}[\tau(\mathbf{S})]$.
- **Fast local rate.** Remarkably, we establish a local convergence bound for SSCN (Section 9.7) that matches the rate of stochastic subspace descent (SSD) [61] applied to solving the problem

$$\min_{x \in \mathbb{R}^d} \frac{1}{2} (x - x^*)^\top \nabla^2 f(x^*) (x - x^*), \quad (9.3)$$

where x^* is the solution of (9.1). Thus, SSCN behaves as *if* it had access to a perfect second-order model of f at the optimum, and was given the (intuitively much simpler) task of minimizing this model instead. Furthermore, note that SSD [61] applied to minimize a convex quadratic can be interpreted as doing an exact subspace search in each iteration, i.e., it minimizes the objective exactly along the active subspace [176]. Therefore, the local rate of SSCN matches the rate of the greediest strategy for choosing h in the active subspace, and as such, this rate is the best one can hope for a method that does not incorporate some form of acceleration.

- **Special cases.** We discuss in Section 9.4.2 how SSCN reduces to several existing stochastic second-order methods in special cases, either recovering the best known rates, or improving upon them. This includes SDSA [62], CN [68, 156] and RBCN [43]. However, our method is more general and hence allows for more applications.

We discuss more remotely related literature in Section 9.5. We now give a simple example of our setting.

Example 6 (Coordinate subspace setup). Let $\mathbf{I}^d \in \mathbb{R}^{d \times d}$ be the identity and let S be a random subset of $\{1, 2, \dots, d\}$. Given that $\mathbf{S} = \mathbf{I}_{(:,S)}^d$ with probability 1, the oracle model reveals $(\nabla f(x))_S$ and $(\nabla^2 f(x))_{(S,S)}$. Therefore, we have access to a random block of partial derivatives of f and a block submatrix of its Hessian, both corresponding to the subset of indices S . Furthermore, the rule (9.2) updates a subset S of coordinates only. In this setting, our method is a new *second-order coordinate subspace descent* method.

9.3 Preliminaries

Throughout the chapter, we assume that f is convex, twice differentiable, and sufficiently smooth and that ψ is convex, albeit possibly non-differentiable, as the next assumption states.²

Assumption 9.3.1. *Function $f : \mathbb{R}^d \rightarrow \mathbb{R}$ is convex and twice differentiable with M -Lipschitz continuous Hessian. Function $\psi : \mathbb{R}^d \rightarrow \mathbb{R} \cup \{+\infty\}$ is proper closed and convex.*

We always assume that a minimum of F exists and by x^* denote any of its minimizers. We let $F^* \stackrel{\text{def}}{=} F(x^*)$.

Since our method always takes steps along random subspaces spanned by the columns of $\mathbf{S} \in \mathbb{R}^{d \times \tau(\mathbf{S})}$, it is reasonable to define the Lipschitzness of the Hessian over the range of \mathbf{S} :³

$$M_{\mathbf{S}} \stackrel{\text{def}}{=} \max_{x \in \mathbb{R}^d} \max_{\substack{h \in \mathbb{R}^{\tau(\mathbf{S})} \\ h \neq 0}} \frac{|\nabla^3 f(x)[\mathbf{S}h]^3|}{\|\mathbf{S}h\|^3}. \quad (9.4)$$

As the next lemma shows, the maximal value of $M_{\mathbf{S}}$ for any \mathbf{S} of width τ can be up to $(\frac{d}{\tau})^{\frac{3}{2}}$ times smaller than M and this will lead to a tighter approximation of the objective.

Lemma 9.3.2. *We have $M \geq \max_{\tau(\mathbf{S})=\tau} M_{\mathbf{S}}$. Moreover, there is a problem where $\max_{\tau(\mathbf{S})=\tau} M_{\mathbf{S}} = (\frac{\tau}{d})^{\frac{3}{2}} M$. Lastly, if $\text{Range}(\mathbf{S}) = \text{Range}(\mathbf{S}')$, then $M_{\mathbf{S}} = M_{\mathbf{S}'}$.*

The next lemma provides a direct motivation for our algorithm. It gives a global upper bound on the objective over a random subspace, given the first and second-order information at the current point.

Lemma 9.3.3. *Let $x \in \mathbb{R}^d$, $\mathbf{S} \in \mathbb{R}^{d \times \tau(\mathbf{S})}$, $h \in \mathbb{R}^{\tau(\mathbf{S})}$ and x^+ be as in (9.2). Then*

$$\left| f(x^+) - f(x) - \langle \nabla_{\mathbf{S}} f(x), h \rangle - \frac{1}{2} \langle \nabla_{\mathbf{S}}^2 f(x) h, h \rangle \right| \leq \frac{M_{\mathbf{S}}}{6} \|\mathbf{S}h\|^3. \quad (9.5)$$

As a consequence, we have

$$F(x^+) \leq F(x) + T_{\mathbf{S}}(x, h), \quad (9.6)$$

where $T_{\mathbf{S}}(x, h) \stackrel{\text{def}}{=} \langle \nabla_{\mathbf{S}} f(x), h \rangle + \frac{1}{2} \langle \nabla_{\mathbf{S}}^2 f(x) h, h \rangle + \frac{M_{\mathbf{S}}}{6} \|\mathbf{S}h\|^3 + \psi(x + \mathbf{S}h)$.

We shall also note that for function ψ we require *separability* with respect to the sampling distribution (see Definition 9.6.5 and the corresponding Assumption 9.6.6 in Section 9.6.1).

For better orientation throughout the chapter, we provide a table of frequently used notation in the Appendix.

²We will also require separability of ψ ; see Section 9.6.1.

³By $\|x\| \stackrel{\text{def}}{=} \langle x, x \rangle^{1/2}$ we denote the standard Euclidean norm.

9.4 The SSCN algorithm

For a given \mathbf{S} and current iterate x^k , it is a natural idea to choose h as a minimizer of the upper bound (9.6) in h for $x = x^k$, and subsequently set $x^{k+1} = x^+$ via (9.2). Note that we are choosing \mathbf{S} randomly according to a fixed distribution \mathcal{D} (with a possibly random number of columns). We have just described SSCN—Stochastic Subspace Cubic Newton—formally stated as Algorithm 21.

Algorithm 21 SSCN: Stochastic Subspace Cubic Newton

- 1: **Initialization:** x^0 , distribution \mathcal{D} of random matrices with d rows and full column rank
 - 2: **for** $k = 0, 1, 2, \dots$ **do**
 - 3: Sample \mathbf{S} from distribution \mathcal{D}
 - 4: $h^k = \arg \min_{h \in \mathbb{R}^{\tau(\mathbf{S})}} T_{\mathbf{S}}(x^k, h)$
 - 5: Set $x^{k+1} = x^k + \mathbf{S}h^k$
 - 6: **end for**
-

Remark 12. Inequality (9.6) becomes an equality with $h = 0$. As a consequence, we must have $F(x^{k+1}) \leq F(x^k)$, and thus the sequence $\{F(x^k)\}_{k \geq 0}$ is non-increasing.

9.4.1 Solving the subproblem

Algorithm 21 requires $T_{\mathbf{S}}$ to be minimized in h each iteration. As this operation does not have a closed-form solution in general, it requires an optimization subroutine itself of a possibly non-trivial complexity, which we discuss here.

The subproblem without ψ . Let us now consider the case when $\psi(x) \equiv 0$ in which our problem (9.1) does not contain any nondifferentiable components. Various techniques for minimizing regularized quadratic functions were developed during the development of Trust-region methods (see [26]), and applied to Cubic regularization in [156]. The classical approach consists in performing some diagonalization of the matrix $\nabla_{\mathbf{S}}^2 f(x)$ first, by computing the *eigenvalue* or *tridiagonal* decomposition, which costs $\mathcal{O}(\tau(\mathbf{S})^3)$ arithmetical operations. Then, to find the minimizer, it merely remains to solve a one-dimensional nonlinear equation (this part can be done by $\tilde{\mathcal{O}}(1)$ iterations of the one-dimensional Newton method, with a linear cost per step). More details and analysis of this procedure can be found in [56].

The next example gives a setting in which an explicit formula for the minimizer of $T_{\mathbf{S}}$ can be deduced.

Example 7. Let e_i be the i th unit basis vector in \mathbb{R}^d . If $\mathbf{S} \in \{e_1, \dots, e_d\}$ with probability 1 and $\psi(x) = 0$, the update rule can be written as $x^{k+1} = x^k - \alpha_i^k e_i$, with

$$\alpha_i^k = \frac{2\nabla_i f(x^k)}{\nabla_i^2 f(x^k) + \sqrt{(\nabla_{ii}^2 f(x^k))^2 + 2M_{e_i} |\nabla_i f(x^k)|}},$$

thus the cost of solving the subproblem is $\mathcal{O}(1)$.

Subproblem with simple ψ . In some scenarios, minimization of T_S can be done using a simple algorithm if ψ is simple enough. We now give an example of this.

Example 8. If $S \in \{e_1, \dots, e_d\}$ with probability 1, the subproblem can be solved using a binary search given that the evaluation of ψ is cheap. In particular, if we can evaluate $\psi(x^k + Sh) - \psi(x^k)$ in $\tilde{O}(1)$, the cost of solving the subproblem will be $\tilde{O}(1)$.

The subproblem with general ψ . In the case of general regularizers, recent line of work [17] explores to the use of *first-order* optimization methods (Gradient Methods) for computing an approximate minimizer of T_S . We note that the backbone of such Gradient Methods is an implementation of the following operation (for a any given vector $b \in \mathbb{R}^{\tau(S)}$, and positive scalars α, β):

$$\arg \min_{h \in \mathbb{R}^{\tau(S)}} \langle b, h \rangle + \frac{\alpha}{2} \|Sh\|^2 + \frac{\beta}{3} \|Sh\|^3 + \psi(x^k + Sh).$$

To the best of our knowledge, the most efficient gradient method is the Fast Gradient Method (FGM)[155], achieving an $\mathcal{O}(1/k^6)$ convergence rate. However, FGM can deal with any ψ as long as the above subproblem is cheap to solve. We shall also note that gradient methods do not require a storage of $\nabla_S^2 f(x)$; but rather iteratively access partial Hessian-vector products $\nabla_S^2 f(x)h$.

Line search. Note that in Algorithm 21 we use the Lipschitz constants M_S of the subspace Hessian (see Definition (9.4)) as the regularization parameters. In many application, M_S can be estimated cheaply (see Section 9.8). In general, however, M_S might be unknown or hard to estimate. In such a case, one might use a simple one-dimensional search on each iteration: multiply the estimate of M_S by the factor of two until the bound (9.6) is satisfied, and divide it by two at the start of each iteration. Note that the average number of such line search steps per iteration can be bounded by two (see [66] for the details).

9.4.2 Special cases

There are several scenarios where SSCN becomes an already known algorithm. We list them below.

Quadratic minimization. If $M = 0$ and $\psi = 0$, SSCN reduces to the stochastic dual subspace ascent (SDSA) method [62], first analyzed in an equivalent primal form as a *sketch-and-project* method in [61]. In such a case, SSCN performs both first-order, second-order updates, and exact minimization over a subspace at the same time due to the quadratic structure of the objective [176]. The convergence rate we provide in Section 9.7 exactly matches the rate of sketch-and-project as well. As a consequence, we recover a subclass of matrix inversion algorithms [63] together with stochastic spectral (coordinate) descent [108] along with their convergence theory.

Full-space method. If $S = \mathbf{I}^d$ with probability 1, SSCN reduces to cubically regularized Newton (CN) [68, 156]. In this case, we recover both existing global convergence rates and superlinear local convergence rates.

Separable non-quadratic part of f . The RBCN method [43] aims to minimize (9.1) with $f(x) = g(x) + \phi(x)$, where g, ϕ are both convex, and ϕ is separable.⁴ They assume that $\nabla^2 g(x) \preceq \mathbf{A} \in \mathbb{R}^{d \times d}, \forall x \in \mathbb{R}^d$, while ϕ has Lipschitz continuous Hessian. In each iteration, RBCN constructs an upper bound on the objective using first-order information from g only. This is unlike SSCN, which uses second-order information from g . In a special case when $\nabla^2 g(x) = \mathbf{A}$ for all x , SSCN and RBCN are identical algorithms. However, RBCN is less general: it requires separable ϕ , and thus does not cover some of our applications, and takes directions along coordinates only. Further, the rates we provide are better even in the setting where the two methods coincide ($\nabla^2 g(x) = \mathbf{A}$). The simplest way to see that is by looking at local convergence – RBCN does not achieve the local convergence rate of block CD to minimize (9.3), which is the best one might hope for.

Besides these particular cases, for a general twice-differentiable f , SSCN is a new second-order method.

9.5 Related literature

Several methods in the literature are related to SSCN. We briefly review them below.

- *Cubic regularization of Newton method* was proposed first in [68], and received substantial attention after the work of [156], where its global complexity guarantees were established. During the last decade, there was a steady increase of research in second-order methods, discovering Accelerated [151, 141], Adaptive [18, 19], and Universal [66, 67, 42] schemes (the latter ones are adjusting automatically to the smoothness properties of the objective).
- There is a vast literature on *first-order coordinate descent* (CD) methods. While CD with $\tau = 1$ is consistently the same method within the literature [152, 173, 215], there are several ways to deal with $\tau > 1$. The first approach constructs a separable upper bound on the objective (in expectation) in the direction of a random subset of coordinates [166, 167], which is minimized to obtain the next iterate. The second approach—SDNA [168]—works with a tighter non-separable upper bound. SDNA is, therefore, more costly to implement but requires a smaller number of iterations to converge. The literature on first-order subspace descent algorithms is slightly less rich, the notable examples are random pursuit [199] or stochastic subspace descent [109].
- *Randomized subspace Newton* (RSN) [59] is a method of the form

$$x^{k+1} = x^k - \hat{L}^{-1} \mathbf{S} (\nabla_{\mathbf{S}}^2 f(x^k))^{-1} \nabla_{\mathbf{S}} f(x^k)$$

⁴Separability is defined in Section 9.6.1.

for some specific fixed \hat{L} . In particular, it can be seen as a method minimizing the following upper bound on the function, which follows from their assumption:

$$h^k = \arg \min_h \langle \nabla_{\mathbf{S}} f(x^k), h \rangle + \frac{\hat{L}}{2} \langle \nabla_{\mathbf{S}}^2 f(x^k) h, h \rangle.$$

This is followed by an update over the subspace: $x^{k+1} = x^k + \mathbf{S}h^k$. Since both RSN and SSCN are analyzed under different assumptions, the global linear rates are not directly comparable. However, the local rate of SSCN is superior to RSN. We shall also note that RSN is a stochastic subspace version of a method from [94].

- *Subsampled Newton* (SN) methods [15, 25, 219, 181] and *subsampled cubic regularized Newton methods* [98, 218, 211] and *stochastic (cubic regularized) Newton methods* [205, 20, 107] are stochastic second-order algorithms to tackle finite sum minimization. Their major disadvantage is a requirement of an immense sample size, which makes them often impractical if used as theory prescribes. A notable exception that does not require a large sample size was recently proposed in [107]. However, none of these methods are directly comparable to SSCN as they are not subspace descent methods, but rather randomize over data points (or sketch the Hessian from “inside” [163]).

9.6 Global complexity bounds

We first start presenting the global complexity results of SSCN.

9.6.1 Setup

Throughout this section, we require some kind of uniformity of the distribution \mathcal{D} over subspaces given by \mathbf{S} . In particular, we require $\mathbf{Z} = \mathbf{Z}(\mathbf{S}) \stackrel{\text{def}}{=} \mathbf{S}(\mathbf{S}^\top \mathbf{S})^{-1} \mathbf{S}^\top$, the projection matrix onto the range of \mathbf{S} , to be a scalar multiple of identity matrix in expectation.

Assumption 9.6.1. $\exists \tau > 0$ such that distribution \mathcal{D} satisfies

$$\mathbb{E}[\mathbf{Z}] = \frac{\tau}{d} \mathbf{I}^d. \quad (9.7)$$

A direct consequence of Assumption 9.6.1 is that τ is an expected width of \mathbf{S} , as the next lemma states.

Lemma 9.6.2. *If Assumption 9.6.1 holds, then $\mathbb{E}[\tau(\mathbf{S})] = \tau$.*

As mentioned before, the global complexity results are interpolating between convergence rate of (first-order) CD and (global) convergence rate of Cubic Newton. However, first-order CD requires Lipschitzness of gradients, and thus we will require it as well.

Assumption 9.6.3. *Function f has L -Lipschitz continuous gradients, i.e. $\nabla^2 f(x) \preceq L \mathbf{I}^d$ for all $x \in \mathbb{R}^d$.*

We will also need an extra assumption on ψ . It is well known that proximal (first-order) CD with fixed step size does not converge if ψ is not separable – in such case, even if $f(x^k) = f(x^*)$ we might have $f(x^{k+1}) > f(x^*)$. Therefore, we might not hope that SSCN will converge without additional assumptions on ψ . Informally speaking, separability of ψ with respect to directions given by columns of \mathbf{S} is required. To define it formally, let us introduce first the notion of a separable set.

Definition 9.6.4. Set $Q \subseteq \mathbb{R}^d$ is called D -separable, if $\forall x, y \in Q, \mathbf{S} \in D$:

$$\mathbf{Z}x + (\mathbf{I}^d - \mathbf{Z})y \in Q.$$

Let $e \in \mathbb{R}^d$ be the vector of all ones. Then, for arbitrary functions, we have

Definition 9.6.5. Function $\phi : \mathbb{R}^d \rightarrow \mathbb{R} \cup \{+\infty\}$ is D -separable if $\text{dom } \phi$ is D -separable, and there is map $\phi' : \text{dom } \phi \rightarrow \mathbb{R}^d$ such that

1. $\forall x \in \text{dom } \phi : \phi(x) = \langle \phi'(x), e \rangle,$
2. $\forall x, y \in \text{dom } \phi, \mathbf{S} \in D : \phi'(\mathbf{Z}x + (\mathbf{I}^d - \mathbf{Z})y) = \mathbf{Z}\phi'(x) + (\mathbf{I}^d - \mathbf{Z})\phi'(y).$

Example 9. If D is a set of matrices whose columns are standard basis vectors, D -separability reduces to classical (coordinate-wise) separability.

Example 10. If D is set of matrices which are column-wise submatrices of orthogonal \mathbf{U} , D -separability of ϕ reduces to classical coordinate-wise separability of $\phi(\mathbf{U}^\top x)$.

Example 11. $\phi(x) = \frac{1}{2}\|x\|^2$ is D -separable for any D .

Assumption 9.6.6. Function ψ is $\text{Range}(D)$ -separable.

We are now ready to present the convergence rate of SSCN.

9.6.2 Theory

First, let us introduce the critical lemma from which the main global complexity results are derived. Our first lemma gives a bound on the expected progress after a single step of SSCN.

Lemma 9.6.7. Let Assumptions 9.3.1, 9.6.1, 9.6.3 and 9.6.6 hold. Then, for every $k \geq 0$ and $y \in \mathbb{R}^d$ we have

$$\mathbb{E}[F(x^{k+1}) | x^k] \leq \left(1 - \frac{\tau}{d}\right) F(x^k) + \frac{\tau}{d} F(y) + \frac{\tau}{d} \left(\frac{d - \tau L}{d} \frac{L}{2} \|y - x^k\|^2 + \frac{M}{3} \|y - x^k\|^3 \right). \quad (9.8)$$

Now we are ready to present global complexity results for the general class of convex functions. The convergence rate is obtained by summing (9.8) over the different iterations k , and with a specific choice of y .

Theorem 9.6.8. *Let Assumptions 9.3.1, 9.6.1, 9.6.3 and 9.6.6 hold. Denote*

$$R \stackrel{\text{def}}{=} \sup_{x \in \mathbb{R}^d} \left\{ \|x - x^*\| : F(x) \leq F(x^0) \right\}, \quad (9.9)$$

and suppose that $R < +\infty$. Then, for every $k \geq 1$ we have

$$\mathbb{E} [F(x^k)] - F^* \leq \frac{d - \tau}{\tau} \cdot \frac{4.5LR^2}{k} + \left(\frac{d}{\tau}\right)^2 \cdot \frac{9MR^3}{k^2} + \frac{F(x^0) - F^*}{1 + \frac{1}{4} \left(\frac{\tau}{d}k\right)^3}. \quad (9.10)$$

Note that convergence rate of the minibatch version⁵ of first-order CD is $\mathcal{O}\left(\frac{d}{\tau} \frac{LR^2}{k}\right)$. At the same time, (global) convergence rate of cubically regularized Newton method is $\mathcal{O}\left(\frac{MR^3}{k^2}\right)$. Therefore, Theorem 9.6.8 shows that the global rate of SSCN well interpolates between the two extremes, depending on the sample size τ we choose.

Remark 13. According to estimate (9.10), in order to have $\mathbb{E} [F(x^k)] - F^* \leq \varepsilon$, it is enough to perform

$$k = \mathcal{O} \left(\frac{d - \tau}{\tau} \frac{LR^2}{\varepsilon} + \frac{d}{\tau} \sqrt{\frac{MR^3}{\varepsilon}} + \frac{d}{\tau} \left(\frac{F(x^0) - F^*}{\varepsilon} \right)^{1/3} \right)$$

iterations of SSCN.

Next, we move to the strongly convex case.

Assumption 9.6.9. *Function f is μ -strongly convex, i.e. $\nabla^2 f(x) \succeq \mu \mathbf{I}^d$ for all $x \in \mathbb{R}^d$.*

Remark 14. Strong convexity of the objective (assumed for Theorem 9.6.10 later) implies: $R < +\infty$. Furthermore, due to monotonicity of the sequence $\{F(x_k)\}_{k \geq 0}$ (see Remark 12), we have $\|x^k - x^*\| \leq R$ for all k . Therefore, it is sufficient to require Lipschitzness of gradients over the sublevel set, which holds with $L = \lambda_{\max}(\nabla^2 f(x^*)) + MR$.

As both extremes cubic regularized Newton (where $\mathbf{S} = \mathbf{I}^d$ always) and (first-order) CD ($\mathbf{S} = e_i$ for randomly chosen i) enjoy (global) linear rate under strong convexity, linear convergence of SSCN is expected as well. At the same time, the leading complexity term should be in between the two extremes. Such a result is established as Theorem 9.6.10.

Theorem 9.6.10. *Let Assumptions 9.3.1, 9.6.1, 9.6.6 and 9.6.9 hold. Then, $\mathbb{E} [F(x^k)] - F^* \leq \varepsilon$, as long as the number of iterations of SSCN is*

$$k = \mathcal{O} \left(\left(\frac{d - \tau}{\tau} \frac{L}{\mu} + \frac{d}{\tau} \sqrt{\frac{MR}{\mu}} + \frac{d}{\tau} \right) \cdot \log \frac{F(x^0) - F^*}{\varepsilon} \right).$$

Indeed, if $\mathbf{S} = \mathbf{I}^d$ with probability 1 and $MR \geq \mu$, the leading complexity term becomes $\sqrt{\frac{MR}{\mu}} \log \frac{1}{\varepsilon}$ which corresponds to the global complexity of cubically regularized Newton for minimizing strongly convex functions [156]. On the other side of the spectrum

⁵Sampling τ coordinates at a time for objectives with L -Lipschitz gradients.

if $\mathbf{S} = e_i$ with probability $\frac{1}{d}$, the leading complexity term becomes $\frac{dL}{\mu} \log \frac{1}{\varepsilon}$, which again corresponds to convergence rate of CD [152]. Lastly, if $1 < \tau < d$, the global linear rate interpolates the rates mentioned above.

Remark 15. Proof of Theorem 9.6.10 only uses the following consequence of strong convexity:

$$\frac{\mu}{2} \|x - x^*\|^2 \leq F(x) - F^*, \quad x \in \mathbb{R}^d \quad (9.11)$$

and thus the conditions of Theorem 9.6.10 might be slightly relaxed.⁶ For detailed comparison of various relaxations of strong convexity, see [91].

9.7 Local convergence

Throughout this section, assume that $\psi = 0$. We first present the key descent lemma, which will be used to obtain local rates. Let $\mathbf{H}_{\mathbf{S}}(x) \stackrel{\text{def}}{=} \nabla_{\mathbf{S}}^2 f(x) + \sqrt{\frac{M_{\mathbf{S}}}{2}} \|\nabla_{\mathbf{S}} f(x)\|^{\frac{1}{2}} \mathbf{I}^{\tau(\mathbf{S})}$.

Lemma 9.7.1.

$$f(x^k) - f(x^{k+1}) \geq \frac{1}{2} \|\nabla_{\mathbf{S}} f(x^k)\|_{\mathbf{H}^{-1}(x^k)}^2. \quad (9.12)$$

Before stating the convergence theorem, it will be suitable to define the stochastic condition number of $\mathbf{H}_* \stackrel{\text{def}}{=} \nabla^2 f(x^*)$:

$$\zeta \stackrel{\text{def}}{=} \lambda_{\min} \left(\mathbf{H}_*^{\frac{1}{2}} \mathbb{E} \left[\mathbf{S} (\mathbf{S}^{\top} \mathbf{H}_* \mathbf{S})^{-1} \mathbf{S}^{\top} \right] \mathbf{H}_*^{\frac{1}{2}} \right), \quad (9.13)$$

as it will drive the local convergence rate of SSCN.

Theorem 9.7.2 (Local Convergence). *Let Assumptions 9.3.1, 9.6.9 hold, and suppose that $\psi = 0$. For any $\varepsilon > 0$ there exists $\delta > 0$ such that if $F(x^0) - F^* \leq \delta$, we have*

$$\mathbb{E} [F(x^k) - F^*] \leq (1 - (1 - \varepsilon)\zeta)^k (F(x^0) - F^*) \quad (9.14)$$

and therefore the local complexity of SSCN is $\mathcal{O}(\zeta^{-1} \log \frac{1}{\varepsilon})$. If further $M = 0$ (i.e. f is quadratic), then $\varepsilon = 0$ and $\delta = \infty$, and thus the rate is global.

The proof of Theorem 9.7.2 along with the exact formulas for ε, δ can be found in Section I.3 of the Appendix.

Theorem 9.7.2 provides a local linear convergence rate of SSCN. While one might expect a superlinear rate to be achievable, this is not the case, and we argue that the rate from Theorem 9.7.2 is the best one can hope for.

In particular, if $M = 0$, Algorithm 21 becomes subspace descent for minimizing positive definite quadratic which is a specific instance of sketch-and-project [61]. However, sketch-and-project only converges linearly – the iteration complexity of sketch-and-project to minimize $(x - x^*)^{\top} \mathbf{A} (x - x^*)$ with $\mathbf{A} \succ 0$ is

$$\mathcal{O} \left(\left(\mathbf{A}^{\frac{1}{2}} \mathbb{E} \left[\mathbf{S} (\mathbf{S}^{\top} \mathbf{A} \mathbf{S})^{-1} \mathbf{S}^{\top} \right] \mathbf{A}^{\frac{1}{2}} \right)^{-1} \log \frac{1}{\varepsilon} \right).$$

⁶However, this relaxation is not sufficient to obtain the local convergence results.

Notice that this rate is matched by Theorem 9.7.2 in this case.

Next, we compare the local rate of SSCN to the rate of SDNA [168]. To best of our knowledge, SDNA requires the least oracle calls to minimize f among all first-order non-accelerated methods.

Remark 16. SDNA is a first-order analogue to Algorithm 21 with $\mathbf{S} = \mathbf{I}_{(:,S)}^d$. In particular, given matrix \mathbf{L} such that $\mathbf{L} \succeq \nabla^2 f(x) \succ 0$ for all x , the update rule of SDNA is

$$x^+ = x - \mathbf{S} (\mathbf{S}^\top \mathbf{L} \mathbf{S})^{-1} \nabla_{\mathbf{S}} f(x),$$

where $\mathbf{S} = \mathbf{I}_{(:,S)}^d$ for a random subset of columns S . SDNA enjoys linear convergence rate with leading complexity term $(\mu \lambda_{\min} (\mathbb{E} [\mathbf{S} (\mathbf{S}^\top \mathbf{L} \mathbf{S})^{-1} \mathbf{S}^\top]))^{-1}$. The leading complexity term of SSCN is ζ^{-1} , and we can bound

$$\begin{aligned} \zeta &\geq \lambda_{\min}(\mathbf{H}_*) \lambda_{\min} \left(\mathbb{E} \left[\mathbf{S} (\mathbf{S}^\top \mathbf{H}_* \mathbf{S})^{-1} \mathbf{S}^\top \right] \right) \\ &\geq \mu \lambda_{\min} \left(\mathbb{E} \left[\mathbf{S} (\mathbf{S}^\top \mathbf{L} \mathbf{S})^{-1} \mathbf{S}^\top \right] \right). \end{aligned}$$

Hence, the local rate of SSCN is no worse than the rate of SDNA. Furthermore, both of the above inequalities might be very loose in some cases (i.e., there are examples where $\frac{\zeta}{\mu \lambda_{\min} \mathbb{E} [\mathbf{S} (\mathbf{L} \mathbf{S})^{-1} \mathbf{S}^\top]}$ can be arbitrarily high). Therefore, local convergence rate of SSCN might be arbitrarily better than the convergence rate of SDNA. As a consequence, the local convergence of SSCN is better than convergence rate of any non-accelerated first-order method.⁷

Lastly, the local convergence rate provided by Theorem 9.7.2 recovers the superlinear rate of cubic regularized Newton's method, as the next remark states.

Remark 17. If $\mathbf{S} = \mathbf{I}^d$ with probability 1, Algorithm 21 becomes cubic regularized Newton method [68, 156]. For $\mathbf{H}_* \stackrel{\text{def}}{=} \nabla^2 f(x^*)$ we have

$$\zeta = \lambda_{\min} \left(\mathbf{H}_*^{\frac{1}{2}} \mathbf{H}_*^{-1} \mathbf{H}_*^{\frac{1}{2}} \right) = \lambda_{\min}(\mathbf{I}^d) = 1.$$

As a consequence of Theorem 9.7.2, for any $\varepsilon > 0$ there exists $\delta > 0$ such that if $F(x) - F(x^*) \leq \delta$, we have

$$F(x^+) - F(x^*) \leq \varepsilon (F(x) - F(x^*)).$$

Therefore, we obtain a superlinear convergence rate.

⁷The rate of SSCN and rate of accelerated subspace descent methods are not directly comparable – while the (local) rate of SSCN might be better than rate of ACD, the reverse might happen as well. However, both ACD and SSCN are faster than non-accelerated subspace descent.

9.8 Applications

9.8.1 Linear models

Consider only $S = \mathbf{I}_{(:,S)}^d$ for simplicity. Let

$$F(x) \stackrel{\text{def}}{=} \frac{1}{n} \sum_{i=1}^n \phi_i(\langle a_i, x \rangle) + \psi(x), \quad (9.15)$$

and $f(x) \stackrel{\text{def}}{=} \frac{1}{n} \sum_{i=1}^n \phi_i(\langle a_i, x \rangle)$ and suppose that $|\nabla^3 \phi_i(y)| \leq c$. Then clearly, for any $h \in \mathbb{R}^d$, we have

$$\nabla^3 f(x)[h]^3 = \frac{1}{n} \sum_{i=1}^n \nabla^3 \phi_i(\langle a_i, x \rangle) \langle a_i, h \rangle^3.$$

While evaluating

$$E \stackrel{\text{def}}{=} \max_{\|h\|=1, x} \nabla^3 f(x)[h]^3$$

is infeasible, we might bound it instead via

$$E \leq \max_{\|h\|=1} \frac{c}{n} \sum_{i=1}^n |\langle a_i, h \rangle|^3 \leq \frac{c}{n} \sum_{i=1}^n \|a_i\|^3, \quad (9.16)$$

which means that $M = \frac{c}{n} \sum_{i=1}^n \|a_i\|^3$ is a feasible choice. On the other hand, for $S = \{j\}$ we have

$$\max_{\|h_j\|=1, x} \nabla^3 f(x)[h_j]^3 = \max_x \nabla^3 f(x)[e_j]^3 \leq \frac{c}{n} \sum_{i=1}^n |a_{ij}|^3$$

and thus we might set $M_j = \frac{c}{n} \sum_{i=1}^n |a_{ij}|^3$. The next lemma compares the above choices of M and M_j .

Lemma 9.8.1. *We have $M \geq \max_j M_j$. At the same time, there exist vectors a_i that $\max_j M_j = \frac{M}{d^{\frac{3}{2}}}$.*

Proof. The first part is trivial. For the second part, consider $a_{i,j} \in \{-1, 1\}$. □

Remark 18. One might avoid the last inequality from (9.16) using polynomial optimization; however, this might be more expensive than solving the original optimization problem and thus is not preferable. Another strategy would be to use a line search, see Section 9.4.1.

Both the formula for M and the formula for M_j require the prior knowledge of $c \geq 0$ such that $|\nabla^3 \phi_i(y)| \leq c$ for all i . The next Lemma shows how to compute such c for the logistic regression (binary classification model).

Lemma 9.8.2. *Let $\phi_i(y) = \log(1 + e^{-b_i y})$, $b_i \in \{-1, 1\}$. Then $c = \frac{1}{6\sqrt{3}}$.*

Proof. $\nabla^3 \phi_i(y) = -\frac{e^x(e^x-1)}{(1+e^x)^3} \Rightarrow |\nabla^3 \phi_i(y)| \leq \frac{1}{6\sqrt{3}}$. □

Cost of performing a single iteration For the sake of simplicity, let $\tau(\mathbf{S}) = 1$, $\psi \equiv 0$. Any CD method (i.e. method with update rule (9.2) with $\mathbf{S} \in \{e_1, \dots, e_d\}$) can be efficiently implemented by memorizing the residuals $\langle a_i, x^k \rangle$, which is cheap to track since $x^{k+1} - x^k$ is a sparse vector. The overall cost of updating the residuals is $\mathcal{O}(n)$ while the cost of computing $\nabla_i f(x)$ and $\nabla_{i,i}^2 f(x)$ (given the residuals are stored) is $\mathcal{O}(n)$. Therefore the overall cost of performing a single iteration is $\mathcal{O}(n)$. Generalizing to $\tau(\mathbf{S}) = \tau \geq 1$, the overall cost of single iteration of SSCN can be estimated as $\mathcal{O}(n\tau^2 + \tau^3)$, where $\mathcal{O}(n\tau^2)$ comes from evaluating subspace gradient and Hessian, while $\mathcal{O}(\tau^3)$ comes from solving the cubic subproblem.

9.8.2 Dual of linear models

So far, all results and applications for SSCN we mentioned were problems with large model size d . In this section we describe how SSCN can be efficient to tackle big data problems in some settings. Let $\mathbf{A} \in \mathbb{R}^{n \times d}$ is data matrix and consider a specific instance of (9.15) where

$$\min_{x \in \mathbb{R}^d} \left\{ F_P(x) \stackrel{\text{def}}{=} \frac{1}{d} \sum_{i=1}^n \rho_i(\mathbf{A}_{(:,i)} x) + \frac{\lambda}{2} \|x\|^2 \right\}. \quad (9.17)$$

where ρ_i is convex for all i . One can now formulate a dual problem of (9.17) as follows:

$$\max_{y \in \mathbb{R}^n} \left\{ F_D(y) \stackrel{\text{def}}{=} -\frac{1}{2\lambda n^2} \|\mathbf{A}^\top y\|^2 - \frac{1}{n} \sum_{i=1}^n \rho_i^*(e_i^\top x) \right\}. \quad (9.18)$$

Note that (9.18) is of form (9.15), and therefore if ρ_i^* has Lipschitz Hessian, we can apply SSCN to efficiently solve it (same as Section 9.8.1). Given the solution of (9.18), we can recover the solution of (9.17) (duality theory). Thus, SSCN can be used as a data-stochastic method to solve finite-sum optimization problems.

The trick described in this section is rather well known. It was first used in [191], where CD applied to the problem (9.18) (SDCA) was shown to be competitive with the variance reduced methods like SAG [182], SVRG [88] or SAGA [37].

9.9 Experiments

We now numerically verify our theoretical claims. We consider two different objectives: logistic regression (Section 9.9.1) and log-sum-exp (Section 9.9.2).

9.9.1 Logistic regression

Regularized logistic regression is a machine learning model for binary classification. Given data matrix $\mathbf{A} \in \mathbb{R}^{n \times d}$, labels $b \in \{-1, 1\}^n$ and regularization parameter $\lambda \in \mathbb{R}_+$, the training corresponds to solving the following optimization problem

$$f(x) = \frac{1}{n} \sum_{i=1}^n \log(1 + \exp(\mathbf{A}_{i,:} x \cdot b_i)) + \frac{\lambda}{2} \|x\|^2.$$

In the first experiment, we compare SSCN to first-order coordinate descent (CD) on LIBSVM [23]. We consider three different instances of CD: CD with uniform sampling, CD with importance sampling [152], and accelerated CD with importance sampling [7, 158].

In order to be comparable with the mentioned first-order methods, we consider $\mathbf{S} \in \{e_1, \dots, e_d\}$ with probability 1 – the complexity of performing each iteration is about the same for each algorithm now. At the same time, computing M_{e_i} for all $1 \leq i \leq d$ is of cost $\mathcal{O}(nd)$ – the same cost as computing coordinate-wise smoothness constants for (accelerated) CD (see Section 9.8.1 for the details). Figure 9.1 shows the result for non-normalized data, while Figure 9.2 shows the results for normalized data (thus importance sampling is identical to uniform).

In all examples, SSCN outperformed CD with uniform sampling. Moreover, the performance of SSCN was always either about the same or significantly better to CD with importance sampling. Furthermore, SSCN was also competitive to accelerated CD with importance sampling (in about half of the cases, SSCN was better, while in the other half, ACD was better).

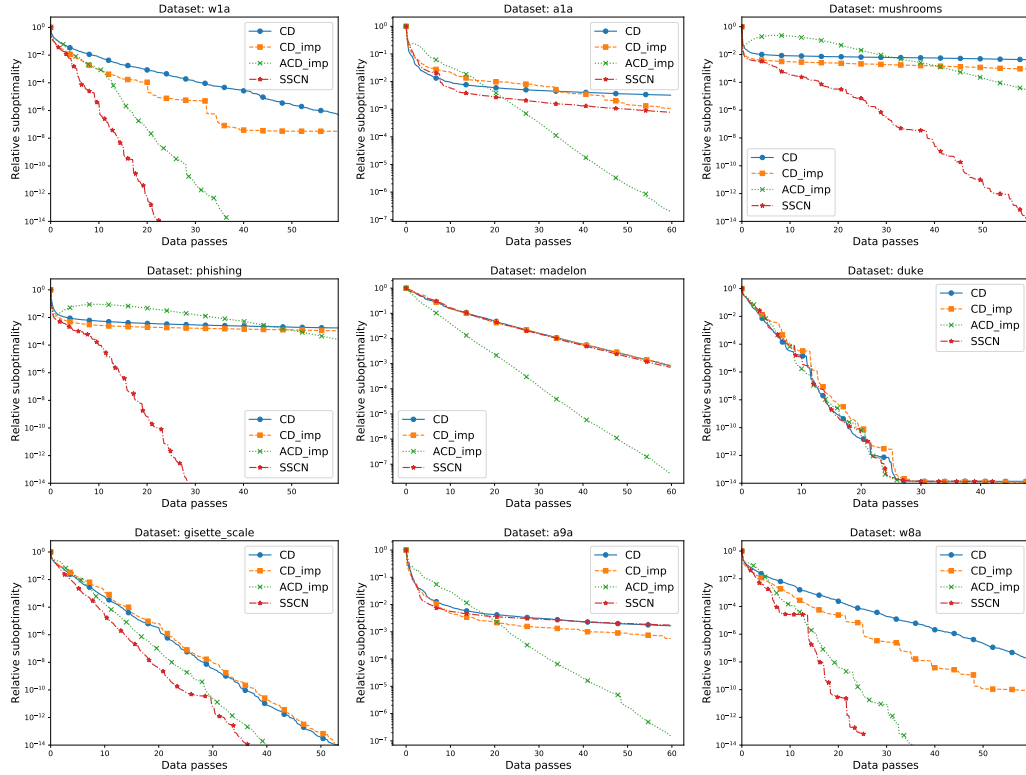


Figure 9.1: Comparison of CD with uniform sampling, CD with importance sampling, accelerated CD with importance sampling and SSCN (Algorithm 21) with uniform sampling on LibSVM datasets.

In the second experiment, we compare methods with $\tau > 1$: SSCN and SDNA [168] (analogous first-order method). Again, we consider the logistic regression problem on LIBSVM data. We consider $\tau \in \{1, 5, 25\}$. In all cases, we sample uniformly – every subset of size τ have equal chance to be chosen at every iteration (independent of the

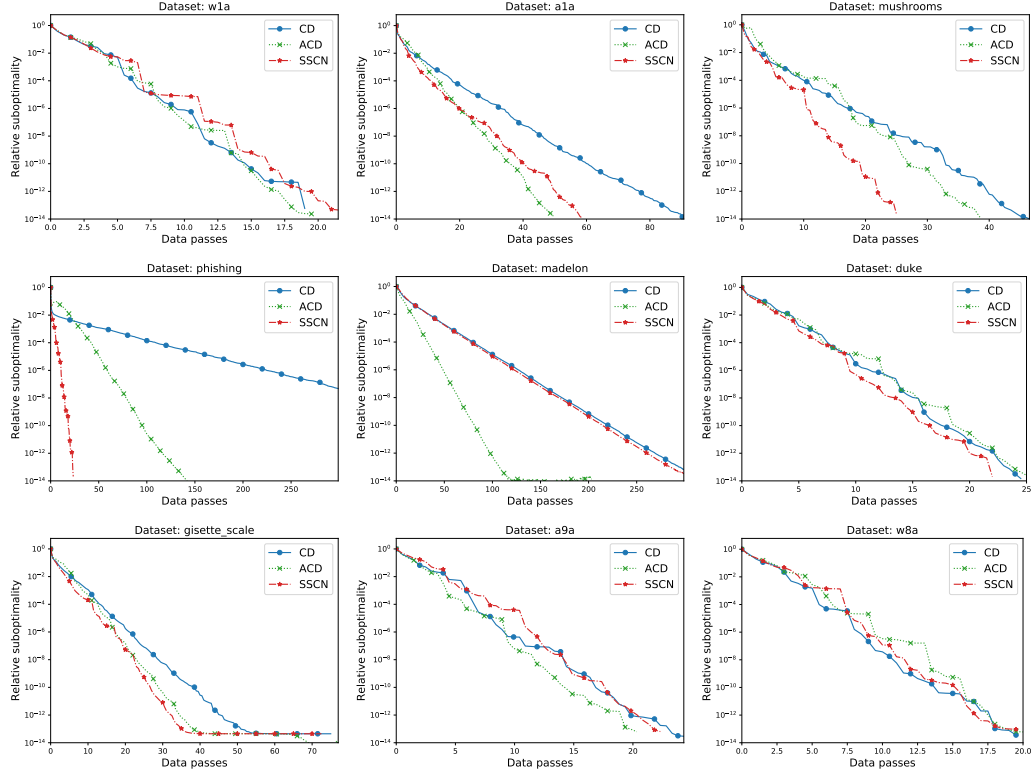


Figure 9.2: Comparison of coordinate descent, accelerated coordinate descent and SSCN (all with uniform sampling) on LibSVM datasets. In each case we have normalized the data matrix to have identical norms of all columns.

past).

There is, however, one tricky part in terms of implementation. While we can evaluate and store M_{e_i} ($i \leq d$) cheaply for linear models, this is not the case for evaluating/storing M_S (at least we do not know how to do it efficiently). Therefore, we use $M_S = M$ for $|S| > 1$ for SSCN. Figure 9.3 shows the result. As expected, SSCN has outperformed SDNA.

9.9.2 Log-sum-exp

In this section, let us consider unconstrained minimization of the following Log-sum-exp function

$$f(x) = \sigma \log \left(\sum_{i=1}^m \exp \left(\frac{\langle a_i, x \rangle - b_i}{\sigma} \right) \right), \quad x \in \mathbb{R}^d,$$

where $\sigma > 0$ is a *smoothing* parameter, while $a_i \in \mathbb{R}^n$, $1 \leq i \leq m$ and $b \in \mathbb{R}^m$ are given data. This function has both Lipschitz continuous gradient and Lipschitz continuous Hessian (see Example 1 in [42]).

In our experiments, we first generate randomly elements of $\{\tilde{a}_i\}_{i=1}^m$ and b from uniform

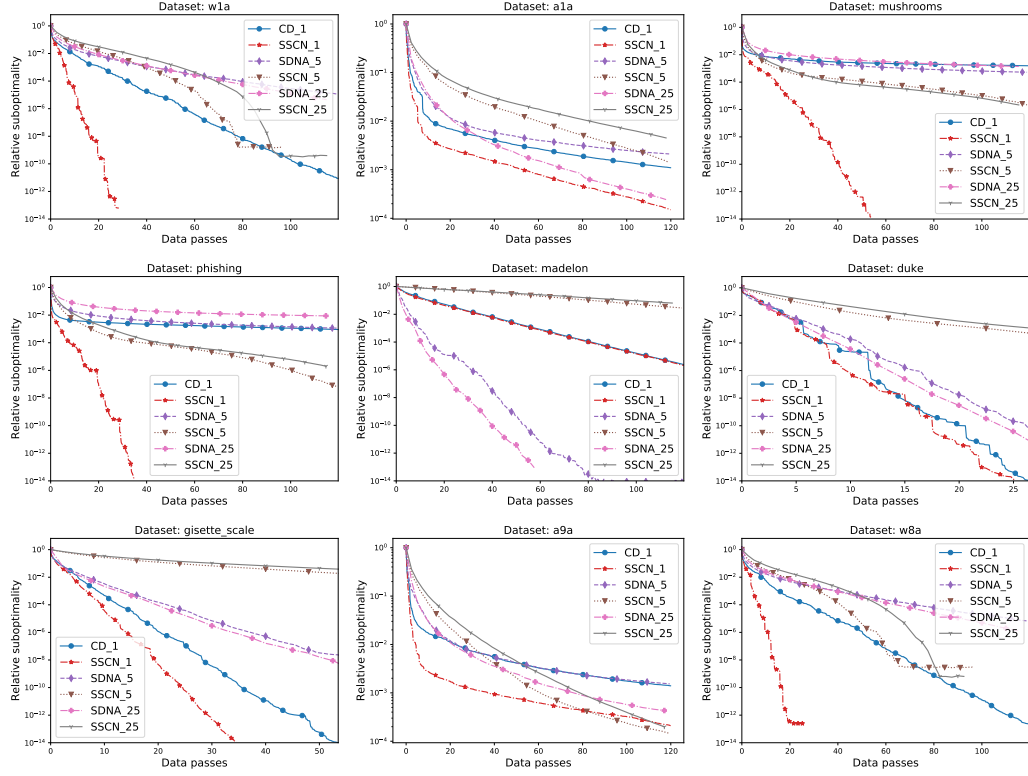


Figure 9.3: SSCN vs. SDNA on LibSVM datasets. All algorithms with uniform sampling.

distribution on $[-1, 1]$. Then, we form an auxiliary function

$$\tilde{f}(x) \stackrel{\text{def}}{=} \sigma \log \left(\sum_{i=1}^m \exp\left(\frac{\langle \tilde{a}_i, x \rangle - b_i}{\sigma}\right) \right),$$

using these parameters, and set

$$a_i \stackrel{\text{def}}{=} \tilde{a}_i - \nabla \tilde{f}(0), \quad 1 \leq i \leq m.$$

Thus, we essentially obtain the optimum x^* of f in the origin, since $\nabla f(0) = 0$. We use $x_0 \stackrel{\text{def}}{=} e$ (vector of all ones) as a starting point, and always set $m \stackrel{\text{def}}{=} 6d$.

For this problem, we compare the performance of SSCN with the first-order Coordinate Descent (CD), using uniform samples of coordinates $S \subseteq [d]$ of a fixed size $\tau = |S|$.

Note, that keeping scalar products $\{\langle a_i, x_k \rangle\}_{i=1}^m$ precomputed for a current point x_k , we are able to compute the partial gradient $\nabla_S f(x^k)$ in time $O(\tau m)$ and the partial Hessian $\nabla_S^2 f(x^k)$ in time $O(\tau^2 m)$. To find the next direction h^k of SSCN (solving the Cubic subproblem), we call Nonlinear Conjugate Gradient method, and use the following condition as a stopping criterion:

$$\|\nabla_h T_S(x^k; h^k)\| \leq 10^{-4},$$

where $T_S(x^k; h) \stackrel{\text{def}}{=} \langle \nabla_S f(x^k), h \rangle + \frac{1}{2} \langle \nabla_S^2 f(x^k) h, h \rangle + \frac{M_k}{6} \|Sh\|^3$ is the Cubic model, and $M_k \geq 0$ is a regularization constant.

For both methods, we use one-dimensional search at every iteration, to fit the corresponding parameter:

1. For the Coordinate Descent, we find L_k such that $f(x^k) - f(x^{k+1}) \geq \frac{1}{2L_k} \|\nabla_S f(x^k)\|^2$, where x^{k+1} is the next point of the method: $x^{k+1} = x^k + \frac{1}{L_k} S \nabla_S f(x^k)$.
2. For SSCN, we find M_k such that (9.6) is satisfied, i.e. $f(x^k) - f(x^{k+1}) \geq -T_S(x^k, h^k)$.

Therefore, we need to evaluate the function value inside the procedure, which is not very expensive.

The results are shown on Figures 9.4, 9.5, for $d = 500$ and 1000 respectively⁸. We see, that SSCN outperforms CD significantly in terms of the iteration rate. For SSCN with a medium batchsize τ , we may obtain the best performance in terms of the total computational time.

9.10 Conclusion

In this chapter, we have introduced SSCN, which is both a subspace version cubically-regularized Newton method [156], and a second-order enhancement of stochastic subspace descent [109]. The algorithm enjoys the global convergence to the optimum along with the fast local rates. We believe our method opens up several new avenues for the future research which we list next.

Acceleration. We believe it would be valuable to incorporate Nesterov's momentum into Algorithm 21. Ideally, one would like to get the global rate in between convergence rate of accelerated cubic regularized Newton [151] and accelerated CD [7, 158]. On the other hand, the local rate (for strongly convex objectives) should recover accelerated sketch-and-project [207, 58]. If accelerated sketch-and-project is optimal (this is yet to be established), then accelerated SSCN (again, given that it recovers accelerated sketch-and-project) would be a locally optimal algorithm as well.

Non-separable ψ . As mentioned in Section 9.6.1, one should not hope for linear convergence of SSCN if ψ is not separable, as the iterates can “jump” away from the optimum in such case. This issue has been resolved for first-order methods using control variates in Chapter 3 via SEGA algorithm. Therefore, the development of second-order SEGA remains an interesting open problem.

Inexact method. SSCN is applicable in the setup, where function f is accessible via zeroth-order oracle only. In such a case, for any $S \in \mathbb{R}^{\tau \times d}$ we can estimate $\nabla_S f(x)$ and $\nabla_S^2 f(x)$ using $\mathcal{O}(\tau^2)$ function value evaluations. However, since both $\nabla_S f(x)$ and $\nabla_S^2 f(x)$ are only evaluated inexactly, a slight modification of our theory is required.

⁸Clock time was evaluated using the machine with Intel Core i7-8700 CPU, 3.20GHz; 16 GB RAM.

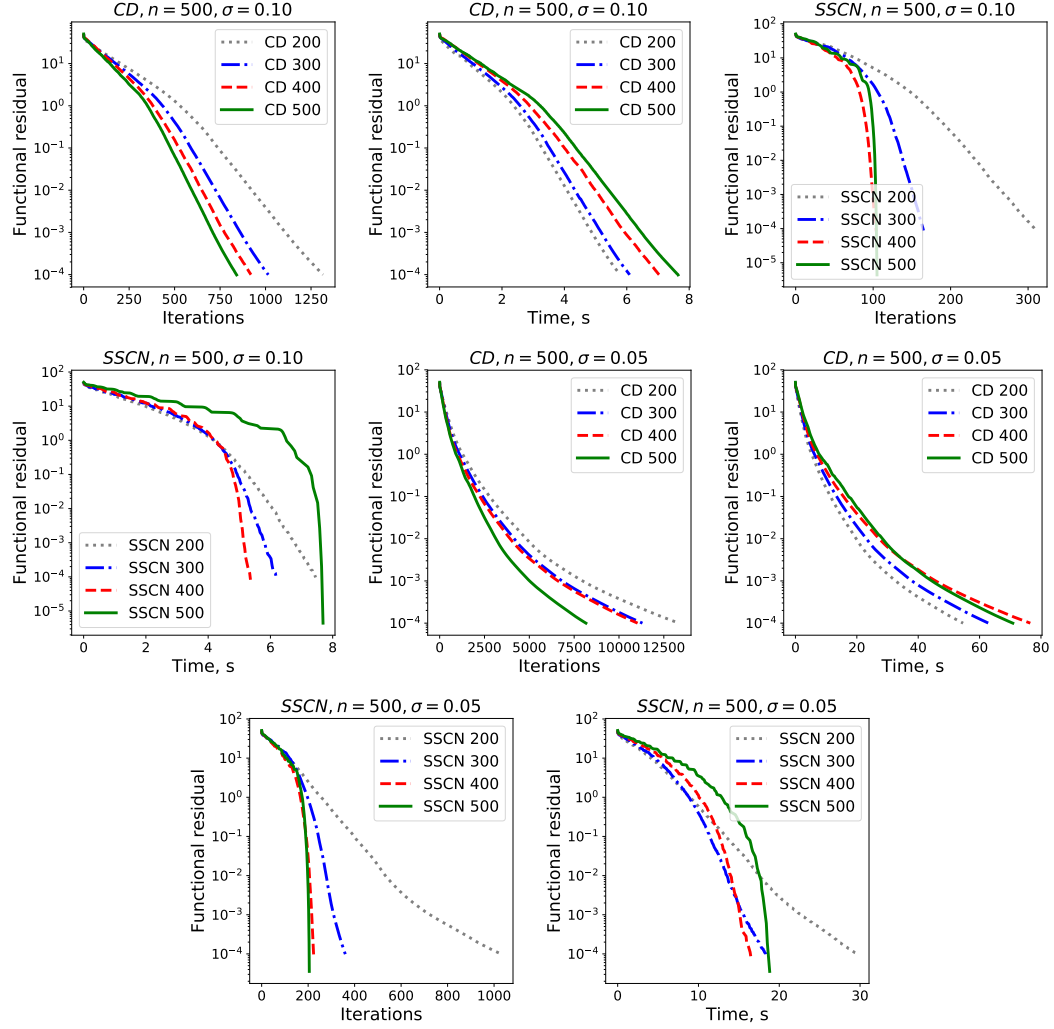


Figure 9.4: SSCN and Coordinate Descent (CD) methods, minimizing Log-Sum-Exp function, $d = 500$.

Non-uniform sampling. Note that our local theory allows for arbitrary non-uniform distribution of S , which might be potentially exploited. While developing optimal and implementable importance sampling for the local convergence is beyond the scope of this work,⁹ we sketch several possible sampling strategies that might yield faster convergence.¹⁰

- Let $\mathbb{P}(S \in \{e_1, e_2, \dots, e_d\}) = 1$. If we evaluate the diagonal of the Hessian close to optimum (cost $\mathcal{O}(nd)$ for linear models) and sample proportionally to it, we obtain local linear rate with leading complexity term $\frac{\text{Tr}(\nabla^2 f(x^*))}{\lambda_{\min} \nabla^2 f(x^*)}$.
- It is unclear how to design an efficient importance sampling for minibatch (i.e. $1 < \mathbb{E}[\tau(S)] < d$) methods. Determinantal point processes (DPP) [180, 144] were

⁹As this is still an open problem even for sketch-and-project [61].

¹⁰This only applies to the local results as the global convergence requires some uniformity; see Assumption 9.6.1.

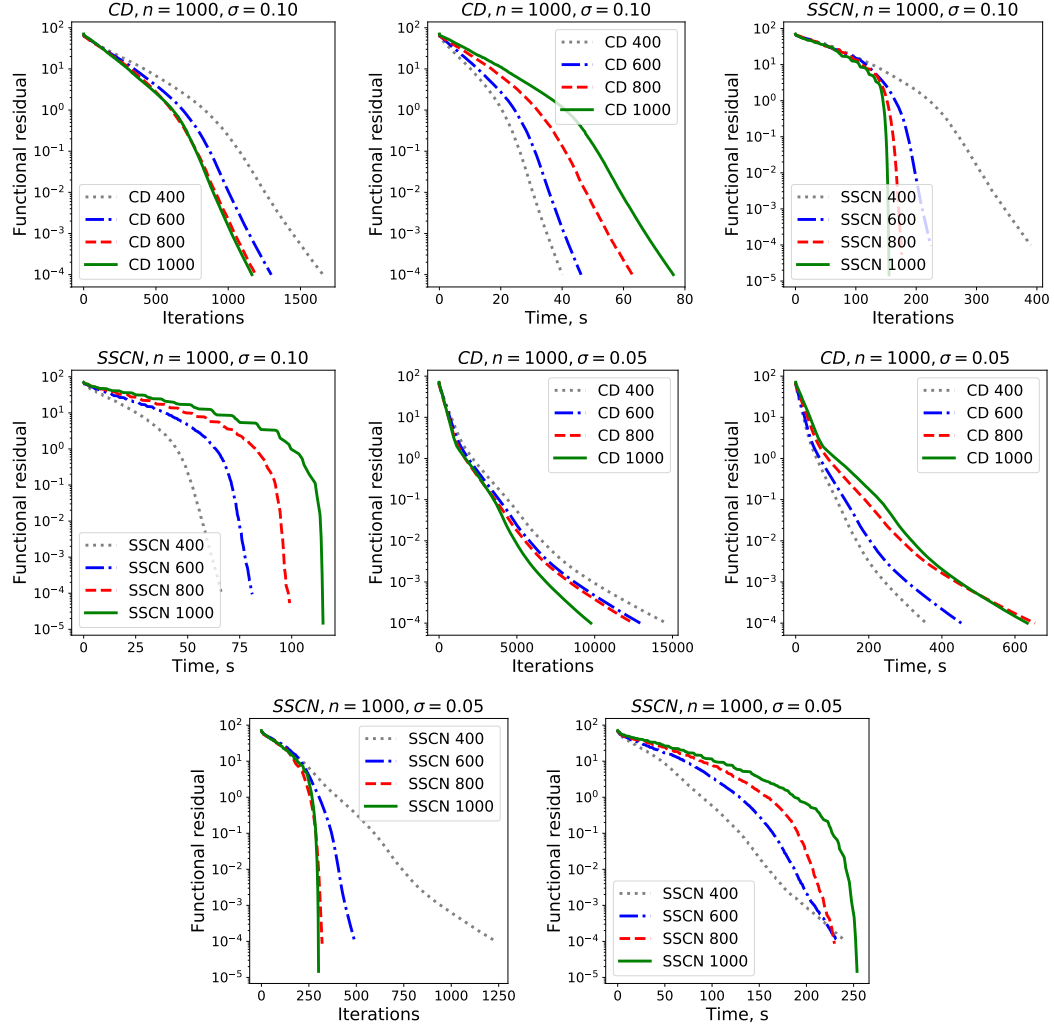


Figure 9.5: SSCN and Coordinate Descent (CD) methods, minimizing Log-Sum-Exp function, $d = 1000$.

proposed to speed up SDNA from [168] (i.e., analogous CD with static matrix upper bound) – we thus believe they might be applicable on our setting too. However, in such a case, one would need to evaluate the whole Hessian close to optimum, which is infeasible for applications where d is large.

- It is known that SDNA (see related literature) is faster than minibatch CD under the ESO assumption [166, 167]. Therefore, we might instead apply minibatch importance sampling for ESO assumption from [78] (which corresponds to optimizing the upper bound on iteration complexity). Using the mentioned sampling, we only require evaluating the diagonal of Hessian at some point close to optimum, which is of the same cost as computing the full gradient for linear models – thus is feasible.
- It is a natural question to ask whether one can speed up the convergence using greedy rule instead of random one. For standard CD, greedy rule was shown to have a superior iteration complexity to any randomized rule [161, 93]. For simplicity,

consider case where $\mathbb{P}(\mathbf{S} \in \{e_1, e_2, \dots, e_d\}) = 1$. Far from the optimum, (approximate) greedy rule at iteration k chooses index $i = \arg \max_j |\nabla_j f(x^k)|^{\frac{3}{2}} M_{e_j}^{-\frac{1}{2}}$. Close to optimum, if a diagonal of a Hessian was evaluated, (approximate) greedy index would be $\arg \max_j |\nabla_j f(x^k)|^2 \nabla_{j,j} f(x)^{-1}$. For linear models, both of the mentioned cases are implementable using the efficient nearest neighbour search [41] with sublinear complexity in terms of d .

Chapter 10

Accelerated Stochastic Matrix Inversion: General Theory and Speeding up BFGS Rules for Faster Second-Order Optimization

A new wave of second-order stochastic methods are being developed nowadays with the aim of solving large scale optimization problems. In particular, many of these new methods are often based on stochastic BFGS updates [186, 210, 140, 142, 16, 32, 11]. Another approach to scaling up second-order methods is to use randomized *sketching* to reduce the dimension, and hence the complexity of the Hessian and the updates involving the Hessian [163, 219], or *subsampling* Hessian matrices when the objective function is a sum of many loss functions [15, 10, 1, 218].

In this chapter we develop a new stochastic accelerated BFGS update that can form the backbone of new stochastic quasi-Newton methods. Since the BFGS update mechanism which we improve upon is as an optimization routine on its own, this chapter tackles two different objectives in two different domains at the same time. For this reason, the notation will be slightly inconsistent with respect to the rest of the thesis. Specifically, our high-level goal is to minimize smooth function f in variable w :

$$\min_{w \in \mathbb{R}^d} f(w), \quad (10.1)$$

while the mentioned BFGS subroutine is (as we shall see) a quadratic objective in matrix variable \mathbf{X} (or x in the vectorized form). Given the (admittedly inconsistent) notation is explained, let us properly motivate our work.

The starting point for developing second-order methods is arguably Newton's method, which performs the iterative process

$$w_{k+1} = w_k - (\nabla^2 f(w_k))^{-1} \nabla f(w_k), \quad (10.2)$$

where $\nabla^2 f(w_k)$ and $\nabla f(w_k)$ are the Hessian and gradient of f , respectively. However, it is inefficient for solving large scale problems as it requires the computation of the Hessian and then solving a linear system at each iteration. Several methods have been developed to address this issue, based on the idea of approximating the exact update.

Quasi-Newton methods, in particular BFGS [14, 51, 53, 195], have been the leading optimization algorithm in various fields since the late 60's until the rise of big data, which brought a need for simpler first-order algorithms. It is well known that Nesterov's acceleration [149] is a reliable way to speed up first-order methods. However until now, acceleration techniques have been applied exclusively to speeding up gradient updates. In this chapter we present an accelerated BFGS algorithm, opening up new applications for acceleration. The acceleration in fact comes from an accelerated algorithm for inverting

the Hessian matrix.

To be more specific, recall that quasi-Newton rules aim to maintain an estimate of the inverse Hessian \mathbf{X}_k , adjusting it every iteration so that the inverse Hessian acts appropriately in a particular direction, while enforcing symmetry:

$$\mathbf{X}_k(\nabla f(w_k) - \nabla f(w_{k-1})) = w_k - w_{k-1}, \quad \mathbf{X}_k = \mathbf{X}_k^\top. \quad (10.3)$$

A notable research direction is the development of stochastic quasi-Newton methods [64], where the estimated inverse is equal to the true inverse over a subspace:

$$\mathbf{X}_k \nabla^2 f(w_k) \mathbf{S}_k = \mathbf{S}_k, \quad \mathbf{X}_k = \mathbf{X}_k^\top, \quad (10.4)$$

where $\mathbf{S}_k \in \mathbb{R}^{d \times \tau}$ is a randomly generated matrix.

In fact, (10.4) can be seen as the so called sketch-and-project iteration for inverting $\nabla^2 f(w_k)$. In this chapter we first develop the accelerated algorithm for inverting positive definite matrices. As a direct application, our algorithm can be used as a primitive in quasi-Newton methods which results in a novel accelerated (stochastic) quasi-Newton method of the type (10.4). In addition, our acceleration technique can also be incorporated in the classical (non stochastic) BFGS method. This results in the accelerated BFGS method. Whereas the matrix inversion contribution is accompanied by strong theoretical justifications, this does not apply to the latter. Rather, we verify the effectiveness of this new accelerated BFGS method through numerical experiments.

10.1 Sketch-and-project for linear systems

Our accelerated algorithm can be applied to more general tasks than only inverting matrices. In its most general form, it can be seen as an accelerated version of a *sketch-and-project* method in Euclidean spaces which we present now. Consider a linear system $\mathbf{A}x = b$ such that $b \in \text{Range}(\mathbf{A})$. One step of the sketch-and-project algorithm reads as:

$$x_{k+1} = \arg \min_x \|x_k - x\|_{\mathbf{B}}^2 \quad \text{subject to} \quad \mathbf{S}_k^\top \mathbf{A}x = \mathbf{S}_k^\top b, \quad (10.5)$$

where $\|x\|_{\mathbf{B}}^2 = \langle \mathbf{B}x, x \rangle$ for some $\mathbf{B} \succ 0$ and \mathbf{S}_k is a random sketching matrix sampled i.i.d at each iteration from a fixed distribution.

Randomized Kaczmarz [89, 203] was the first algorithm of this type. In [61], this sketch-and-project algorithm was analyzed in its full generality. Note that the dual problem of (10.5) takes the form of a quadratic minimization problem [62], and randomized methods such as coordinate descent [152, 215], random pursuit [197, 200] or stochastic dual ascent [62] can thus also be captured as special instances of this method. Richtárik and Takáč [176] adopt a new point of view through a theory of stochastic reformulations of linear systems. In addition, they consider the addition of a relaxation parameter, as well as mini-batch and accelerated variants. Acceleration was only achieved for the expected iterates, and not in the L2 sense as we do here. We refer to Richtárik and Takáč [176] for interpretation of sketch-and-project as stochastic gradient descent, stochastic Newton, stochastic proximal point method, and stochastic fixed point method.

Gower [64] observed that the procedure (10.5) can also be applied to find the inverse of a matrix. Assume the optimization variable itself is a matrix, $x = \mathbf{X}$, $b = \mathbf{I}$, the identity matrix, then sketch-and-project converges (under mild assumptions) to a solution of $\mathbf{A}\mathbf{X} = \mathbf{I}$. Even the symmetry constraint $\mathbf{X} = \mathbf{X}^\top$ can be incorporated into the sketch-and-project framework since it is a linear constraint.

There has been recent development in speeding up the sketch-and-project method using the idea of Nesterov's acceleration [149]. In [125] an accelerated Kaczmarz algorithm was presented for special sketches of rank one. Arbitrary sketches of rank one were considered in [197], block sketches in [157] and recently, Tu and coauthors [207] developed acceleration for special sketching matrices, assuming the matrix \mathbf{A} is square. This assumption, along with any assumptions on \mathbf{A} , was later dropped in [178]. Another notable way to accelerate the sketch-and-project algorithm is by using momentum or stochastic momentum [129].

We build on recent work of Richtárik and Takáč [178] and further extend their analysis by studying accelerated sketch-and-project in general Euclidean spaces. This allows us to deduce the result for matrix inversion as a special case. However, there is one additional caveat that has to be considered for the intended application in quasi-Newton methods: ideally, all iterates of the algorithm should be symmetric positive definite matrices. This is not the case in general, but we address this problem by constructing special sketch operators that preserve symmetry and positive definiteness.

Our accelerated sketch-and-project algorithm for solving linear systems in Euclidean spaces is developed and analyzed in Section 10.3, and is used later in Section 10.4 to analyze an accelerated sketch-and-project algorithm for matrix inversion. The accelerated sketch-and-project algorithm for matrix inversion is then used to accelerate the BFGS update, which in turn leads to the development of an accelerated BFGS optimization method. Lastly in Section 10.5, we perform numerical experiments to gain different insights into the newly developed methods. Proofs of all results and additional insights can be found in the appendix.

10.2 Contributions

We now present our main contributions.

- **Accelerated Sketch and Project in Euclidean Spaces.** We generalize the analysis of an accelerated version of the sketch-and-project algorithm [178] to linear operator systems in Euclidean spaces. We provide a self-contained convergence analysis, recovering the original results in a more general setting.
- **Faster Algorithms for Matrix Inversion.** We develop an accelerated algorithm for inverting positive definite matrices. This algorithm can be seen as a special case of the accelerated sketch-and-project in Euclidean space, thus its convergence follows from the main theorem. However, we also provide a different formulation of the proof that is specialized to this setting. Similarly to [207], the performance of the algorithm depends on two parameters θ and ν that capture spectral properties of the

input matrix and the sketches that are used. Whilst for the non-accelerated sketch-and-project algorithm for matrix inversion [64] the knowledge of these parameters is not necessary, they need to be given as input to the accelerated scheme. When employed with the correct choice of parameters, the accelerated algorithm is always faster than the non-accelerated one. We also provide a theoretical rate for sub-optimal parameters θ, ν , and we perform numerical experiments to argue the choice of θ, ν in practice.

- **Randomized Accelerated Quasi-Newton.** The proposed iterative algorithm for matrix inversion is designed in such a way that each iterate is a symmetric matrix. This means, we can use the generated approximate solutions as estimators for the inverse Hessian in quasi-Newton methods, which is a direct extension of stochastic quasi-Newton methods. To the best of our knowledge, this yields the first accelerated (stochastic) quasi-Newton method.
- **Accelerated Quasi-Newton.** In the standard BFGS method the updates to the Hessian estimate are not chosen randomly, but deterministically. Based on the intuition gained from the accelerated random method, we propose an accelerated scheme for BFGS. The main idea is that we replace the random sketching of the Hessian with a deterministic update. The theoretical convergence rates do not transfer to this scheme, but we demonstrate by numerical experiments that it is possible to choose a parameter combination which yields a slightly faster convergence. We believe that the novel idea of accelerating BFGS update is extremely valuable, as until now, acceleration techniques were only considered to improve gradient updates.

10.3 Accelerated stochastic algorithm for matrix inversion

In this section we propose an accelerated randomized algorithm to solve linear systems in Euclidean spaces. This is a very general problem class which comprises the matrix inversion problem as well. Thus, we will use the result of this section later to analyze our newly proposed matrix inversion algorithm, which we then use to estimate the inverse of the Hessian within a quasi-Newton method.¹

Let \mathcal{X} and \mathcal{Y} be finite dimensional Euclidean spaces and let $\mathcal{A} : \mathcal{X} \mapsto \mathcal{Y}$ be a linear operator. Let $L(\mathcal{X}, \mathcal{Y})$ denote the space of linear operators that map from \mathcal{X} to \mathcal{Y} . Consider the linear system

$$\mathcal{A}x = b, \tag{10.6}$$

where $x \in \mathcal{X}$ and $b \in \text{Range}(\mathcal{A})$. Consequently there exists a solution to the equation (10.6). In particular, we aim to find the solution closest to a given initial point $x_0 \in \mathcal{X}$:

$$x^* \stackrel{\text{def}}{=} \arg \min_{x \in \mathcal{X}} \frac{1}{2} \|x - x_0\|^2 \quad \text{subject to} \quad \mathcal{A}x = b. \tag{10.7}$$

¹Quasi-Newton methods do not compute an exact matrix inverse, rather, they only compute an incremental update. Thus, it suffices to apply *one step* of our proposed scheme per iteration. This will be detailed in Section 10.4.

Using the pseudoinverse and Lemma J.8.8 item 6, the solution to (10.7) is given by

$$x^* = x_0 - \mathcal{A}^\dagger(\mathcal{A}x_0 - b) \in x_0 + \mathbf{Range}(\mathcal{A}^*), \quad (10.8)$$

where \mathcal{A}^\dagger and \mathcal{A}^* denote the pseudoinverse and the adjoint of \mathcal{A} , respectively.

10.3.1 The algorithm

Let \mathcal{W} be a Euclidean space and consider a random linear operator $\mathcal{S}_k \in L(\mathcal{Y}, \mathcal{W})$ chosen from some distribution \mathcal{D} over $L(\mathcal{Y}, \mathcal{W})$ at iteration k . Our method is given in Algorithm 22, where $\mathcal{Z}_k \in L(\mathcal{X})$ is a random linear operator given by the following compositions

$$\mathcal{Z}_k = \mathcal{Z}(\mathcal{S}_k) \stackrel{\text{def}}{=} \mathcal{A}^* \mathcal{S}_k^* (\mathcal{S}_k \mathcal{A} \mathcal{A}^* \mathcal{S}_k^*)^\dagger \mathcal{S}_k \mathcal{A}. \quad (10.9)$$

The updates of variables g_k and x_{k+1} on lines 8 and 9, respectively, correspond to what is known as the *sketch-and-project* update:

$$x_{k+1} = \arg \min_{x \in \mathcal{X}} \frac{1}{2} \|x - y_k\|^2 \quad \text{subject to} \quad \mathcal{S}_k \mathcal{A}x = \mathcal{S}_k b, \quad (10.10)$$

which can also be written as the following operation

$$x_{k+1} - x_* = (\mathcal{I} - \mathcal{Z}_k)(y_k - x_*), \quad (10.11)$$

where \mathcal{I} is the identity operator. This follows from the fact that $b \in \mathbf{Range}(\mathcal{A})$, together with item 1 of Lemma J.8.8. Furthermore, note that the adjoint \mathcal{A}^* and the pseudoinverse in Algorithm 22 are taken with respect to the norm in (10.7).

Algorithm 22 Accelerated Sketch-and-Project for solving (10.10) [178]

- 1: **Parameters:** $\theta, \nu > 0$, \mathcal{D} = distribution over random linear operators.
 - 2: Choose $x_0 \in \mathcal{X}$ and set $v_0 = x_0$, $\beta = 1 - \sqrt{\frac{\theta}{\nu}}$, $\gamma = \sqrt{\frac{1}{\theta\nu}}$, $\eta = \frac{1}{1+\gamma\nu}$.
 - 3: **for** $k = 0, 1, 2, \dots$ **do**
 - 4: $y_k = \eta v_k + (1 - \eta)x_k$
 - 5: Sample an independent copy $\mathcal{S}_k \sim \mathcal{D}$
 - 6: $g_k = \mathcal{A}^* \mathcal{S}_k^* (\mathcal{S}_k \mathcal{A} \mathcal{A}^* \mathcal{S}_k^*)^\dagger \mathcal{S}_k (\mathcal{A}y_k - b) = \mathcal{Z}_k(y_k - x_*)$
 - 7: $x_{k+1} = y_k - g_k$
 - 8: $v_{k+1} = \beta v_k + (1 - \beta)y_k - \gamma g_k$
 - 9: **end for**
-

Algorithm 22 was first proposed and analyzed by Richtárik and Takáč [178] for the special case when $\mathcal{X} = \mathbb{R}^d$ and $\mathcal{Y} = \mathbb{R}^m$. Our contribution here is in extending the algorithm and analysis to the more abstract setting of Euclidean spaces. In addition, we provide some further extensions of this method in Sections J.3 and J.4, allowing for a non-unit stepsize and variable η , respectively.

10.3.2 Key assumptions and quantities

Denote $\mathcal{Z} = \mathcal{Z}(S)$ for $S \sim \mathcal{D}$. Assume that the *exactness property* holds

$$\text{Null}(\mathcal{A}) = \text{Null}(\mathbb{E}[\mathcal{Z}]); \quad (10.12)$$

this is also equivalent to $\text{Range}(\mathcal{A}^*) = \text{Range}(\mathbb{E}[\mathcal{Z}])$. The exactness assumption is of key importance in the sketch-and-project framework, and indeed it is not very strong. For example, it holds for the matrix inversion problem with every sketching strategy we consider. We further assume that $\mathcal{A} \neq 0$ and $\mathbb{E}[\mathcal{Z}]$ is finite. First we collect a few observation on the \mathcal{Z} operator

Lemma 10.3.1. *The \mathcal{Z} operator (10.9) is a self-adjoint positive projection. Consequently $\mathbb{E}[\mathcal{Z}]$ is a self-adjoint positive operator.*

The two parameters that govern the acceleration are

$$\theta \stackrel{\text{def}}{=} \inf_{x \in \text{Range}(\mathcal{A}^*)} \frac{\langle \mathbb{E}[\mathcal{Z}]x, x \rangle}{\langle x, x \rangle}, \quad \nu \stackrel{\text{def}}{=} \sup_{x \in \text{Range}(\mathcal{A}^*)} \frac{\langle \mathbb{E}[\mathcal{Z}\mathbb{E}[\mathcal{Z}]^\dagger \mathcal{Z}]x, x \rangle}{\langle \mathbb{E}[\mathcal{Z}]x, x \rangle}. \quad (10.13)$$

The supremum in the definition of ν is well defined due to the exactness assumption together with $\mathcal{A} \neq 0$.

Lemma 10.3.2. *We have*

$$1 \leq \nu \leq \frac{1}{\theta} = \left\| \mathbb{E}[\mathcal{Z}]^\dagger \right\|. \quad (10.14)$$

Moreover, if $\text{Range}(\mathcal{A}^*) = \mathcal{X}$, we have

$$\frac{\text{Rank}(\mathcal{A}^*)}{\mathbb{E}[\text{Rank}(\mathcal{Z})]} \leq \nu. \quad (10.15)$$

10.3.3 Convergence and change of the norm

For a positive self-adjoint $\mathcal{G} \in L(\mathcal{X})$ and $x \in \mathcal{X}$ let $\|x\|_{\mathcal{G}} \stackrel{\text{def}}{=} \sqrt{\langle x, x \rangle_{\mathcal{G}}} \stackrel{\text{def}}{=} \sqrt{\langle \mathcal{G}x, x \rangle}$. We now informally state the convergence rate of Algorithm 22. Theorem 10.3.3 generalizes the main theorem from [178] to linear systems in Euclidean spaces.

Theorem 10.3.3. *Let x_k, v_k be the random iterates of Algorithm 22. Then*

$$\mathbb{E} \left[\|v_k - x_*\|_{\mathbb{E}[\mathcal{Z}]^\dagger}^2 + \frac{1}{\theta} \|x_k - x_*\|^2 \right] \leq \left(1 - \sqrt{\frac{\theta}{\nu}} \right)^k \mathbb{E} \left[\|v_0 - x_*\|_{\mathbb{E}[\mathcal{Z}]^\dagger}^2 + \frac{1}{\theta} \|x_0 - x_*\|^2 \right].$$

This theorem shows the accelerated Sketch-and-Project algorithm converges linearly with a rate of $(1 - \sqrt{\frac{\theta}{\nu}})$, which translates to a total of $O(\sqrt{\nu/\theta} \log(1/\epsilon))$ iterations to bring the given error in Theorem 10.3.3 below $\epsilon > 0$. This is in contrast with the non-accelerated Sketch-and-Project algorithm which requires $O((1/\theta) \log(1/\epsilon))$ iterations, as

shown in [61] for solving linear systems. From (10.14), we have the bounds $1/\sqrt{\theta} \leq \sqrt{\nu/\theta} \leq 1/\theta$. On one extreme, this inequality shows that the iteration complexity of the accelerated algorithm is at least as good as its non-accelerated counterpart. On the other extreme, the accelerated algorithm might require as little as the square root of the number of iterations of its non-accelerated counterpart. Since the cost of a single iteration of the accelerated algorithm is of the same order as the non-accelerated algorithm, this theorem shows that acceleration can offer a significant speed-up, which is verified numerically in Section 10.5. It is also possible to get the convergence rate of accelerated sketch-and-project where projections are taken with respect to a different weighted norm. For technical details, see Section J.1.4 of the Appendix.

10.3.4 Coordinate sketches with convenient probabilities

Let us consider a simple example in the setting for Algorithm 22 where we can understand parameters θ, ν . In particular, consider a linear system $\mathbf{A}x = b$ in \mathbb{R}^d where \mathbf{A} is symmetric positive definite.

Corollary 10.3.4. *Choose $\mathbf{B} = \mathbf{A}$ and $\mathbf{S} = e_i$ with probability proportional to $\mathbf{A}_{i,i}$. Then*

$$\theta = \frac{\lambda_{\min}(\mathbf{A})}{\text{Tr}(\mathbf{A})} =: \theta^P \quad \text{and} \quad \nu = \frac{\text{Tr}(\mathbf{A})}{\min_i \mathbf{A}_{i,i}} =: \nu^P \quad (10.16)$$

and therefore the convergence rate given in Theorem 10.3.3 for the accelerated algorithm is

$$\left(1 - \sqrt{\frac{\theta}{\nu}}\right)^k = \left(1 - \frac{\sqrt{\lambda_{\min}(\mathbf{A}) \min_i \mathbf{A}_{i,i}}}{\text{Tr}(\mathbf{A})}\right)^k. \quad (10.17)$$

Rate (10.17) of our accelerated method is to be contrasted with the rate of the non-accelerated method: $(1 - \theta)^k = (1 - \lambda_{\min}(\mathbf{A})/\text{Tr}(\mathbf{A}))^k$. Clearly, we gain from acceleration if the smallest diagonal element of \mathbf{A} is significantly larger than the smallest eigenvalue.

In fact, parameters θ^P, ν^P above are the correct choice for the matrix inversion algorithm, when symmetry is not enforced, as we shall see later. Unfortunately, we are not able to estimate the parameters while enforcing symmetry for different sketching strategies. We dedicate a section in numerical experiments to test, if the parameter selection (10.16) performs well under enforced symmetry and different sketching strategies, and also how one might safely choose θ, ν in practice.

10.4 Accelerated stochastic BFGS update

The update of the inverse Hessian used in quasi-Newton methods (e.g., in BFGS) can be seen as a sketch-and-project update applied to the linear system $\mathbf{A}\mathbf{X} = \mathbf{I}$, while $\mathbf{X} = \mathbf{X}^\top$ is enforced, and where \mathbf{A} denotes an approximation of the Hessian. In this section, we present an accelerated version of these updates. We provide two different proofs: one based on Theorem 10.3.3 and one based on vectorization. By mimicking the updates of the accelerated stochastic BFGS method for inverting matrices, we determine a heuristic for

accelerating the classic deterministic BFGS update. We then incorporate this acceleration into the classic BFGS optimization method and show that the resulting algorithm can offer a speed-up of the standard BFGS algorithm.

10.4.1 The AMI algorithm

Consider the symmetric positive definite matrix $\mathbf{A} \in \mathbb{R}^{d \times d}$ and the following projection problem

$$\mathbf{A}^{-1} = \arg \min_{\mathbf{X}} \|\mathbf{X}\|_{F(\mathbf{A})}^2 \quad \text{subject to} \quad \mathbf{A}\mathbf{X} = \mathbf{I}, \quad \mathbf{X} = \mathbf{X}^\top, \quad (10.18)$$

where $\|\mathbf{X}\|_{F(\mathbf{A})}^2 \stackrel{\text{def}}{=} \text{Tr}(\mathbf{A}\mathbf{X}^\top \mathbf{A}\mathbf{X}) = \|\mathbf{A}^{1/2} \mathbf{X} \mathbf{A}^{1/2}\|_F^2$. This projection problem can be cast as an instantiation of the general projection problem (10.7). Indeed, we need only note that the constraint in (10.18) is linear and equivalent to $\mathbf{A}(\mathbf{X}) \stackrel{\text{def}}{=} \begin{pmatrix} \mathbf{A}\mathbf{X} \\ \mathbf{X} - \mathbf{X}^\top \end{pmatrix} = \begin{pmatrix} \mathbf{I} \\ \mathbf{0} \end{pmatrix}$. The matrix inversion problem can be efficiently solved using sketch-and-project with a symmetric sketch [64]. The symmetric sketch is given by $\mathcal{S}_k \mathcal{A}(\mathbf{X}) = \begin{pmatrix} \mathbf{S}_k^\top \mathbf{A}\mathbf{X} \\ \mathbf{X} - \mathbf{X}^\top \end{pmatrix}$, where $\mathbf{S}_k \in \mathbb{R}^{d \times \tau}$ is a random matrix drawn from a distribution \mathcal{D} and $\tau \in \mathbb{N}$. The resulting sketch-and-project method is as follows

$$\mathbf{X}_{k+1} = \arg \min_{\mathbf{X}} \|\mathbf{X} - \mathbf{X}_k\|_{F(\mathbf{A})}^2 \quad \text{subject to} \quad \mathbf{S}_k^\top \mathbf{A}\mathbf{X} = \mathbf{S}_k^\top, \quad \mathbf{X} = \mathbf{X}^\top, \quad (10.19)$$

the closed form solution of which is

$$\mathbf{X}_{k+1} = \mathbf{S}_k (\mathbf{S}_k^\top \mathbf{A} \mathbf{S}_k)^{-1} \mathbf{S}_k^\top + (\mathbf{I} - \mathbf{S}_k (\mathbf{S}_k^\top \mathbf{A} \mathbf{S}_k)^{-1} \mathbf{S}_k^\top \mathbf{A}) \mathbf{X}_k (\mathbf{I} - \mathbf{A} \mathbf{S}_k (\mathbf{S}_k^\top \mathbf{A} \mathbf{S}_k)^{-1} \mathbf{S}_k^\top). \quad (10.20)$$

By observing that (10.20) is the sketch-and-project algorithm applied to a linear operator equation, we have constructed an accelerated version in Algorithm 23. We can also apply Theorem 10.3.3 to prove that Algorithm 23 is indeed accelerated.

Theorem 10.4.1. *Let $\mathbf{L}^k \stackrel{\text{def}}{=} \|\mathbf{V}_k - \mathbf{A}^{-1}\|_C^2 + \frac{1}{\theta} \|\mathbf{X}_k - \mathbf{A}^{-1}\|_{F(\mathbf{A})}^2$. The iterates of Algorithm 23 satisfy*

$$\mathbb{E} [\mathbf{L}_{k+1}] \leq \left(1 - \sqrt{\frac{\theta}{\nu}}\right) \mathbb{E} [\mathbf{L}_k], \quad (10.21)$$

where $\|\mathbf{X}\|_C^2 = \text{Tr}(\mathbf{A}^{1/2} \mathbf{X}^\top \mathbf{A}^{1/2} \mathbb{E} [\mathbf{Z}]^\dagger \mathbf{A}^{1/2} \mathbf{X} \mathbf{A}^{1/2})$. Furthermore,

$$\theta \stackrel{\text{def}}{=} \inf_{\mathbf{X} \in \mathbb{R}^{d \times d}} \frac{\langle \mathbb{E} [\mathbf{Z}] \mathbf{X}, \mathbf{X} \rangle}{\langle \mathbf{X}, \mathbf{X} \rangle} = \lambda_{\min}(\mathbb{E} [\mathbf{Z}']), \quad \nu \stackrel{\text{def}}{=} \sup_{\mathbf{X} \in \mathbb{R}^{d \times d}} \frac{\langle \mathbb{E} [\mathbf{Z} \mathbb{E} [\mathbf{Z}]^\dagger \mathbf{Z}] \mathbf{X}, \mathbf{X} \rangle}{\langle \mathbb{E} [\mathbf{Z}] \mathbf{X}, \mathbf{X} \rangle}, \quad (10.22)$$

where

$$\mathbf{Z}' \stackrel{\text{def}}{=} \mathbf{I} \otimes \mathbf{I} - (\mathbf{I} - \mathbf{P}) \otimes (\mathbf{I} - \mathbf{P}), \quad \mathbf{P} \stackrel{\text{def}}{=} \mathbf{A}^{1/2} \mathbf{S} (\mathbf{S}^\top \mathbf{A} \mathbf{S})^{-1} \mathbf{S}^\top \mathbf{A}^{1/2}, \quad (10.23)$$

and $\mathbf{Z} : \mathbf{X} \in \mathbb{R}^{d \times d} \rightarrow \mathbb{R}^{d \times d}$ is given by $\mathbf{Z}(\mathbf{X}) = \mathbf{X} - (\mathbf{I} - \mathbf{P}) \mathbf{X} (\mathbf{I} - \mathbf{P}) = \mathbf{X} \mathbf{P} + \mathbf{P} \mathbf{X} (\mathbf{I} -$

\mathbf{P}). Moreover, $2\lambda_{\min}(\mathbb{E}[\mathbf{P}]) \geq \lambda_{\min}(\mathbb{E}[\mathbf{Z}']) \geq \lambda_{\min}(\mathbb{E}[\mathbf{P}])$.

Notice that preserving symmetry yields $\theta = \lambda_{\min}(\mathbb{E}[\mathbf{Z}'])$, which can be up to twice as large as $\lambda_{\min}(\mathbb{E}[\mathbf{P}])$, which is the value of the θ parameter of the method without preserving symmetry. This improved rate is new, and was not present in the algorithm's debut publication [64]. In terms of parameter estimation, once symmetry is not preserved, we fall back onto the setting from Section 10.3.4. Unfortunately, we were not able to quantify the effect of enforcing symmetry on the parameter ν .

Algorithm 23 AMI (Accelerated BFGS Matrix Inversion)

- 1: **Parameters:** $\theta, \nu > 0$, \mathcal{D} = distribution over random linear operators.
 - 2: Choose $\mathbf{X}_0 \in \mathcal{X}$ and set $\mathbf{V}_0 = \mathbf{X}_0$, $\beta = 1 - \sqrt{\frac{\theta}{\nu}}$, $\gamma = \sqrt{\frac{1}{\theta\nu}}$, $\eta = \frac{1}{1+\gamma\nu}$
 - 3: **for** $k = 0, 1, 2, \dots$ **do**
 - 4: $\mathbf{Y}_k = \eta\mathbf{V}_k + (1 - \eta)\mathbf{X}_k$
 - 5: Sample an independent copy $S \sim \mathcal{D}$
 - 6: $\mathbf{X}_{k+1} = \mathbf{Y}_k + (\mathbf{Y}_k\mathbf{A} - \mathbf{I})\mathbf{S}(\mathbf{S}^\top\mathbf{A}\mathbf{S})^{-1}\mathbf{S}^\top - \mathbf{S}(\mathbf{S}^\top\mathbf{A}\mathbf{S})^{-1}\mathbf{S}^\top\mathbf{A}\mathbf{Y}_k$
 - 7: $\quad + \mathbf{S}(\mathbf{S}^\top\mathbf{A}\mathbf{S})^{-1}\mathbf{S}^\top\mathbf{A}\mathbf{Y}_k\mathbf{A}\mathbf{S}(\mathbf{S}^\top\mathbf{A}\mathbf{S})^{-1}\mathbf{S}^\top$
 - 8: $\mathbf{V}_{k+1} = \beta\mathbf{V}_k + (1 - \beta)\mathbf{Y}_k - \gamma(\mathbf{Y}_k - \mathbf{X}_{k+1})$
 - 9: **end for**
-

10.4.2 Vectorizing – a different insight

Define $\text{Vec} : \mathbb{R}^{d \times d} \rightarrow \mathbb{R}^{d^2}$ to be a vectorization operator of column-wise stacking and denote $x \stackrel{\text{def}}{=} \text{Vec}(\mathbf{X})$. It can be shown that the sketch-and-project operation for matrix inversion (10.19) is equivalent to

$$\begin{aligned} x_{k+1} &= \arg \min_{x \in \mathbb{R}^{d^2}} \|x - x_k\|_{\mathbf{A} \otimes \mathbf{A}}^2 \\ \text{subject to } (\mathbf{I} \otimes \mathbf{S}_k^\top)(\mathbf{I} \otimes \mathbf{A})x &= (\mathbf{I} \otimes \mathbf{S}_k^\top)\text{Vec}(\mathbf{I}), \mathbf{C}x = 0, \end{aligned}$$

where \mathbf{C} is defined so that $\mathbf{C}x = 0$ if and only if $\mathbf{X} = \mathbf{X}^\top$. The above is a sketch-and-project update for a linear system in \mathbb{R}^{d^2} , which allows to obtain an alternative proof of Theorem 10.4.1, without using our results from Euclidean spaces. The details are provided in Section J.7.2 of the Appendix.

10.4.3 Accelerated BFGS as an optimization algorithm

As a tweak in the stochastic BFGS allows for a faster estimation of Hessian inverse and therefore more accurate steps of the method, one might wonder if a equivalent tweak might speed up the standard, deterministic BFGS algorithm for solving (10.1). The mentioned tweaked version of standard BFGS is proposed as Algorithm 24. We do not state a convergence theorem for this algorithm—due to the deterministic updates the analysis is currently elusive—nor propose to use it as a default solver, but we rather introduce it as a novel idea for accelerating optimization algorithms. We leave theoretical analysis for the

future work. For now, we perform several numerical experiments, in order to understand the potential and limitations of this new method.

Algorithm 24 BFGS method with accelerated BFGS update for solving (10.1)

- 1: **Parameters:** $\theta, \nu > 0$, stepsize α .
 - 2: Choose $\mathbf{X}_0 \in \mathcal{X}$, w_0 and set $\mathbf{V}_0 = \mathbf{X}_0$, $\beta = 1 - \sqrt{\frac{\theta}{\nu}}$, $\gamma = \sqrt{\frac{1}{\theta\nu}}$, $\eta = \frac{1}{1+\gamma\nu}$.
 - 3: **for** $k = 0, 1, 2, \dots$ **do**
 - 4: $w_{k+1} = w_k - \alpha \mathbf{X}_k \nabla f(w_k)$
 - 5: $s_k = w_{k+1} - w_k$, $\zeta_k = \nabla f(w_{k+1}) - \nabla f(w_k)$
 - 6: $\mathbf{Y}_k = \eta \mathbf{V}_k + (1 - \eta) \mathbf{X}_k$
 - 7: $\mathbf{X}_{k+1} = \frac{\delta_k \delta_k^\top}{\delta_k^\top \zeta_k} + \left(\mathbf{I} - \frac{\delta_k \zeta_k^\top}{\delta_k^\top \zeta_k} \right) \mathbf{Y}_k \left(\mathbf{I} - \frac{\zeta_k \delta_k^\top}{\delta_k^\top \zeta_k} \right)$
 - 8: $\mathbf{V}_{k+1} = \beta \mathbf{V}_k + (1 - \beta) \mathbf{Y}_k - \gamma (\mathbf{Y}_k - \mathbf{X}_{k+1})$
 - 9: **end for**
-

To better understand Algorithm 24, recall that the BFGS updates an estimate of the inverse Hessian via

$$\mathbf{X}_{k+1} = \arg \min_{\mathbf{X}} \|\mathbf{X} - \mathbf{X}_k\|_{F(A)}^2 \quad \text{subject to} \quad \mathbf{X} \delta_k = \zeta_k, \mathbf{X} = \mathbf{X}^\top, \quad (10.24)$$

where $\delta_k = w_{k+1} - w_k$ and $\zeta_k = \nabla f(w_{k+1}) - \nabla f(w_k)$. The above has the following closed form solution $\mathbf{X}_{k+1} = \frac{\delta_k \delta_k^\top}{\delta_k^\top \zeta_k} + \left(\mathbf{I} - \frac{\delta_k \zeta_k^\top}{\delta_k^\top \zeta_k} \right) \mathbf{X}_k \left(\mathbf{I} - \frac{\zeta_k \delta_k^\top}{\delta_k^\top \zeta_k} \right)$. This update appears on line 7 of Algorithm 24 with the difference being that it is applied to a matrix \mathbf{Y}_k .

10.5 Experiments

We perform extensive numerical experiments to bring additional insight to both the performance of and to parameter selection for Algorithms 23 and 24. We first test our accelerated matrix inversion algorithm, and subsequently perform experiments related to Section 10.4.3.

10.5.1 Accelerated matrix inversion

We consider the problem of inverting a symmetric positive matrix A . We focus on a few particular choices of matrices A (specified when describing each experiment), that differ in their eigenvalue spectra. Three different sketching strategies are studied: Coordinate sketches with convenient probabilities ($\mathbf{S} = e_i$ with probability proportional to $\mathbf{A}_{i,i}$), coordinate sketches with uniform probabilities ($\mathbf{S} = e_i$ with probability $\frac{1}{n}$) and Gaussian sketches ($\mathbf{S} \sim \mathcal{N}(0, \mathbf{I})$). As matrices to be inverted, we use both artificially generated matrices with the access to the spectrum and also Hessians of ridge regression problems from LibSVM.

We compare the speed of the accelerated method with pre-computed estimates of the parameters θ, ν to the nonaccelerated method. The pre-computed estimates of θ^P, ν^P

are set as per (10.16):

$$\theta^P = \frac{\lambda_{\min}(\mathbf{A})}{\text{Tr}(\mathbf{A})}, \quad \nu^P = \frac{\text{Tr}(\mathbf{A})}{\min_i(\mathbf{A}_{i,i})},$$

which is the optimal choice for coordinate sketches with convenient probabilities without enforcing symmetry. In practice we might not have an access to $\lambda_{\min}(\mathbf{A})$, thus we cannot compute θ^P exactly. Therefore we also test sensitivity of the algorithm to the choice of parameters, and we run some experiments where we only guess parameter θ^P .

Lastly, the tests are performed on both artificial examples and LibSVM [23] data. We shall also explain the legend of plots: “a” indicates acceleration, “nsym” indicates the algorithm without enforcing symmetry and “h” indicates the setting when ν^P is not known, and a naive heuristic choice is casted.

The first experiment: synthetic and real-world data

Let us start with a simple experiment (Figure 10.1) to give a quick taste of the numerical performance.

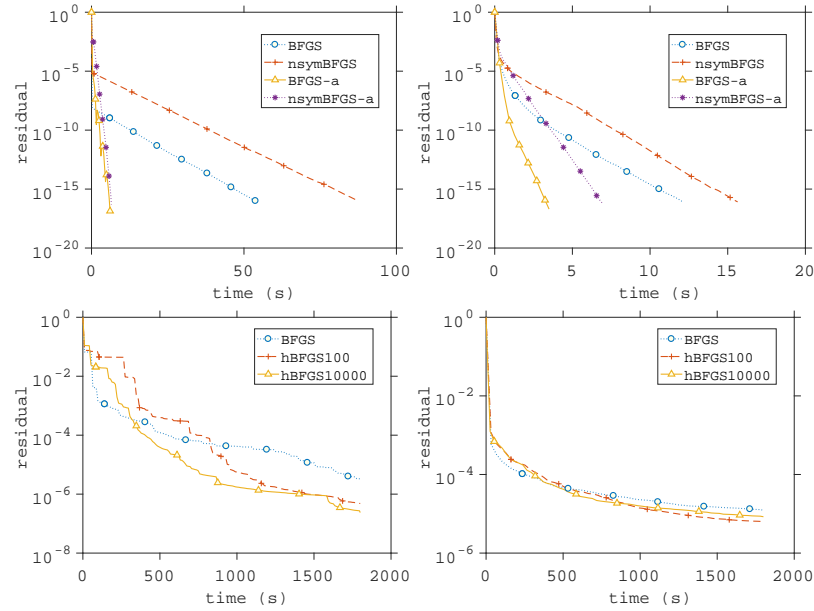


Figure 10.1: Accelerated matrix inversion on synthetic data. From left to right: (i) Eigenvalues of $A \in \mathbb{R}^{100 \times 100}$ are $1, 10^3, 10^3, \dots, 10^3$ and coordinate sketches with convenient probabilities are used. (ii) Eigenvalues of $A \in \mathbb{R}^{100 \times 100}$ are $1, 2, \dots, n$ and Gaussian sketches are used. Label “nsym” indicates non-enforcing symmetry and “-a” indicates acceleration. (iii) Epsilon dataset ($n = 2000$), coordinate sketches with uniform probabilities. (iv) SVHN dataset ($n = 3072$), coordinate sketches with convenient probabilities. Label “h” indicates that λ_{\min} was not precomputed, but θ was chosen as described in the text.

The experiments suggest that once the parameters θ, ν are estimated exactly, we get a speedup comparing to the nonaccelerated method; and the amount of speedup depends

on the structure of A and the sketching strategy. We observe from Figure 10.1 that we gain a great speedup for ill conditioned problems once the eigenvalues are concentrated around the largest eigenvalue. We also observe from Figure 10.1 that enforcing symmetry combines well with θ, ν computed by (10.16), which does not consider the symmetry. On top of that, choice of θ, ν per (10.16) seems to be robust to different sketching strategies, and in worst case performs as fast as the nonaccelerated algorithm.

The second experiment: well understood artificial data

Let us consider inverting the matrix $\mathbf{A} = \eta \mathbf{I} + \beta \mathbf{1}\mathbf{1}^\top$ for $\eta > 0$ and $\beta \geq -\frac{\eta}{n}$ so as in this case we have control over both θ and ν . This artificial example was considered in [207] for solving linear systems. In particular, we show that for coordinate sketches with convenient probabilities (which is indeed the same as uniform probabilities in this example), we have

$$\begin{aligned}\theta^P &\stackrel{\text{def}}{=} \lambda_{\min}(\mathbb{E}[\mathbf{P}]) = \frac{\min(\eta, \eta + n\beta)}{n(\eta + \beta)}, \\ \nu^P &\stackrel{\text{def}}{=} \lambda_{\max}\left(\mathbb{E}\left[\mathbb{E}[\mathbf{P}]^{-\frac{1}{2}} \mathbf{P} \mathbb{E}[\mathbf{P}]^{-1} \mathbf{P} \mathbb{E}[\mathbf{P}]^{-\frac{1}{2}}\right]\right) = n.\end{aligned}$$

Due to the fact that we do not have a theoretical justification of θ, ν for $n > 2$ when enforcing symmetry, we set $\theta = \theta^P$ and $\nu = \nu^P$ for Gaussian sketches as well.

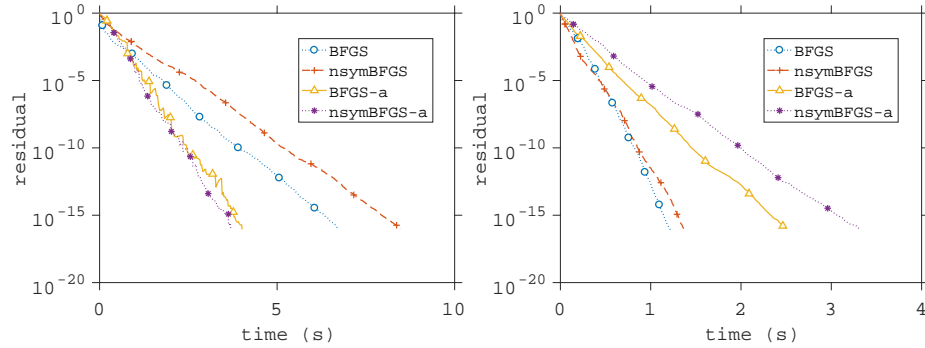


Figure 10.2: Accelerated matrix inversion on synthetic data. Parameter choice: $\eta = 1 + 10^{-1}, \beta = -n^{-1}, n = 100$. From left to right we have: Coordinate sketch with uniform (convenient) probabilities and Gaussian sketch respectively.

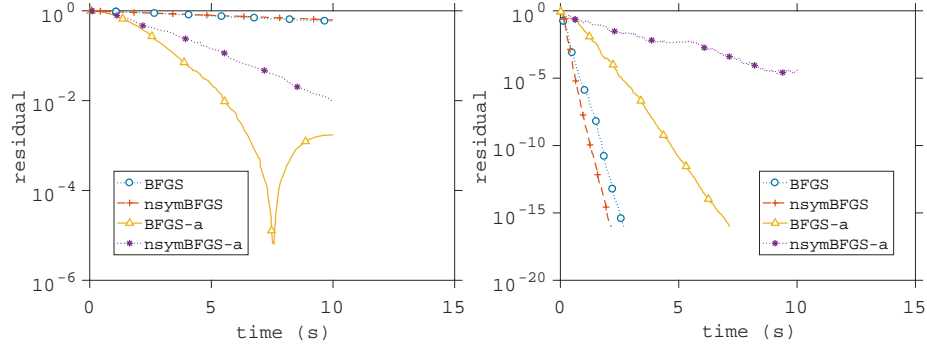


Figure 10.3: Accelerated matrix inversion on synthetic data. Parameter choice: $\eta = 1 + 10^{-3}$, $\beta = -n^{-1}$, $n = 100$. From left to right we have: Coordinate sketch with uniform (convenient) probabilities and Gaussian sketch respectively.

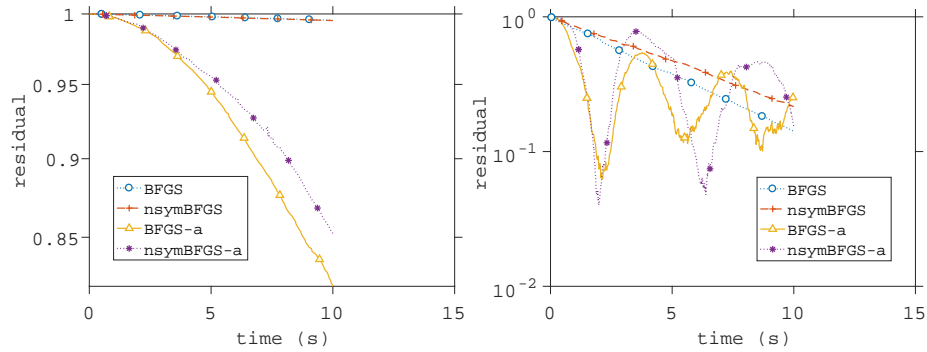


Figure 10.4: Accelerated matrix inversion on synthetic data. Parameter choice: $\eta = 1 + 10^{-5}$, $\beta = -n^{-1}$, $n = 100$. From left to right we have: Coordinate sketch with uniform (convenient) probabilities and Gaussian sketch, respectively.

As expected from the theory, as the matrix to be inverted becomes more ill conditioned, the accelerated method performs significantly better compared to the nonaccelerated method for coordinate sketches. In fact, an arbitrary speedup can be obtained by setting $\beta = -n^{-1}$ and $\eta \rightarrow 1$ for the coordinate sketches setup. On the other hand, Gaussian sketches report the slowing of the algorithm, most likely caused by the fact that the theoretical parameters θ, ν for Gaussian sketches with enforced symmetry are different to θ^P, ν^P , which are estimated for coordinate sketches without enforced symmetry. In the case of coordinate sketches with symmetry enforced, we suspect a great speedup even though the parameters θ, ν were set to θ^P, ν^P .

The third experiment: more complex artificial data

We randomly generate an orthonormal matrix \mathbf{U} , choose diagonal matrix \mathbf{D} , and set $\mathbf{A} = \mathbf{U}\mathbf{D}\mathbf{U}^\top$. Clearly, diagonal elements of \mathbf{D} are eigenvalues of \mathbf{A} . We set them in the following way:

- Uniform grid. The eigenvalues are set to $1, 2, \dots, n$.

- One small, the rest larger. The smallest eigenvalue is 1, remaining eigenvalues are all 10 in the first example, all 100 in the second example and all 1000 in the third example in this category.
- One large, the rest small. The largest eigenvalue is 10^4 , the remaining eigenvalues are all 1.

Firstly, consider coordinate sketches with convenient probabilities. Notice that we can easily estimate ν^P, θ^P due to the results from Section 10.3.4 since we have control of $\lambda_{\min}(\mathbf{A})$ and therefore also of θ . Therefore, we set $\theta = \theta^P = \min \mathbf{D}_{i,i}$ and $\nu = \nu^P$ for Algorithm 23. Then, we consider coordinate sketches with uniform probabilities and Gaussian sketches. In both cases, we set the parameters θ, ν as for coordinate sketches with convenient probabilities.

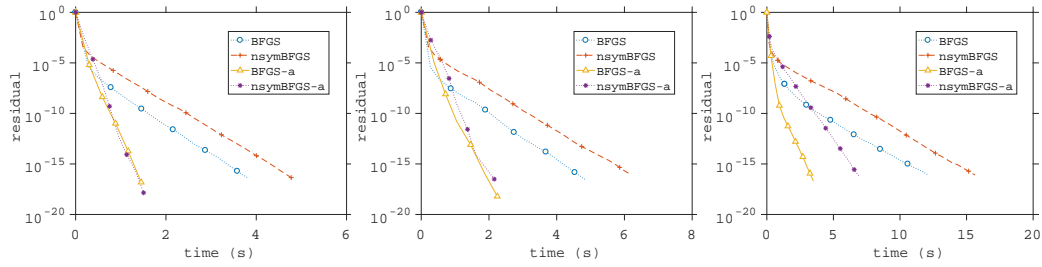


Figure 10.5: Eigenvalues set to $1, 2, 3, \dots, n$. From left to right we have: Coordinate sketch with convenient probabilities, coordinate sketch with uniform probabilities and Gaussian sketch respectively.

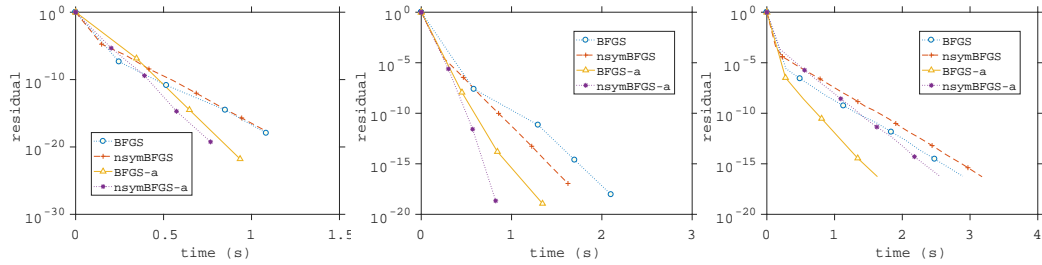


Figure 10.6: Eigenvalues set to $1, 10, 10, \dots, 10$. From left to right we have: Coordinate sketch with convenient probabilities, coordinate sketch with uniform probabilities and Gaussian sketch respectively.

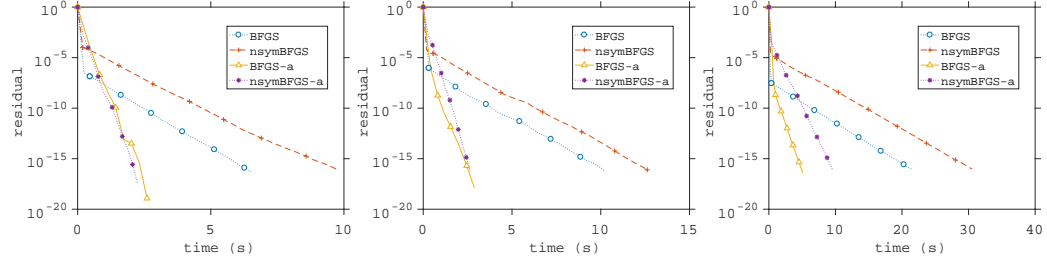


Figure 10.7: Accelerated matrix inversion on synthetic data. Eigenvalues set to 1, 100, 100, ..., 100. From left to right we have: Coordinate sketch with convenient probabilities, coordinate sketch with uniform probabilities and Gaussian sketch respectively.

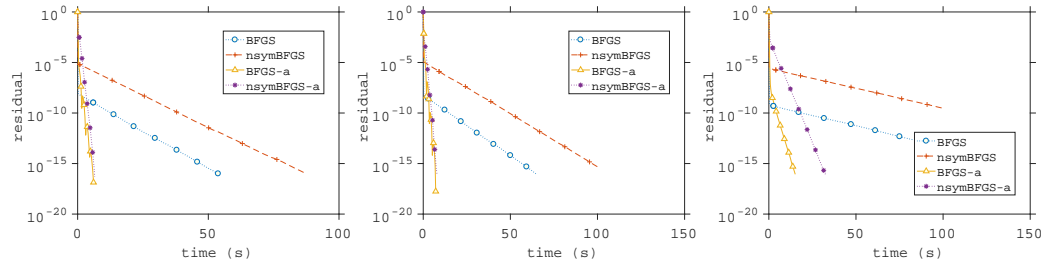


Figure 10.8: Accelerated matrix inversion on synthetic data. Eigenvalues set to 1, 1000, 1000, ..., 1000. From left to right we have: Coordinate sketch with convenient probabilities, coordinate sketch with uniform probabilities and Gaussian sketch respectively.

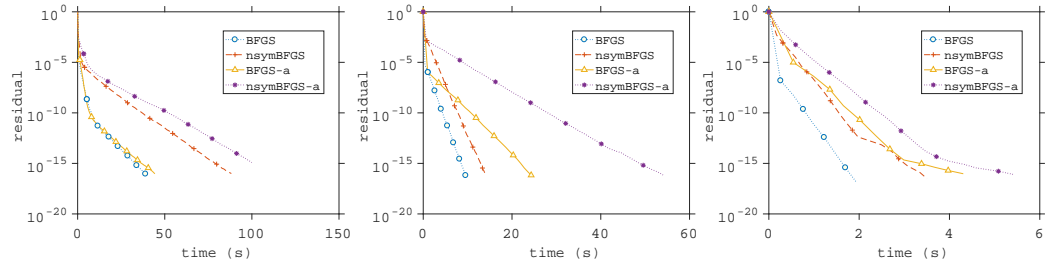


Figure 10.9: Accelerated matrix inversion on synthetic data. Eigenvalues set to 10000, 1, 1, ..., 1. From left to right we have: Coordinate sketch with convenient probabilities, coordinate sketch with uniform probabilities and Gaussian sketch respectively.

The numerical experiments in this section indicate that one might choose θ, ν as per Section 10.3.4. In other words, one might pretend to be in the setting when symmetry is not enforced and coordinate sketches with convenient probabilities are used. In fact, the practical speedup coming from the acceleration depends very strongly on the structure of matrix \mathbf{A} . Another message to be delivered is that both preserving symmetry and acceleration yield a better convergence and they combine together well.

We also consider a problem where we pretend to not have access to $\lambda_{\min}(\mathbf{A})$, therefore we cannot choose $\theta = \theta^P$. Instead, we naively choose $\theta = \frac{1}{100\nu}$ and $\theta = \frac{1}{10000\nu}$.

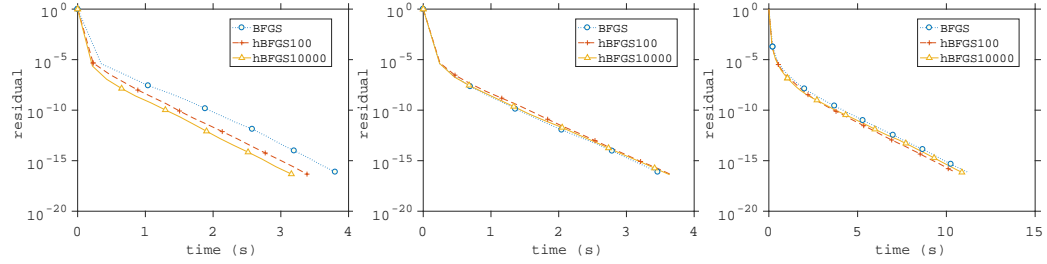


Figure 10.10: Accelerated matrix inversion on synthetic data. Eigenvalues set to $1, 2, \dots, n$. From left to right we have: Coordinate sketch with convenient probabilities, coordinate sketch with uniform probabilities and Gaussian sketch respectively.

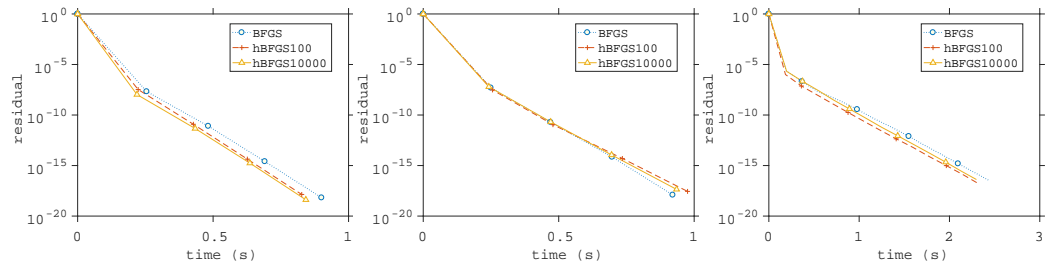


Figure 10.11: Accelerated matrix inversion on synthetic data. Eigenvalues set to $1, 10, 10, \dots, 10$. Coordinate sketch with convenient probabilities, coordinate sketch with uniform probabilities and Gaussian sketch respectively.

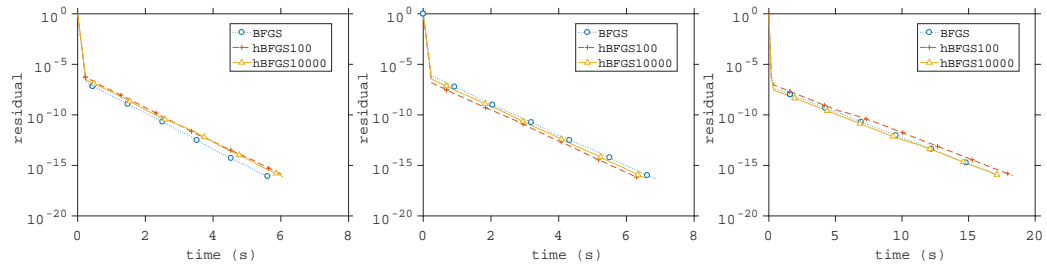


Figure 10.12: Accelerated matrix inversion on synthetic data. Eigenvalues set to $1, 100, 100, \dots, 100$. From left to right we have: Coordinate sketch with convenient probabilities, coordinate sketch with uniform probabilities and Gaussian sketch respectively.

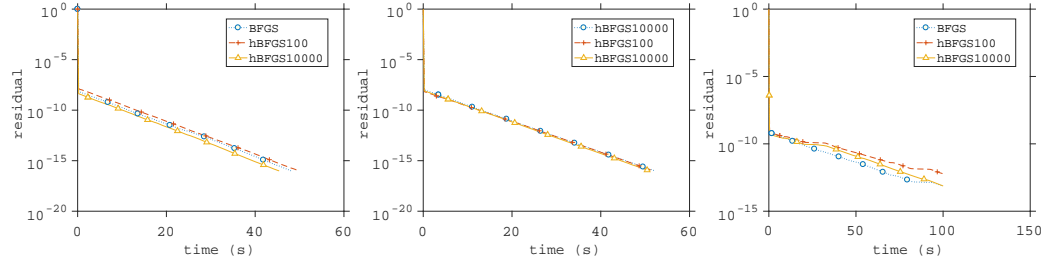


Figure 10.13: Accelerated matrix inversion on synthetic data. Eigenvalues set to 1, 1000, 1000, ..., 1000. From left to right we have: Coordinate sketch with convenient probabilities, coordinate sketch with uniform probabilities and Gaussian sketch respectively.

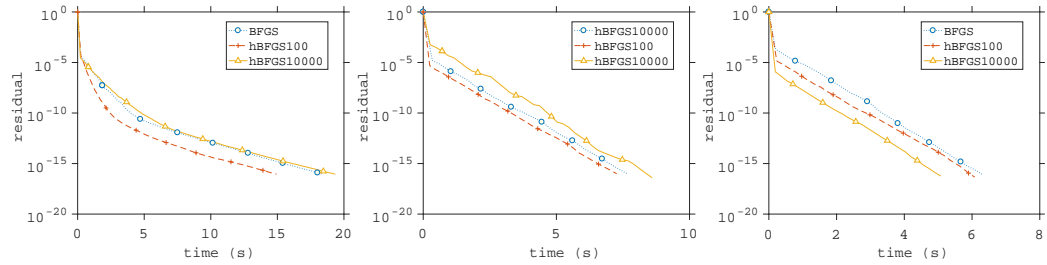


Figure 10.14: Accelerated matrix inversion on synthetic data. Eigenvalues set to 10000, 1, 1, ..., 1. From left to right we have: Coordinate sketch with convenient probabilities, coordinate sketch with uniform probabilities and Gaussian sketch respectively.

Notice that once the acceleration parameters are not set exactly (but they are still reasonable), we observe that the performance of the accelerated algorithm is essentially the same as the performance of the nonaccelerated algorithm. We have observed the similar behavior when setting $\theta = \theta^P$ for Gaussian sketches.

The fifth experiment: LibSVM data

Next we investigate if the accelerated BFGS update improves upon the standard BFGS update when applied to the Hessian $\nabla^2 f(x)$ of ridge regression problems of the form

$$\min_{x \in \mathbb{R}^d} f(x) \stackrel{\text{def}}{=} \frac{1}{2} \|\mathbf{A}x - b\|_2^2 + \frac{\lambda}{2} \|x\|_2^2, \quad \nabla^2 f(x) = \mathbf{A}^\top \mathbf{A} + \lambda \mathbf{I}, \quad (10.25)$$

using data from LibSVM [23]. Datapoints (rows of \mathbf{A}) were normalized such that $\|\mathbf{A}_{i,:}\|^2 = 1$ for all i and the regularization parameter was chosen as $\lambda = \frac{1}{m}$.

First, we run the experiments on smaller problems when parameters θ , ν are precomputed for coordinate sketches with convenient probabilities (10.16).

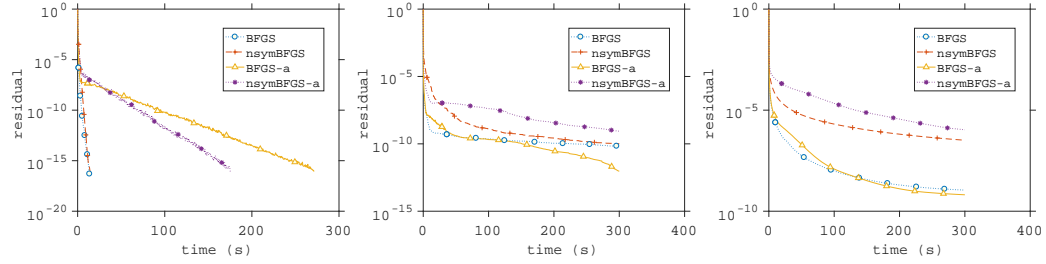


Figure 10.15: Accelerated matrix inversion on real data. Dataset aloi: $n = 128$. From left to right we have: Coordinate sketch with convenient probabilities, coordinate sketch with uniform probabilities and Gaussian sketch respectively.

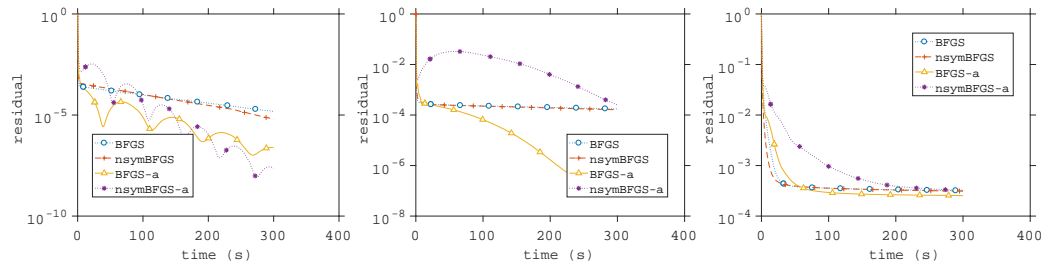


Figure 10.16: Accelerated matrix inversion on real data. Dataset w1a: $n = 300$. From left to right we have: Coordinate sketch with convenient probabilities, coordinate sketch with uniform probabilities and Gaussian sketch respectively.

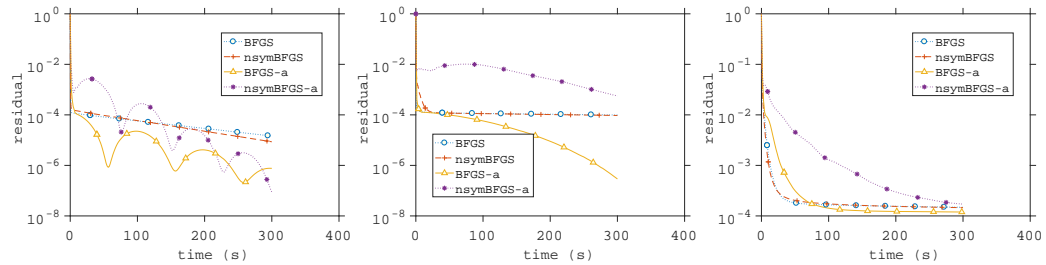


Figure 10.17: Accelerated matrix inversion on real data. Dataset w2a: $n = 300$. From left to right we have: Coordinate sketch with convenient probabilities, coordinate sketch with uniform probabilities and Gaussian sketch respectively.

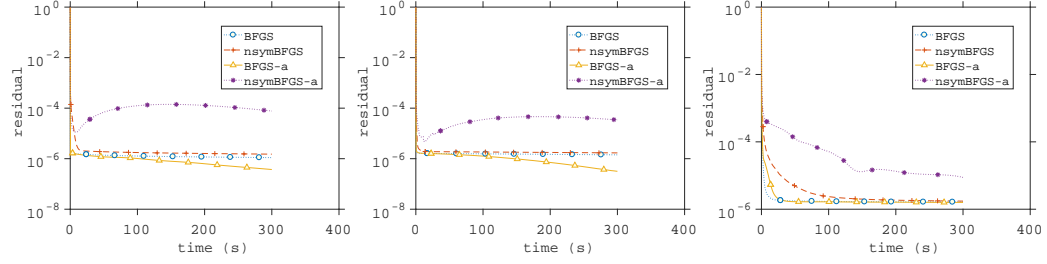


Figure 10.18: Accelerated matrix inversion on real data. Dataset mushrooms: $n = 112$. From left to right we have: Coordinate sketch with convenient probabilities, coordinate sketch with uniform probabilities and Gaussian sketch respectively.

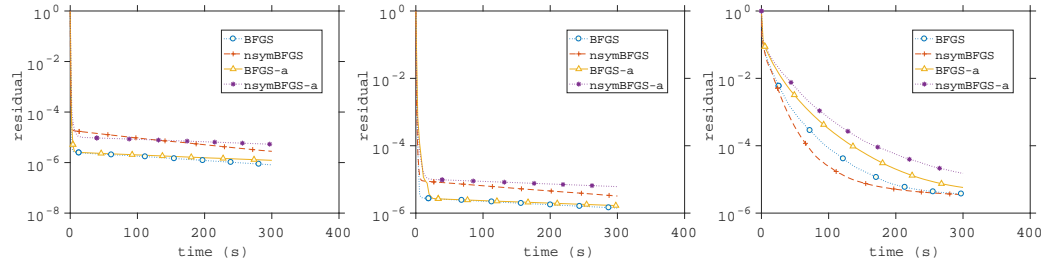


Figure 10.19: Accelerated matrix inversion on real data. Dataset protein: $n = 357$. From left to right we have: Coordinate sketch with convenient probabilities, coordinate sketch with uniform probabilities and Gaussian sketch respectively.

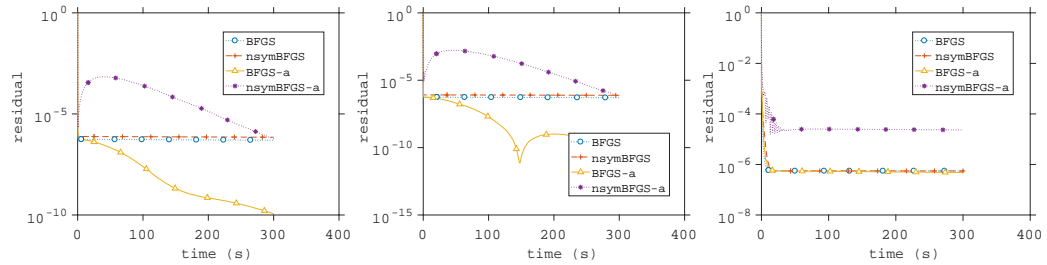


Figure 10.20: Accelerated matrix inversion on real data. Dataset phishing: $n = 68$. From left to right we have: Coordinate sketch with convenient probabilities, coordinate sketch with uniform probabilities and Gaussian sketch respectively.

In the vast majority of examples, the accelerated method performed significantly better than the nonaccelerated method for coordinate sketches (with both convenient and uniform probabilities), however the methods were comparable for Gaussian sketches. We believe that this is due to the fact that choice of parameters as per (10.16) is close to the optimal parameters for coordinate sketches, and further for Gaussian sketches. However, the experiments on coordinate sketches indicates that for some classes of problems, accelerated algorithms with finely tuned parameters bring a great speedup compared to nonaccelerated ones.

We also consider a problem where we do not compute $\lambda_{\min}(\mathbf{A})$, and therefore we cannot choose $\theta = \theta^P$ in (10.16). Instead, we choose $\theta = \frac{1}{100\nu}$ and $\theta = \frac{1}{10000\nu}$.

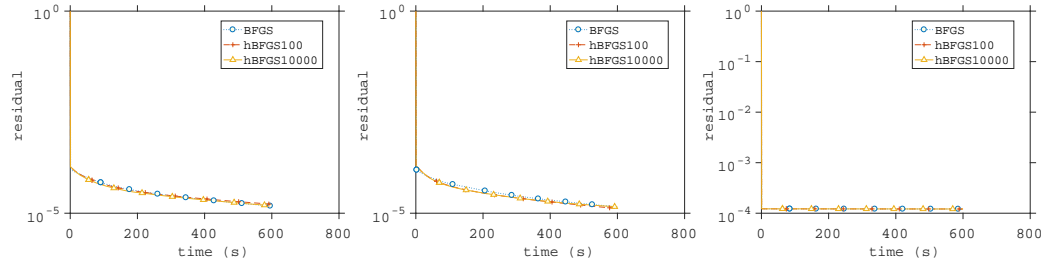


Figure 10.21: Accelerated matrix inversion on real data. Dataset madelon: $n = 500$. From left to right we have: Coordinate sketch with convenient probabilities, coordinate sketch with uniform probabilities and Gaussian sketch respectively.

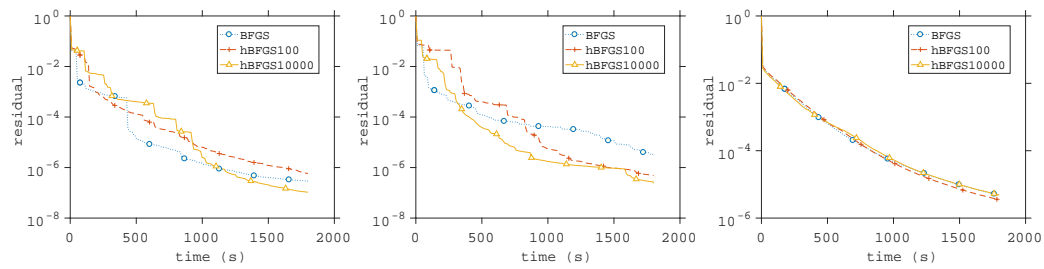


Figure 10.22: Accelerated matrix inversion on real data. Dataset epsilon: $n = 2000$. From left to right we have: Coordinate sketch with convenient probabilities, coordinate sketch with uniform probabilities and Gaussian sketch respectively.

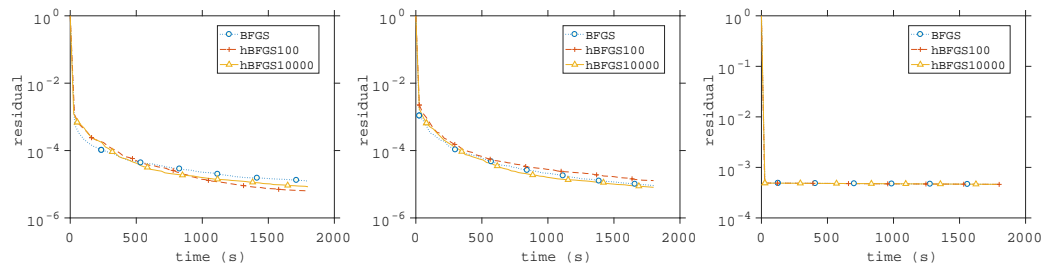


Figure 10.23: Accelerated matrix inversion on real data. Dataset svhn: $n = 3072$. From left to right we have: Coordinate sketch with convenient probabilities, coordinate sketch with uniform probabilities and Gaussian sketch respectively.

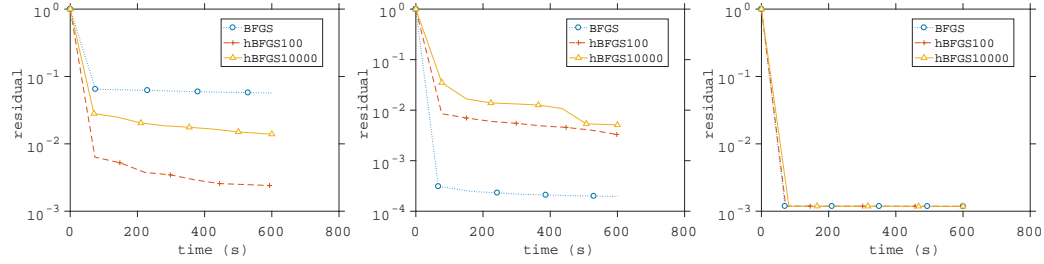


Figure 10.24: Accelerated matrix inversion on real data. Dataset gisette: $n = 5000$. From left to right we have: Coordinate sketch with convenient probabilities, coordinate sketch with uniform probabilities and Gaussian sketch respectively.

Notice that once the acceleration parameters are not set exactly (but they are still reasonable), we observe that the performance of the accelerated algorithm is essentially the same as the performance of the nonaccelerated algorithm, which is essentially the same conclusion as for artificially generated examples.

The fourth experiment: sensitivity to the acceleration parameters

Here we investigate the sensitivity of the accelerated BFGS to the parameters θ and ν . First we compute ν^P, θ^P and from this we extract the following exponential grids: $\theta_i = 2^{i-4}\theta$ and $\nu_i = 5^{i-4}\nu$ for $i = 1, 2, \dots, 7$. To gauge the gain is using acceleration with a particular (θ, ν) pair, we run the accelerated algorithm for a fixed time then store the error of the final iterate. We then compute average per iteration decrease and divide it by average per iteration decrease of nonaccelerated algorithm. Thus if the resulting difference is less than one, then the accelerated algorithm was faster to nonaccelerated.

In the plots below, $n = 200$ was chosen. We focused on 2 problems described in the previous section—when the eigenvalues are uniformly distributed and when the the largest eigenvalue have multiplicity $n - 1$.

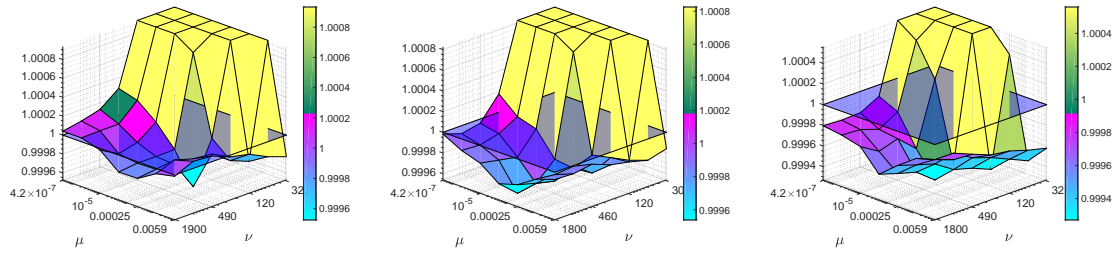


Figure 10.25: Accelerated matrix inversion on synthetic data. Sensitivity to acceleration parameters. Eigenvalues of A are set to $1, 2, \dots, n$. From left to right we have: Coordinate sketches with convenient probabilities, coordinate sketches with uniform probabilities and Gaussian sketches. Choice of parameters as per (10.16) in the middle of plots. Each instance was run for 5 seconds.

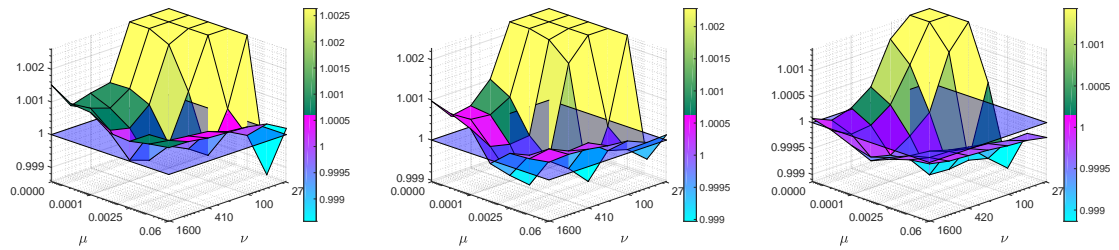


Figure 10.26: Accelerated matrix inversion on synthetic data. Sensitivity to acceleration parameters. Eigenvalues of A are set to $1, 10, 10, \dots, 10$. From left to right we have: Coordinate sketches with convenient probabilities, coordinate sketches with uniform probabilities and Gaussian sketches. Choice of parameters as per (10.16) in the middle of plots. Each instance was run for 2 seconds.

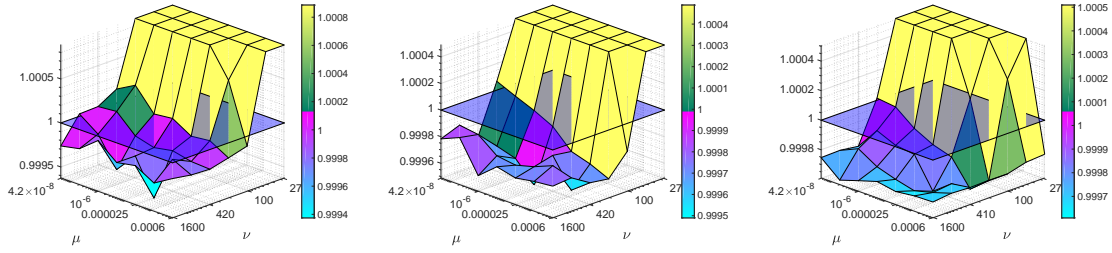


Figure 10.27: Accelerated matrix inversion on synthetic data. Sensitivity to acceleration parameters. Eigenvalues of \mathbf{A} are set to 1, 1000, 1000, \dots , 1000. From left to right we have: Coordinate sketches with convenient probabilities, coordiante sketches with uniform probabilities and Gaussian sketches. Choice of parameters as per (10.16) in the middle of plots. Each instance was run for 10 seconds.

The crucial aspect to make the accelerated algorithm to converge is to set ν large enough. In fact, combination of both small ν and small θ leads almost always to non-convergent algorithm. On the other hand, it seems that once ν is chosen correctly, big enough θ leads to fast convergence. This indicates how to compute θ in practice (recall that computing ν is feasible)—one needs just to choose it small enough (definitely smaller than $\frac{1}{\nu}$).

10.5.2 BFGS optimization method

We test Algorithm 24 on several logistic regression problems using data from LibSVM [23]. In all our tests we centered and normalized the data, included a bias term (a linear intercept), and choose the regularization parameter as $\lambda = 1/m$, where m is the number of data points. To keep things as simple as possible, we also used a fixed stepsize which was determined using grid search. Since our theory regarding the choice for the parameters θ and ν does not apply in this setting, we simply probed the space of parameters manually and reported the best found result, see Figure 10.28. In the legend we use BFGS-a- θ - ν to denote the accelerated BFGS method (Algorithm 24) with parameters θ and ν .

On all four datasets, our method outperforms the classic BFGS method, indicating that replacing classic BFGS update rules for learning the inverse Hessian by our new accelerated rules can be beneficial in practice.

Much like the phishing problem in Figure 10.28, the problems *madelon*, *covtype* and *a9a* in Figure 10.29 did not benefit that much from acceleration.

Indeed, we found in our experiments that even when choosing extreme values of θ and ν , the generated inverse Hessian would not significantly deviate from the estimate that one would obtain using the standard BFGS update. Thus on these two problems there is apparently little room for improvement by using acceleration.

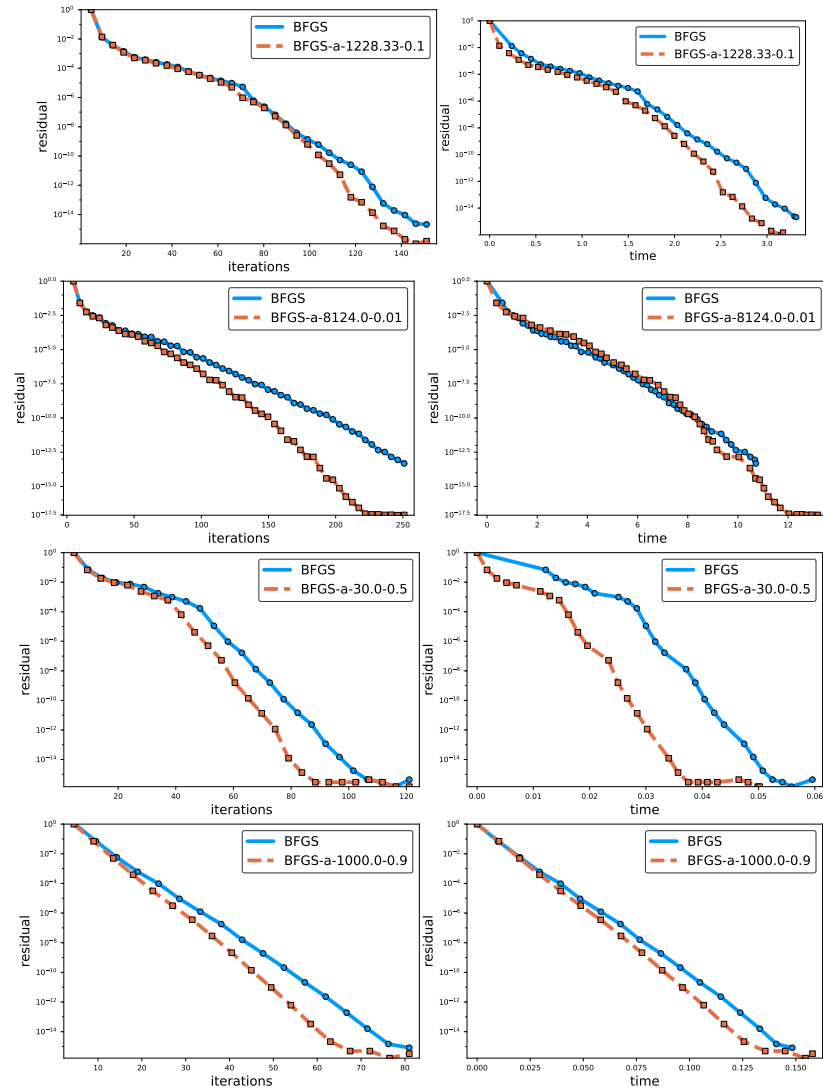


Figure 10.28: Algorithm 24 (BFGS with accelerated matrix inversion quasi-Newton update) vs standard BFGS. Left column: time, right column: iteration. From top to bottom: phishing, mushrooms, australian and splice dataset.

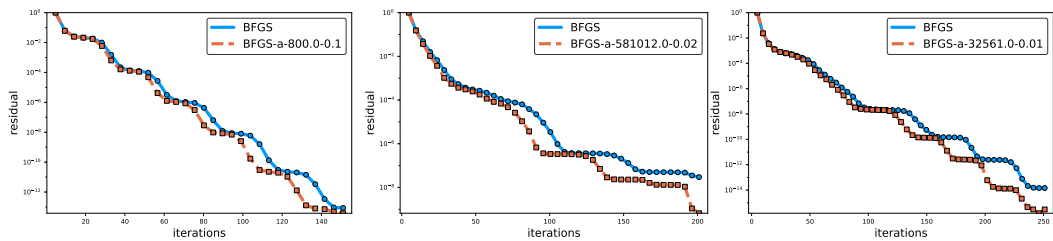


Figure 10.29: Accelerated BFGS applied on real data. Left to right: modeln, covtype, a9a

10.6 Conclusion

In this chapter, we developed an accelerated sketch-and-project method for solving linear systems in Euclidean spaces. The method was applied to invert positive definite matrices, while keeping their symmetric structure. Our accelerated matrix inversion algorithm was then incorporated into an optimization framework to develop both accelerated stochastic and deterministic BFGS, which to the best of our knowledge, are *the first accelerated quasi-Newton updates*.

We show that under a careful choice of the parameters of the method, and depending on the problem structure and conditioning, acceleration might result into significant speedups both for the matrix inversion problem and for the stochastic BFGS algorithm. We confirm experimentally that our accelerated methods can lead to speed-ups when compared to the classical BFGS algorithm.

As a future line of research, it might be interesting to study the accelerated BFGS algorithm (either deterministic or stochastic) further, and provide a convergence analysis on a suitable class of functions. Another interesting area of research might be to combine accelerated BFGS with limited memory [124] or engineer the method so that it can efficiently compete with first-order algorithms for some empirical risk minimization problems, such as, for example [57].

As we show in this work, *Nesterov's acceleration can be applied to quasi-Newton updates*. We believe this is a surprising fact, as quasi-Newton updates have not been understood as optimization algorithms, which prevented the idea of applying acceleration in this context.

Since second-order methods are becoming more and more ubiquitous in machine learning and data science, we hope that our work will motivate further advances at the frontiers of big data optimization.

Note that this whole chapter was devoted to accelerating an algorithm for solving linear systems, and applying the obtained knowledge to obtain an algorithm to solve general convex optimization (10.1). In the next chapter, we will tackle problem (10.1) directly, developing a specific accelerated stochastic algorithm.

Chapter 11

Concluding Remarks

11.1 Summary

In this work, we developed a number of stochastic iterative optimization algorithms with primary focus on solving supervised machine learning problems cast in the form of regularized empirical risk minimization problems of the form

$$\min_{x \in \mathbb{R}^d} \left\{ F(x) \stackrel{\text{def}}{=} \frac{1}{n} \sum_{i=1}^n f_i(x) + \psi(x) \right\}. \quad (11.1)$$

$\underbrace{\hspace{10em}}_{\stackrel{\text{def}}{=} f(x)}$

Each chapter of the thesis introduced a state-of-the-art approach for solving (11.1) under further assumptions on both the problem structure and the oracle model.

In Chapter 2, we considered an instance of (11.1) in the regime where the dimension d is very large, and in the simplified setting with $n = 1$ and $\psi \equiv 0$. In high-dimensional optimization, variants of coordinate descent methods reign supreme. In this chapter, we proposed an accelerated coordinate descent (ACD) method, explicitly allowing for *arbitrary* sampling. This generalized previous results which considered sampling of a single coordinate only. We further designed a novel non-uniform minibatch sampling strategy can provably outperform uniform minibatch sampling, which is the first result of its kind in the literature. The mentioned sampling can be applied in many contexts beyond coordinate descent methods, as demonstrated in Chapters 5 and 6 where we applied this sampling in the context of stochastic gradient descent methods.

In Chapter 3, we considered problem (11.1) in the $n = 1$ setting, also assuming that d is very large. As mentioned above, in this regime, randomized CD methods are the state of the art. However, as they may not converge otherwise, these methods are always studied either in the nonregularized regime, or with a separable regularizer. With a goal to lift this fundamental limitation, in this chapter we designed new variants of CD methods which can provably work with any, even nonseparable, regularizer ψ . We further consider a more general subspace gradient oracle: we allow our method access to gradients of f over a random subspace of \mathbb{R}^d with a very general notion of randomness going further than that used in randomized CD methods. Our algorithm, SEGA, can be interpreted as a variance-reduced CD method, with a novel variance reduction strategy aimed at removing the adverse effect the (nonseparable) regularizer has on classical CD methods. SEGA is the first CD-type method capable of converging even with nonseparable regularizers. Moreover,

when specialized to random subspaces spanned coordinate vectors, our convergence results for SEGA match the state-of-the-art convergence rates of CD methods, up to a small constant factor (less than 10). For instance, SEGA can be coupled with both importance sampling and Nesterov’s acceleration, achieving rates similar to those of the ACD method developed in Chapter 2.

In Chapter 4, we considered a distributed optimization problem with n workers and a centralized parameter server of the form (11.1), with f_i interpreted as the loss of model x on the data stored on the i th worker. We proposed a generic technique for reducing worker→server communication in several popular iterative methods for solving (11.1) by a factor up to n , without hurting the convergence rate by more than a small constant. The key idea is based on combining the underlying iterative solver with a novel independent coordinate descent / random sparsification mechanism. We demonstrated that our approach can be incorporated on top of distributed implementations of various algorithms such as GD, SGD, or SAGA.

In Chapter 5, we proposed and analyzed a remarkably general randomized algorithm for solving (11.1)—Generalized Jacobian Sketching method (GJS)—capable of reducing variance coming both from subsampling the data and parameters. GJS is the first variance reduced method with this property. This was enabled by studying gradient estimators arising from arbitrary sketching operators applied to the Jacobian matrix of the mapping $x \mapsto (f_1(x), \dots, f_n(x))$. GJS recovers many well-known and recently developed algorithms as a special case, including SAGA [37, 165], LSVRG [83, 106], SEGA from Chapter 3 and ISEGA from Chapter 4. Remarkably, our convergence theory for GJS either recovers the best-known convergence rate in each special case, or improves upon the current best rates. Besides the unification and improvement upon the well-established algorithms, GJS can specialize to a large number of new specific methods with intriguing properties.

In Chapter 6, we go even one step further in our unification efforts. In particular, we proposed a framework capable of analyzing both variance-reduced and non-variance reduced variants of SGD at the same time. This is the first framework with this property. Further, our framework is capable of accurate modeling standard, parallel and distributed SGD methods, with gradient estimator formed through various mechanisms, including subsampling (e.g., minibatching and importance sampling) and compression (e.g., sparsification and quantization) and their combination. The development of our framework was motivated by the need establish a general theory capable of taming large swaths of the almost impenetrable wilderness of SGD methods, while at the same time facilitating faster development of new variants. Our framework includes GJS as a special case, as well as many other methods.

In Chapter 7, we establish a novel and fundamental link between the novel variance reduced CD methods first developed in this thesis, and the world of SGD methods for finite-sum optimization. In particular, we were able to show that our SEGA (resp. SVRCD¹) method reduces to the well-known SAGA (resp. LSVRG) method when applied to a carefully constructed problem with a special regularizer. Moreover, our general theory is able to recover the best best-known oracle complexity for these methods. We have also tightened the analysis of both SEGA and SVRCD, so that these methods are capable of exploiting the

¹SVRCD is an algorithm similar to SEGA, proposed in Chapter 5.

structure of the regularizer ψ for faster convergence. Further, we succeeded in incorporating Nesterov’s momentum into the SVRCD algorithm. The resulting method—ASVRCD—is the first accelerated variance reduced coordinate descent method. Moreover, in a special case, ASVRCD reduces to a (variant of) the celebrated Katyusha algorithm [4, 164], thus achieving the optimal rate for finite-sum problems.

In Chapter 8, we introduced a novel optimization formulation of federated learning aimed at allowing device-specific personalization. Unlike the standard federated learning formulation which seeks to find a single global model to be used on all devices, we constructed a separate loss for each device, with an extra penalty that ensures the local models do not deviate too much from each other. We first applied standard SGD to our two-sum formulation, and observed that the resulting method is a novel variant of the celebrated local gradient descent method. However, in sharp contrast with all preceding formulations of federated learning, we showed that local methods need fewer communication rounds if more personalization is desired. This is the first result in the literature suggesting that local GD methods can lead to communication complexity benefits. However, we went much beyond this in the chapter, proposing a number of new methods capable of working with a regularizer, capable of achieving variance reduction and partial participation.

In Chapter 9, we proposed a second-order subspace descent algorithm (SSCN) designed to solve (11.1) with $n = 1$, separable ψ , and large d . We proved that SSCN enjoys a global convergence rate and a fast local convergence rate. While our global result interpolates between the rate of CD and the rate of the cubically regularized Newton method of Polyak and Nesterov, the local rate is identical to the convergence of stochastic subspace descent applied to minimizing quadratic function $\frac{1}{2}(x - x^*)^\top \nabla^2 f(x^*)(x - x^*)$, which we find remarkable.

In Chapter 10 we developed an accelerated stochastic algorithm for solving linear systems in Euclidean spaces. Our method can be specialized to obtain a large class of accelerated algorithms designed to invert a positive definite matrix. In particular, we were able to accelerate the subroutine employed in stochastic quasi-Newton methods which updates the inverse Hessian estimator. Despite more than half a century of research into quasi-Newton methods, we have developed the first provably accelerated quasi-Newton matrix update formula.

11.2 Future Research Work

In this section, we outline a few challenges that remain open problems to be addressed in the future.

- **Unified framework for accelerated stochastic algorithms.** We believe that an accelerated variant of the GJS algorithm and/or a general analysis of accelerated stochastic algorithms analogous to Chapter 6 would be of immense value. Such results would immediately lead to countless optimal algorithms in terms of oracle complexity.
- **Understanding Nesterov’s acceleration.** Throughout this thesis, we incorporated some form of Nesterov’s acceleration into several algorithms. The effect of

the acceleration mechanism, however, slightly varied from chapter to chapter.² This can be well demonstrated comparing the effect of the sampling on ACD and ASVRCD. Specifically, the optimal sampling for SVRCD and ASVRCD are identical, while the optimal sampling for CD and ACD are vastly different. The effect of the acceleration on the sketch-and-project algorithm studied in Chapter 10 is even more complex. We are certain that a better understanding of randomized algorithms with acceleration would enable the development of a broader range of methods, such as accelerated SSCN, for example.

- **SSCN with non-separable ψ .** In Chapter 3, we discovered a mechanism allowing CD algorithms to deal with a non-separable regularizer ψ . Can a similar result be established for second-order algorithms? We believe that control variates—a tool from statistics widely used throughout this work—might help to resolve the issue. However, to prove a tight convergence rate of such a method is highly non-trivial, especially since the literature on variance-reduced second-order methods is very limited at the moment.

²This is consistent with the related literature.

REFERENCES

- [1] Naman Agarwal, Brian Bullins, and Elad Hazan. Second-order stochastic optimization for machine learning in linear time. *The Journal of Machine Learning Research*, 18(1):4148–4187, 2017.
- [2] Dan Alistarh, Demjan Grubić, Jerry Li, Ryota Tomioka, and Milan Vojnović. QSGD: Communication-efficient SGD via gradient quantization and encoding. In *Advances in Neural Information Processing Systems*, pages 1709–1720, 2017.
- [3] Dan Alistarh, Torsten Hoefler, Mikael Johansson, Nikola Konstantinov, Sarit Khirirat, and Cédric Renggli. The convergence of sparsified gradient methods. In *Advances in Neural Information Processing Systems*, pages 5977–5987, 2018.
- [4] Zeyuan Allen-Zhu. Katyusha: The first direct acceleration of stochastic gradient methods. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, pages 1200–1205. ACM, 2017.
- [5] Zeyuan Allen-Zhu and Elad Hazan. Variance reduction for faster non-convex optimization. In *International Conference on Machine Learning*, pages 699–707, 2016.
- [6] Zeyuan Allen-Zhu and Lorenzo Orecchia. Linear coupling: An ultimate unification of gradient and mirror descent. In *International Conference on Machine Learning*, 2017.
- [7] Zeyuan Allen-Zhu, Zheng Qu, Peter Richtárik, and Yang Yuan. Even faster accelerated coordinate descent using non-uniform sampling. In *International Conference on Machine Learning*, pages 1110–1119, 2016.
- [8] Amir Beck. *First-Order Methods in Optimization*. MOS-SIAM Series on Optimization. SIAM, 2017.
- [9] Amir Beck and Marc Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences*, 2(1):183–202, 2009.
- [10] Albert S Berahas, Raghu Bollapragada, and Jorge Nocedal. An investigation of Newton-sketch and subsampled Newton methods. *Optimization Methods and Software*, pages 1–20, 2020.

- [11] Albert S Berahas, Jorge Nocedal, and Martin Takáč. A multi-batch L-BFGS method for machine learning. In *Advances in Neural Information Processing Systems*, pages 1055–1063, 2016.
- [12] El Houcine Bergou, Eduard Gorbunov, and Peter Richtarik. Stochastic three points method for unconstrained smooth minimization. *SIAM Journal on Optimization (to appear)*, 2020.
- [13] Jeremy Bernstein, Jiawei Zhao, Kamyar Azizzadenesheli, and Anima Anandkumar. SignSGD with majority vote is communication efficient and Byzantine fault tolerant. In *International Conference on Learning Representations*, 2019.
- [14] Charles G Broyden. Quasi-Newton methods and their application to function minimisation. *Mathematics of Computation*, 21(99):368–381, 1967.
- [15] Richard H Byrd, Gillian M Chin, Will Neveitt, and Jorge Nocedal. On the use of stochastic hessian information in optimization methods for machine learning. *SIAM Journal on Optimization*, 21(3):977–995, 2011.
- [16] Richard H Byrd, Samantha L Hansen, Jorge Nocedal, and Yoram Singer. A stochastic quasi-Newton method for large-scale optimization. *SIAM Journal on Optimization*, 26(2):1008–1031, 2016.
- [17] Yair Carmon and John Duchi. Gradient descent finds the cubic-regularized nonconvex Newton step. *SIAM Journal on Optimization*, 29(3):2146–2178, 2019.
- [18] Coralia Cartis, Nicholas IM Gould, and Philippe L Toint. Adaptive cubic regularisation methods for unconstrained optimization. Part I: motivation, convergence and numerical results. *Mathematical Programming*, 127(2):245–295, 2011.
- [19] Coralia Cartis, Nicholas IM Gould, and Philippe L Toint. Adaptive cubic regularisation methods for unconstrained optimization. Part II: worst-case function- and derivative-evaluation complexity. *Mathematical Programming*, 130(2):295–319, 2011.
- [20] Coralia Cartis and Katya Scheinberg. Global convergence rate analysis of unconstrained optimization methods based on probabilistic models. *Mathematical Programming*, 169(2):337–375, 2018.

- [21] Antonin Chambolle, Matthias J. Ehrhardt, Peter Richtárik, and Carola-Bibiane Schöenlieb. Stochastic primal-dual hybrid gradient algorithm with arbitrary sampling and imaging applications. *SIAM Journal on Optimization*, 28(4):2783–2808, 2018.
- [22] Antonin Chambolle and Thomas Pock. A first-order primal-dual algorithm for convex problems with applications to imaging. *Journal of Mathematical Imaging and Vision*, 40(1):120–145, 2011.
- [23] Chih-Chung Chang and Chih-Jen Lin. LibSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(3):27, 2011.
- [24] Kai-Wei Chang, Cho-Jui Hsieh, and Chih-Jen Lin. Coordinate descent method for large-scale l2-loss linear support vector machines. *Journal of Machine Learning Research*, 9(Jul):1369–1398, 2008.
- [25] Kamalika Chaudhuri, Sham M Kakade, Praneeth Netrapalli, and Sujay Sanghavi. Convergence rates of active learning for maximum likelihood estimation. In *Advances in Neural Information Processing Systems 28*, pages 1090–1098, 2015.
- [26] Andrew R Conn, Nicholas IM Gould, and Philippe L Toint. *Trust region methods*, volume 1. SIAM, 2000.
- [27] Andrew R Conn, Katya Scheinberg, and Luis N Vicente. *Introduction to derivative-free optimization*, volume 8. SIAM, 2009.
- [28] Luca Corinzia and Joachim M Buhmann. Variational federated multi-task learning. *arXiv preprint arXiv:1906.06268*, 2019.
- [29] Dominik Csiba, Zheng Qu, and Peter Richtárik. Stochastic dual coordinate ascent with adaptive probabilities. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 674–683, Lille, France, 2015.
- [30] Dominik Csiba and Peter Richtárik. Coordinate descent face-off: primal or dual? In *JMLR Workshop and Conference Proceedings, The 29th International Conference on Algorithmic Learning Theory*, 2018.
- [31] Dominik Csiba and Peter Richtárik. Importance sampling for minibatches. *The Journal of Machine Learning Research*, 19(1):962–982, 2018.

- [32] Frank Curtis. A self-correcting variable-metric algorithm for stochastic optimization. In *International Conference on Machine Learning*, pages 632–641, 2016.
- [33] Lisandro D Dalcin, Rodrigo R Paz, Pablo A Kler, and Alejandro Cosimo. Parallel distributed computing using Python. *Advances in Water Resources*, 34(9):1124–1139, 2011.
- [34] Alexandre d’Aspremont. Smooth optimization with approximate gradient. *SIAM Journal on Optimization*, 19(3):1171–1183, 2008.
- [35] Jeffrey Dean, Greg Corrado, Rajat Monga, Kai Chen, Matthieu Devin, Mark Mao, Andrew Senior, Paul Tucker, Ke Yang, Quoc V Le, and et al. Large scale distributed deep networks. In *Advances in Neural Information Processing Systems*, pages 1223–1231, 2012.
- [36] Aaron Defazio. A simple practical accelerated method for finite sums. In *Advances in Neural Information Processing Systems*, pages 676–684, 2016.
- [37] Aaron Defazio, Francis Bach, and Simon Lacoste-Julien. SAGA: A fast incremental gradient method with support for non-strongly convex composite objectives. In *Advances in Neural Information Processing Systems*, pages 1646–1654, 2014.
- [38] Aaron Defazio, Justin Domke, and Caetano. Finito: A faster, permutable incremental gradient method for big data problems. In *International Conference on Machine Learning*, pages 1125–1133, 2014.
- [39] Charles A Desoer and Barry H Whalen. A note on pseudoinverses. *Journal of the Society of Industrial and Applied Mathematics*, 11(2):442–447, 1963.
- [40] Olivier Devolder, François Glineur, and Yurii Nesterov. First-order methods of smooth convex optimization with inexact oracle. *Mathematical Programming*, 146(1-2):37–75, 2014.
- [41] Inderjit S Dhillon, Pradeep K Ravikumar, and Ambuj Tewari. Nearest neighbor based greedy coordinate descent. In *Advances in Neural Information Processing Systems*, pages 2160–2168, 2011.
- [42] Nikita Doikov and Yurii Nesterov. Minimizing uniformly convex functions by cubic regularization of Newton method. *arXiv preprint arXiv:1905.02671*, 2019.

- [43] Nikita Doikov and Peter Richtárik. Randomized block cubic Newton method. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80, pages 1290–1298. PMLR, 10–15 Jul 2018.
- [44] Aritra Dutta, Filip Hanzely, Jingwei Liang, and Peter Richtárik. Best pair formulation & accelerated scheme for non-convex principal component pursuit. *IEEE Transactions on Signal Processing*, 2020.
- [45] Aritra Dutta, Filip Hanzely, and Peter Richtárik. A nonconvex projection method for robust PCA. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 1468–1476, 2019.
- [46] Hubert Eichner, Tomer Koren, Brendan McMahan, Nati Srebro, and Kunal Talwar. Semi-cyclic stochastic gradient descent. In *International Conference on Machine Learning*, pages 1764–1773, 2019.
- [47] Cong Fang, Chris Junchi Li, Zhouchen Lin, and Tong Zhang. Spider: Near-optimal non-convex optimization via stochastic path-integrated differential estimator. In *Advances in Neural Information Processing Systems*, pages 689–699, 2018.
- [48] Olivier Fercoq, Zheng Qu, Peter Richtárik, and Martin Takáč. Fast distributed coordinate descent for minimizing non-strongly convex losses. *IEEE International Workshop on Machine Learning for Signal Processing*, 2014.
- [49] Olivier Fercoq and Peter Richtárik. Accelerated, parallel and proximal coordinate descent. *SIAM Journal on Optimization*, 25(4):1997–2023, 2015.
- [50] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning*, pages 1126–1135, 2017.
- [51] Roger Fletcher. A new approach to variable metric algorithms. *The Computer Journal*, 13(3):317–322, 1970.
- [52] Nidham Gazagnadou, Robert M Gower, and Joseph Salmon. Optimal mini-batch and step sizes for SAGA. In *Proceedings of the 36th International Conference on Machine Learning*, pages 2142–2150, 2019.
- [53] Donald Goldfarb. A family of variable-metric methods derived by variational means. *Mathematics of Computation*, 24(109):23–26, 1970.

- [54] Eduard Gorbunov, Darina Dvinskikh, and Alexander Gasnikov. Optimal decentralized distributed algorithms for stochastic convex optimization. *arXiv preprint arXiv:1911.07363*, 2019.
- [55] Eduard Gorbunov, Filip Hanzely, and Peter Richtárik. A unified theory of SGD: Variance reduction, sampling, quantization and coordinate descent. In *The 23rd International Conference on Artificial Intelligence and Statistics*, 2020.
- [56] Nicholas IM Gould, Daniel P Robinson, and Hilary S Thorne. On solving trust-region and other regularised subproblems in optimization. *Mathematical Programming Computation*, 2(1):21–57, 2010.
- [57] Robert M Gower, Donald Goldfarb, and Peter Richtárik. Stochastic block BFGS: squeezing more curvature out of data. In *Proceedings of The 33rd International Conference on Machine Learning*, pages 1869–1878, 2016.
- [58] Robert M Gower, Filip Hanzely, Peter Richtárik, and Sebastian U Stich. Accelerated stochastic matrix inversion: general theory and speeding up bfgs rules for faster second-order optimization. In *Advances in Neural Information Processing Systems*, pages 1619–1629, 2018.
- [59] Robert M Gower, Dmitry Kovalev, Felix Lieder, and Peter Richtárik. RSN: Randomized subspace Newton. In *Advances in Neural Information Processing Systems* 32, 2019.
- [60] Robert M Gower, Nicolas Loizou, Xun Qian, Alibek Sailanbayev, Egor Shulgin, and Peter Richtárik. SGD: General analysis and improved rates. In *Proceedings of the 36th International Conference on Machine Learning*, pages 5200–5209. PMLR, 09–15 Jun 2019.
- [61] Robert M Gower and Peter Richtárik. Randomized iterative methods for linear systems. *SIAM Journal on Matrix Analysis and Applications*, 36(4):1660–1690, 2015.
- [62] Robert M Gower and Peter Richtárik. Stochastic dual ascent for solving linear systems. *arXiv preprint arXiv:1512.06890*, 2015.
- [63] Robert M Gower and Peter Richtárik. Linearly convergent randomized iterative methods for computing the pseudoinverse. *arXiv preprint arXiv:1612.06255*, 2016.

- [64] Robert M Gower and Peter Richtárik. Randomized quasi-Newton updates are linearly convergent matrix inversion algorithms. *SIAM Journal on Matrix Analysis and Applications*, 38(4):1380–1409, 2017.
- [65] Robert M Gower, Peter Richtárik, and Francis Bach. Stochastic quasi-gradient methods: variance reduction via Jacobian sketching. *Mathematical Programming*, 2020.
- [66] Geovani N Grapiglia and Yurii Nesterov. Regularized Newton methods for minimizing functions with Hölder continuous Hessians. *SIAM Journal on Optimization*, 27(1):478–506, 2017.
- [67] Geovani N Grapiglia and Yurii Nesterov. Accelerated regularized Newton methods for minimizing composite convex functions. *SIAM Journal on Optimization*, 29(1):77–99, 2019.
- [68] Andreas Griewank. The modification of Newton’s method for unconstrained optimization by bounding cubic terms. Technical report, Technical report NA/12, 1981.
- [69] Benjamin Grimmer. Convergence rates for deterministic and stochastic subgradient methods without Lipschitz continuity. *SIAM Journal on Optimization*, 29(2):1350–1365, 2019.
- [70] Dmitry Grishchenko, Franck Iutzeler, Jérôme Malick, and Massih-Reza Amini. Asynchronous distributed learning with sparse communications and identification. *arXiv preprint arXiv:1812.03871*, 2018.
- [71] Suyog Gupta, Ankur Agrawal, Kailash Gopalakrishnan, and Pritish Narayanan. Deep learning with limited numerical precision. In *Proceedings of the 32nd International Conference on Machine Learning - Volume 37, ICML’15*, pages 1737–1746. JMLR.org, 2015.
- [72] Mert Gürbüzbalaban, Asuman Ozdaglar, Pablo A Parrilo, and Nuri Vanli. When cyclic coordinate descent outperforms randomized coordinate descent. In *Advances in Neural Information Processing Systems*, pages 7002–7010, 2017.
- [73] David H Gutman and Javier F Pena. The condition number of a function relative to a set. *arXiv preprint arXiv:1901.08359*, 2019.

- [74] Filip Hanzely, Nikita Doikov, Peter Richtárik, and Yurii Nesterov. Stochastic subspace cubic Newton method. In *International Conference on Machine Learning*, 2020.
- [75] Filip Hanzely, Slavomír Hanzely, Samuel Horváth, and Peter Richtárik. Lower bounds and optimal algorithms for personalized federated learning. Technical Report, 2020.
- [76] Filip Hanzely, Dmitry Kovalev, and Peter Richtárik. Variance reduced coordinate descent with acceleration: New method with a surprising application to finite-sum problems. In *International Conference on Machine Learning*, 2020.
- [77] Filip Hanzely, Konstantin Mishchenko, and Peter Richtárik. SEGA: Variance reduction via gradient sketching. In *Advances in Neural Information Processing Systems*, pages 2083–2094, 2018.
- [78] Filip Hanzely and Peter Richtárik. Accelerated coordinate descent with arbitrary sampling and best rates for minibatches. In *Proceedings of Machine Learning Research*, pages 304–312. PMLR, 16–18 Apr 2019.
- [79] Filip Hanzely and Peter Richtárik. One method to rule them all: Variance reduction for data, parameters and many new methods. *arXiv preprint arXiv:1905.11266*, 2019.
- [80] Filip Hanzely and Peter Richtárik. Federated learning of a mixture of global and local models. *arXiv preprint arXiv:2002.05516*, 2020.
- [81] Filip Hanzely, Peter Richtárik, and Lin Xiao. Accelerated Bregman proximal gradient methods for relatively smooth convex optimization. *arXiv preprint arXiv:1808.03045*, 2018.
- [82] Fred J Hickernell, Christiane Lemieux, Art B Owen, et al. Control variates for quasi-Monte Carlo. *Statistical Science*, 20(1):1–31, 2005.
- [83] Thomas Hofmann, Aurelien Lucchi, Simon Lacoste-Julien, and Brian McWilliams. Variance reduced stochastic gradient descent with neighbors. In *Advances in Neural Information Processing Systems*, pages 2305–2313, 2015.
- [84] Robert Hooke and Terry A Jeeves. “Direct search” solution of numerical and statistical problems. *Journal of the ACM (JACM)*, 8(2):212–229, 1961.

- [85] Samuel Horváth, Dmitry Kovalev, Konstantin Mishchenko, Sebastian U Stich, and Peter Richtárik. Stochastic distributed learning with gradient quantization and variance reduction. *arXiv preprint arXiv:1904.05115*, 2019.
- [86] Samuel Horváth, Lihua Lei, Peter Richtárik, and Michael I Jordan. Adaptivity of stochastic gradient methods for nonconvex optimization. *arXiv preprint arXiv:2002.05359*, 2020.
- [87] Samuel Horváth and Peter Richtarik. Nonconvex variance reduced optimization with arbitrary sampling. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 2781–2789. PMLR, 09–15 Jun 2019.
- [88] Rie Johnson and Tong Zhang. Accelerating stochastic gradient descent using predictive variance reduction. In *Advances in Neural Information Processing Systems 26*, pages 315–323, 2013.
- [89] Stefan Kaczmarz. Angenäherte Auflösung von Systemen linearer Gleichungen. *Bulletin International de l'Académie Polonaise des Sciences et des Lettres*, 35:355–357, 1937.
- [90] Peter Kairouz, Brendan McMahan, and et al. Advances and open problems in federated learning. *arXiv preprint arXiv:1912.04977v1*, 2019.
- [91] Hamed Karimi, Julie Nutini, and Mark Schmidt. Linear convergence of gradient and proximal-gradient methods under the Polyak-Łojasiewicz condition. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 795–811. Springer, 2016.
- [92] Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank J Reddi, Sebastian U Stich, and Ananda Theertha Suresh. SCAFFOLD: stochastic controlled averaging for on-device federated learning. In *International Conference on Machine Learning*, 2020.
- [93] Sai Praneeth Karimireddy, Anastasia Koloskova, Sebastian U Stich, and Martin Jaggi. Efficient greedy coordinate descent for composite problems. In *International Conference on Artificial Intelligence and Statistics*, 2019.
- [94] Sai Praneeth Karimireddy, Sebastian U Stich, and Martin Jaggi. Global linear convergence of Newton’s method without strong-convexity or Lipschitz gradients. *arXiv preprint arXiv:1806.00413*, 2018.

- [95] Ahmed Khaled, Konstantin Mishchenko, and Peter Richtárik. First analysis of local GD on heterogeneous data. In *NeurIPS Workshop on Federated Learning for Data Privacy and Confidentiality*, pages 1–11, 2019.
- [96] Ahmed Khaled, Konstantin Mishchenko, and Peter Richtárik. Tighter theory for local SGD on identical and heterogeneous data. In *The 23rd International Conference on Artificial Intelligence and Statistics (AISTATS 2020)*, 2020.
- [97] Mikhail Khodak, Maria-Florina Balcan, and Ameet S Talwalkar. Adaptive gradient-based meta-learning methods. In *Advances in Neural Information Processing Systems*, pages 5917–5928, 2019.
- [98] Jonas M Köhler and Aurelien Lucchi. Sub-sampled cubic regularization for non-convex optimization. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1895–1904. JMLR. org, 2017.
- [99] Tamara G Kolda, Robert M Lewis, and Virginia Torczon. Optimization by direct search: New perspectives on some classical and modern methods. *SIAM Review*, 45(3):385–482, 2003.
- [100] Jakub Konečný and Peter Richtárik. Simple complexity analysis of simplified direct search. *arXiv preprint arXiv:1410.0390*, 2014.
- [101] Jakub Konečný and Peter Richtárik. Semi-stochastic gradient descent methods. *Frontiers in Applied Mathematics and Statistics*, 3:9, 2017.
- [102] Jakub Konečný, Jie Lu, Peter Richtárik, and Martin Takáč. Mini-batch semi-stochastic gradient descent in the proximal setting. *IEEE Journal of Selected Topics in Signal Processing*, 10(2):242–255, 2016.
- [103] Jakub Konečný, Brendan McMahan, Daniel Ramage, and Peter Richtárik. Federated optimization: distributed machine learning for on-device intelligence. *arXiv preprint arXiv:1610.02527*, 2016.
- [104] Jakub Konečný, Brendan McMahan, Felix Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. Federated learning: strategies for improving communication efficiency. In *NIPS Private Multi-Party Machine Learning Workshop*, 2016.
- [105] Jakub Konečný and Peter Richtárik. Randomized distributed mean estimation: accuracy vs. communication. *Frontiers in Applied Mathematics and Statistics*, 4(62):1–11, 2018.

- [106] Dmitry Kovalev, Samuel Horváth, and Peter Richtárik. Dont jump through hoops and remove those loops: SVRG and Katyusha are better without the outer loop. In *Proceedings of the 31st International Conference on Algorithmic Learning Theory*, 2020.
- [107] Dmitry Kovalev, Konstantin Mishchenko, and Peter Richtárik. Stochastic Newton and cubic Newton methods with simple local linear-quadratic rates. *NeurIPS 2019 Workshop Beyond First Order Methods in ML*, 2019.
- [108] Dmitry Kovalev, Peter Richtárik, Eduard Gorbunov, and Elnur Gasanov. Stochastic spectral and conjugate descent methods. In *Advances in Neural Information Processing Systems*, pages 3358–3367, 2018.
- [109] David Kozak, Stephen Becker, Alireza Doostan, and Luis Tenorio. Stochastic subspace descent. *arXiv preprint arXiv:1904.01145*, 2019.
- [110] Andrei Kulunchakov and Julien Mairal. Estimate sequences for variance-reduced stochastic composite optimization. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 3541–3550. PMLR, 09–15 Jun 2019.
- [111] Andrei Kulunchakov and Julien Mairal. A generic acceleration framework for stochastic composite optimization. In *Advances in Neural Information Processing Systems*, pages 12556–12567, 2019.
- [112] Guanghui Lan, Soomin Lee, and Yi Zhou. Communication-efficient algorithms for decentralized and stochastic optimization. *Mathematical Programming*, pages 1–48, 2018.
- [113] Guanghui Lan and Yi Zhou. An optimal randomized incremental gradient method. *Mathematical Programming*, 171(1-2):167–215, 2018.
- [114] Rémi Leblond, Fabian Pedregosa, and Simon Lacoste-Julien. Asaga: Asynchronous parallel saga. In *Artificial Intelligence and Statistics*, pages 46–54, 2017.
- [115] Rémi Leblond, Fabian Pedregosa, and Simon Lacoste-Julien. Improved asynchronous parallel optimization analysis for stochastic incremental methods. *The Journal of Machine Learning Research*, 19(1):3140–3207, 2018.
- [116] Yin Tat Lee and Aaron Sidford. Efficient accelerated coordinate descent methods and faster algorithms for solving linear systems. *Proceedings - Annual IEEE Symposium on Foundations of Computer Science, FOCS*, pages 147–156, 2013.

- [117] Lihua Lei and Michael I Jordan. Less than a single pass: Stochastically controlled stochastic gradient. In *Artificial Intelligence and Statistics*, pages 148–156, 2017.
- [118] Mu Li, Tong Zhang, Yuqiang Chen, and Alexander J Smola. Efficient mini-batch training for stochastic optimization. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 661–670, 2014.
- [119] Tian Li, Anit Kumar Sahu, Ameet S Talwalkar, and Virginia Smith. Federated learning: Challenges, methods, and future directions. *IEEE Signal Processing Magazine*, 37(3):50–60, 2020.
- [120] Xianfeng Liang, Shuheng Shen, Jingchang Liu, Zhen Pan, Enhong Chen, and Yifei Cheng. Variance reduced local SGD with lower communication complexity. *arXiv preprint arXiv:1912.12844*, 2019.
- [121] Hongzhou Lin, Julien Mairal, and Zaid Harchaoui. A universal catalyst for first-order optimization. In *Advances in Neural Information Processing Systems*, pages 3384–3392, 2015.
- [122] Qihang Lin, Zhaosong Lu, and Lin Xiao. An accelerated proximal coordinate gradient method. In *Advances in Neural Information Processing Systems*, pages 3059–3067, 2014.
- [123] Yujun Lin, Song Han, Huizi Mao, Yu Wang, and William J Dally. Deep gradient compression: Reducing the communication bandwidth for distributed training. *arXiv preprint arXiv:1712.01887*, 2017.
- [124] Dong C Liu and Jorge Nocedal. On the limited memory BFGS method for large scale optimization. *Mathematical Programming*, 45(1-3):503–528, 1989.
- [125] Ji Liu and Stephen J Wright. An accelerated randomized Kaczmarz algorithm. *Mathematics of Computation*, 85(297):153–178, 2016.
- [126] Nicos Loizou and Peter Richtárik. Accelerated gossip via stochastic heavy ball method. In *56th Annual Allerton Conference on Communication, Control, and Computing*, 2018.
- [127] Nicolas Loizou and Peter Richtárik. A new perspective on randomized gossip algorithms. In *IEEE Global Conference on Signal and Information Processing (Global-SIP)*, pages 440–444, 2016.

- [128] Nicolas Loizou and Peter Richtárik. Linearly convergent stochastic heavy ball method for minimizing generalization error. In *NIPS Workshop on Optimization for Machine Learning*, 2017.
- [129] Nicolas Loizou and Peter Richtárik. Momentum and stochastic momentum for stochastic gradient, Newton, proximal point and subspace descent methods. *arXiv preprint arXiv:1712.09677*, 2017.
- [130] Zhi-Quan Luo and Paul Tseng. On the convergence of the coordinate descent method for convex differentiable minimization. *Journal of Optimization Theory and Applications*, 72(1):7–35, 1992.
- [131] Chenxin Ma, Virginia Smith, Martin Jaggi, Michael I Jordan, Peter Richtárik, and Martin Takáč. Adding vs. averaging in distributed primal-dual optimization. In *The 32nd International Conference on Machine Learning*, pages 1973–1982, 2015.
- [132] Julien Mairal. Optimization with first-order surrogate functions. In *International Conference on Machine Learning*, pages 783–791, 2013.
- [133] Julien Mairal. Incremental majorization-minimization optimization with application to large-scale machine learning. *SIAM Journal on Optimization*, 25(2):829–855, 2015.
- [134] Brendan McMahan, Eider Moore, Daniel Ramage, and Blaise Agüera y Arcas. Federated learning of deep networks using model averaging. *arXiv preprint arXiv:1602.05629*, 2016.
- [135] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2017.
- [136] Konstantin Mishchenko, Eduard Gorbunov, Martin Takáč, and Peter Richtárik. Distributed learning with compressed gradient differences. *arXiv preprint arXiv:1901.09269*, 2019.
- [137] Konstantin Mishchenko, Filip Hanzely, and Peter Richtárik. 99% of worker-master communication in distributed optimization is not needed. In *36th Conference on Uncertainty in Artificial Intelligence, (UAI 2020)*. AUAI, 2020.

- [138] Konstantin Mishchenko, Franck Iutzeler, and Jérôme Malick. A distributed flexible delay-tolerant proximal gradient algorithm. *SIAM Journal on Optimization*, 30(1):933–959, 2020.
- [139] Konstantin Mishchenko and Peter Richtárik. A stochastic decoupling method for minimizing the sum of smooth and non-smooth functions. *arXiv preprint arXiv:1905.11535*, 2019.
- [140] Aryan Mokhtari and Alejandro Ribeiro. Global convergence of online limited memory BFGS. *The Journal of Machine Learning Research*, 16(1):3151–3181, 2015.
- [141] Renato DC Monteiro and Benar Fux Svaiter. An accelerated hybrid proximal extragradient method for convex optimization and its implications to second-order methods. *SIAM Journal on Optimization*, 23(2):1092–1125, 2013.
- [142] Philipp Moritz, Robert Nishihara, and Michael I Jordan. A linearly-convergent stochastic L-BFGS algorithm. In *Artificial Intelligence and Statistics*, pages 249–258, 2016.
- [143] Éric Moulines and Francis Bach. Non-asymptotic analysis of stochastic approximation algorithms for machine learning. In *Advances in Neural Information Processing Systems*, pages 451–459, 2011.
- [144] Mojmír Mútný, Michał Dereziński, and Andreas Krause. Convergence analysis of block coordinate algorithms with determinantal sampling. In *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, volume 108 of *Proceedings of Machine Learning Research*, pages 3110–3120. PMLR, 2020.
- [145] Ion Necoara, Peter Richtárik, and Andrei Patrascu. Randomized projection methods for convex feasibility: Conditioning and convergence rates. *SIAM Journal on Optimization*, 29(4):2814–2852, 2019.
- [146] Deanna Needell, Nathan Srebro, and Rachel Ward. Stochastic gradient descent, weighted sampling, and the randomized Kaczmarz algorithm. *Mathematical Programming*, 155(1–2):549–573, 2015.
- [147] Deanna Needell and Rachel Ward. Batched stochastic gradient descent with weighted sampling. In *International Conference Approximation Theory*, pages 279–306. Springer, 2016.

- [148] Arkadi Nemirovski, Anatoli Juditsky, Guanghui Lan, and Alexander Shapiro. Robust stochastic approximation approach to stochastic programming. *SIAM Journal on Optimization*, 19(4):1574–1609, 2009.
- [149] Y. Nesterov. A method for solving a convex programming problem with convergence rate $O(1/k^2)$. *Soviet Mathematics - Doklady*, 27(2):372–376, 1983.
- [150] Yurii Nesterov. Smooth minimization of nonsmooth functions. *Mathematical Programming*, 103:127–152, 2005.
- [151] Yurii Nesterov. Accelerating the cubic regularization of Newtons method on convex problems. *Mathematical Programming*, 112(1):159–181, 2008.
- [152] Yurii Nesterov. Efficiency of coordinate descent methods on huge-scale optimization problems. *SIAM Journal on Optimization*, 22(2):341–362, 2012.
- [153] Yurii Nesterov. Random gradient-free minimization of convex functions. *Foundations of Computational Mathematics*, 17(2):527–566, 2017.
- [154] Yurii Nesterov. *Lectures on convex optimization*, volume 137. Springer, 2018.
- [155] Yurii Nesterov. Inexact basic tensor methods. *CORE Discussion Papers 2019/23*, 2019.
- [156] Yurii Nesterov and Boris T Polyak. Cubic regularization of Newton method and its global performance. *Mathematical Programming*, 108(1):177–205, 2006.
- [157] Yurii Nesterov and Sebastian U Stich. Efficiency of the accelerated coordinate descent method on structured optimization problems. *SIAM Journal on Optimization*, 27(1):110–123, 2017.
- [158] Yurii Nesterov and Sebastian U Stich. Efficiency of the accelerated coordinate descent method on structured optimization problems. *SIAM Journal on Optimization*, 27(1):110–123, 2017.
- [159] Lam Nguyen, Phuong Ha, Marten van Dijk, Peter Richtárik, Katya Scheinberg, and Martin Takáč. SGD and Hogwild! Convergence without the bounded gradients assumption. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 3750–3758. PMLR, 10–15 Jul 2018.

- [160] Lam Nguyen, Jie Liu, Katya Scheinberg, and Martin Takáč. SARAH: A novel method for machine learning problems using stochastic recursive gradient. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 2613–2621. PMLR, 2017.
- [161] Julie Nutini, Mark Schmidt, Issam Laradji, Michael Friedlander, and Hoyt Koepke. Coordinate descent converges faster with the Gauss-Southwell rule than random selection. In *International Conference on Machine Learning*, pages 1632–1641, 2015.
- [162] Gert K Pedersen. *Analysis Now*. Graduate Texts in Mathematics. Springer New York, 1996.
- [163] Mert Pilanci and Martin J Wainwright. Newton sketch: A near linear-time optimization algorithm with linear-quadratic convergence. *SIAM Journal on Optimization*, 27(1):205–245, 2017.
- [164] Xun Qian, Zheng Qu, and Peter Richtárik. L-SVRG and L-Katyusha with arbitrary sampling. *arXiv preprint arXiv:1906.01481*, 2019.
- [165] Xun Qian, Zheng Qu, and Peter Richtárik. SAGA with arbitrary sampling. In *Proceedings of the 36th International Conference on Machine Learning*, pages 5190–5199, 2019.
- [166] Zheng Qu and Peter Richtárik. Coordinate descent with arbitrary sampling I: Algorithms and complexity. *Optimization Methods and Software*, 31(5):829–857, 2016.
- [167] Zheng Qu and Peter Richtárik. Coordinate descent with arbitrary sampling II: Expected separable overapproximation. *Optimization Methods and Software*, 31(5):858–884, 2016.
- [168] Zheng Qu, Peter Richtárik, Martin Takáč, and Olivier Fercoq. SDNA: stochastic dual Newton ascent for empirical risk minimization. In *Proceedings of The 33rd International Conference on Machine Learning*, pages 1823–1832, 2016.
- [169] Zheng Qu, Peter Richtárik, and Tong Zhang. Quartz: Randomized dual coordinate ascent with arbitrary sampling. In *Advances in Neural Information Processing Systems 28*, pages 865–873, 2015.
- [170] Benjamin Recht, Christopher Re, Stephen J Wright, and Feng Niu. Hogwild: A lock-free approach to parallelizing stochastic gradient descent. In *Advances in Neural Information Processing Systems*, pages 693–701, 2011.

- [171] Sashank J Reddi, Ahmed Hefny, Suvrit Sra, Barnabás Póczos, and Alex Smola. Stochastic variance reduction for nonconvex optimization. In *International Conference on Machine Learning*, pages 314–323, 2016.
- [172] Sashank J Reddi, Jakub Konečný, Peter Richtárik, Barnabás Póczos, and Alex Smola. AIDE: fast and communication efficient distributed optimization. *arXiv preprint arXiv:1608.06879*, 2016.
- [173] Peter Richtárik and Martin Takáč. Iteration complexity of randomized block-coordinate descent methods for minimizing a composite function. *Mathematical Programming*, 144(1-2):1–38, 2014.
- [174] Peter Richtárik and Martin Takáč. Distributed coordinate descent method for learning with big data. *Journal of Machine Learning Research*, 17(75):1–25, 2016.
- [175] Peter Richtárik and Martin Takáč. On optimal probabilities in stochastic coordinate descent methods. *Optimization Letters*, 10(6):1233–1243, 2016.
- [176] Peter Richtárik and Martin Takáč. Stochastic reformulations of linear systems: algorithms and convergence theory. *SIAM Journal on Matrix Analysis and Applications*, 2020.
- [177] Peter Richtárik and Martin Takáč. Parallel coordinate descent methods for big data optimization. *Mathematical Programming*, 156(1-2):433–484, 2016.
- [178] Peter Richtárik and Martin Takáč. Stochastic reformulations of linear systems: accelerated method. *Manuscript, October 2017*, 2017.
- [179] Herbert Robbins and Sutton Monro. A stochastic approximation method. *The Annals of Mathematical Statistics*, page 400407, 1951.
- [180] Anton Rodomanov and Dmitry Kropotov. A randomized coordinate descent method with volume sampling. *SIAM Journal on Optimization*, 30(3):1878–1904, 2020.
- [181] Farbod Roosta-Khorasani and Michael W Mahoney. Sub-sampled Newton methods. *Mathematical Programming*, 174(1-2):293–326, 2019.
- [182] Nicolas Le Roux, Mark Schmidt, and Francis Bach. A stochastic gradient method with an exponential convergence rate for finite training sets. In *Advances in Neural Information Processing Systems*, pages 2663–2671, 2012.

- [183] Ankan Saha and Ambuj Tewari. On the nonasymptotic convergence of cyclic coordinate descent methods. *SIAM Journal on Optimization*, 23(1):576–601, 2013.
- [184] Amedeo Sapia, Marco Canini, Chen-Yu Ho, Jacob Nelson, Panos Kalnis, Changhoon Kim, Arvind Krishnamurthy, Masoud Moshref, Dan RK Ports, and Peter Richtárik. Scaling distributed machine learning with in-network aggregation. *arXiv preprint arXiv:1903.06701*, 2019.
- [185] Mark Schmidt, Nicolas Le Roux, and Francis Bach. Convergence rates of inexact proximal-gradient methods for convex optimization. In *Advances in Neural Information Processing Systems*, pages 1458–1466, 2011.
- [186] Nicol N Schraudolph, Jin Yu, and Simon Günter. A stochastic quasi-Newton method for online convex optimization. In *Artificial Intelligence and Statistics*, pages 436–443, 2007.
- [187] Frank Seide, Hao Fu, Jasha Droppo, Gang Li, and Dong Yu. 1-bit stochastic gradient descent and its application to data-parallel distributed training of speech dnns. In *Fifteenth Annual Conference of the International Speech Communication Association*, 2014.
- [188] Shai Shalev-Shwartz and Shai Ben-David. *Understanding machine learning: from theory to algorithms*. Cambridge University Press, 2014.
- [189] Shai Shalev-Shwartz and Ambuj Tewari. Stochastic methods for l_1 -regularized loss minimization. *Journal of Machine Learning Research*, 12(Jun):1865–1892, 2011.
- [190] Shai Shalev-Shwartz and Tong Zhang. Proximal stochastic dual coordinate ascent. *arXiv preprint arXiv:1211.2717*, 2012.
- [191] Shai Shalev-Shwartz and Tong Zhang. Stochastic dual coordinate ascent methods for regularized loss. *Journal of Machine Learning Research*, 14(1):567–599, 2013.
- [192] Shai Shalev-Shwartz and Tong Zhang. Accelerated proximal stochastic dual coordinate ascent for regularized loss minimization. In *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pages 64–72, Beijing, China, 2014.
- [193] Ohad Shamir, Nati Srebro, and Tong Zhang. Communication-efficient distributed optimization using an approximate Newton-type method. In *Proceedings of the 31st*

- International Conference on Machine Learning*, PMLR, volume 32, pages 1000–1008, 2014.
- [194] Fanhua Shang, Licheng Jiao, Kaiwen Zhou, James Cheng, Yan Ren, and Yufei Jin. ASVRG: Accelerated proximal SVRG. In *Proceedings of The 10th Asian Conference on Machine Learning*, 2018.
 - [195] David F Shanno. Conditioning of quasi-Newton methods for function minimization. *Mathematics of Computation*, 24(111):647–656, 1970.
 - [196] Virginia Smith, Chao-Kai Chiang, Maziar Sanjabi, and Ameet Talwalkar. Federated multi-task learning. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 4424–4434. Curran Associates, Inc., 2017.
 - [197] Sebastian U Stich. *Convex optimization with random pursuit*. PhD thesis, ETH Zurich, 2014.
 - [198] Sebastian U Stich. Local SGD converges fast and communicates little. In *International Conference on Learning Representations*, 2020.
 - [199] Sebastian U Stich, Christian L Müller, and Bernd Gärtner. Optimization of convex functions with random pursuit. *SIAM Journal on Optimization*, 23(2):1284–1309, 2013.
 - [200] Sebastian U Stich, Christian L Müller, and Bernd Gärtner. Variable metric random pursuit. *Mathematical Programming*, 156(1):549–579, Mar 2016.
 - [201] Sebastian U Stich, Anant Raj, and Martin Jaggi. Approximate steepest coordinate descent. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 3251–3259, International Convention Centre, Sydney, Australia, 2017.
 - [202] Sebastian U Stich, Anant Raj, and Martin Jaggi. Safe adaptive importance sampling. In *Advances in Neural Information Processing Systems*, pages 4384–4394, 2017.
 - [203] Thomas Strohmer and Roman Vershynin. A randomized Kaczmarz algorithm with exponential convergence. *Journal of Fourier Analysis and Applications*, 15(2):262, 2009.

- [204] Quoc Tran-Dinh, Nhan H Pham, Dzung T Phan, and Lam Nguyen. Hybrid stochastic gradient descent algorithms for stochastic nonconvex optimization. *arXiv preprint arXiv:1905.05920*, 2019.
- [205] Nilesh Tripuraneni, Mitchell Stern, Chi Jin, Jeffrey Regier, and Michael I Jordan. Stochastic cubic regularization for fast nonconvex optimization. In *Advances in Neural Information Processing Systems*, pages 2899–2908, 2018.
- [206] Paul Tseng. Convergence of a block coordinate descent method for nondifferentiable minimization. *Journal of Optimization Theory and Applications*, 109(3):475–494, 2001.
- [207] Stephen Tu, Shivaram Venkataraman, Ashia C Wilson, Alex Gittens, Michael I Jordan, and Benjamin Recht. Breaking locality accelerates block gauss-seidel. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 3482–3491. JMLR. org, 2017.
- [208] Sharan Vaswani, Francis Bach, and Mark Schmidt. Fast and faster convergence of SGD for over-parameterized models and an accelerated perceptron. In *22nd International Conference on Artificial Intelligence and Statistics*, volume 89 of *PMLR*, pages 1195–1204, 2019.
- [209] Hongyi Wang, Scott Sievert, Shengchao Liu, Zachary Charles, Dimitris Papailiopoulos, and Stephen J Wright. Atomo: Communication-efficient learning via atomic sparsification. In *Advances in Neural Information Processing Systems 31*, pages 9850–9861. Curran Associates, Inc., 2018.
- [210] Xiao Wang, Shiqian Ma, Donald Goldfarb, and Wei Liu. Stochastic quasi-Newton methods for nonconvex stochastic optimization. *SIAM Journal on Optimization*, 27(2):927–956, 2017.
- [211] Zhe Wang, Yi Zhou, Yingbin Liang, and Guanghui Lan. Stochastic variance-reduced cubic regularization for nonconvex optimization. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 2731–2740, 2019.
- [212] Jianqiao Wangni, Jialei Wang, Ji Liu, and Tong Zhang. Gradient sparsification for communication-efficient distributed optimization. In *Advances in Neural Information Processing Systems*, pages 1306–1316, 2018.
- [213] Wei Wen, Cong Xu, Feng Yan, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. Terngrad: Ternary gradients to reduce communication in distributed deep

- learning. In *Advances in Neural Information Processing Systems*, pages 1509–1519, 2017.
- [214] Blake E Woodworth and Nati Srebro. Tight complexity bounds for optimizing composite objectives. In *Advances in Neural Information Processing Systems*, pages 3639–3647, 2016.
- [215] Stephen J Wright. Coordinate descent algorithms. *Mathematical Programming*, 151(1):3–34, 2015.
- [216] Zhaoxian Wu, Qing Ling, Tianyi Chen, and Georgios B Giannakis. Federated variance-reduced stochastic gradient descent with robustness to byzantine attacks. *IEEE Transactions on Signal Processing*, 2020.
- [217] Lin Xiao and Tong Zhang. A proximal stochastic gradient method with progressive variance reduction. *SIAM Journal on Optimization*, 24(4):2057–2075, 2014.
- [218] Peng Xu, Farbod Roosta-Khorasani, and Michael W Mahoney. Newton-type methods for non-convex optimization under inexact hessian information. *Mathematical Programming*, pages 1–36, 2017.
- [219] Peng Xu, Jiyan Yang, Farbod Roosta-Khorasani, Christopher Ré, and Michael W Mahoney. Sub-sampled Newton methods with non-uniform sampling. In *Advances in Neural Information Processing Systems*, pages 3000–3008, 2016.
- [220] Yang You, Xiangru Lian, Ji Liu, Hsiang-Fu Yu, Inderjit S Dhillon, James Demmel, and Cho-Jui Hsieh. Asynchronous parallel greedy coordinate descent. In *Advances in Neural Information Processing Systems*, pages 4682–4690, 2016.
- [221] Fuzhen Zhang. *Matrix Theory: Basic Results and Techniques*. Springer-Verlag New York, 1999.
- [222] Hantian Zhang, Jerry Li, Kaan Kara, Dan Alistarh, Ji Liu, and Ce Zhang. ZipML: Training linear models with end-to-end low precision, and a little bit of deep learning. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 4035–4043. PMLR, 2017.
- [223] Peilin Zhao and Tong Zhang. Stochastic optimization with importance sampling for regularized loss minimization. In *Proceedings of the 32nd International Conference on Machine Learning*, PMLR, volume 37, pages 1–9, 2015.

- [224] Tuo Zhao, Mo Yu, Yiming Wang, Raman Arora, and Han Liu. Accelerated mini-batch randomized block coordinate descent method. In *Advances in Neural Information Processing Systems*, pages 3329–3337, 2014.
- [225] Yue Zhao, Meng Li, Liangzhen Lai, Naveen Suda, Damon Civin, and Vikas Chandra. Federated learning with non-iid data. *arXiv preprint arXiv:1806.00582*, 2018.
- [226] Kaiwen Zhou, Qinghua Ding, Fanhua Shang, James Cheng, Danli Li, and Zhi-Quan Luo. Direct acceleration of SAGA using sampled negative momentum. In Kamalika Chaudhuri and Masashi Sugiyama, editors, *Proceedings of Machine Learning Research*, volume 89 of *Proceedings of Machine Learning Research*, pages 1602–1610. PMLR, 16–18 Apr 2019.
- [227] Kaiwen Zhou, Fanhua Shang, and James Cheng. A simple stochastic variance reduced algorithm with fast convergence rates. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 5980–5989. PMLR, 10–15 Jul 2018.

APPENDICES

Appendix A

Table of Frequently Used Notation

For all chapters	
Basic	
$\mathbb{E}[\cdot], \mathbb{P}(\cdot)$ $\langle \cdot, \cdot \rangle, \ \cdot\ $ $\langle \cdot, \cdot \rangle_{\mathbf{B}}, \ \cdot\ _{\mathbf{B}}$ $\lambda_{\max}(\cdot), \lambda_{\min}(\cdot)$ $\nabla h(x)$ $\nabla_i h(x)$ $\nabla^2 h(x)$	Expectation / Probability Standard inner product and norm in \mathbb{R}^d : $\langle x, y \rangle = \sum_{i=1}^d x_i y_i$; $\ x\ = \sqrt{\langle x, x \rangle}$ Weighted inner product and norm in \mathbb{R}^d : $\langle x, y \rangle = x^\top \mathbf{B} y$; $\ x\ = \sqrt{\langle x, x \rangle_{\mathbf{B}}}$ Maximal eigenvalue / minimal eigenvalue Gradient of a differentiable function h i th partial derivative of a differentiable function h Hessian of a twice differentiable function h
Objective	
d $F: \mathbb{R}^d \rightarrow \mathbb{R}$ $f: \mathbb{R}^d \rightarrow \mathbb{R}$ $f_j: \mathbb{R}^d \rightarrow \mathbb{R}$ $\psi: \mathbb{R}^d \rightarrow \mathbb{R} \cup \{+\infty\}$ x^* F^* L, \mathbf{M} μ	Dimensionality of space $x \in \mathbb{R}^d$ Objective function Smooth part of the objective, often of finite sum structure ($= \frac{1}{n} \sum_{i=1}^n f_i(x)$) Differentiable convex function ($1 \leq j \leq n$) Non-smooth part of the objective Global optimum of (1.1) $\stackrel{\text{def}}{=} F(x^*)$, the optimum value of the objective Smoothness constant/smoothness matrix of f Strong convexity of f
Linear operators $\mathbb{R}^{d \times n} \rightarrow \mathbb{R}^{d \times n}$	
\mathcal{A} \mathcal{A}^* \mathcal{A}^\dagger $\text{Range}(\mathcal{A})$ $\text{Range}(\mathcal{A})^\perp$ $\text{Null}(\mathcal{A})$ \mathcal{I}	A generic linear operator Adjoint of \mathcal{A} : $\langle \mathcal{A}\mathbf{X}, \mathbf{Y} \rangle \equiv \langle \mathbf{X}, \mathcal{A}^* \mathbf{Y} \rangle$ for all $\mathbf{X}, \mathbf{Y} \in \mathbb{R}^{d \times n}$ Moore Penrose pseudoinverse of \mathcal{A} Image (range space) of \mathcal{A} : $\text{Range}(\mathcal{A}) \stackrel{\text{def}}{=} \{\mathcal{A}\mathbf{X} : \mathbf{X} \in \mathbb{R}^{d \times n}\}$ Orthogonal complement of $\text{Range}(\mathcal{A})$ Kernel (null space) of \mathcal{A} : $\text{Null}(\mathcal{A}) \stackrel{\text{def}}{=} \{\mathbf{X} \in \mathbb{R}^{d \times n} : \mathcal{A}\mathbf{X} = 0\}$ Identity operator: $\mathcal{I}\mathbf{X} \equiv \mathbf{X}$
Other	
e_i \mathbf{I} $\text{prox}_{\alpha\psi}(x)$ $\langle \mathbf{X}, \mathbf{Y} \rangle$ $\ \mathbf{X}\ $ $\mathbf{X} \circ \mathbf{Y}$ $\mathbf{X} \otimes \mathbf{Y}$ $\text{Tr}(\cdot)$ $\text{Diag}(x)$ $[n]$ $D_h(x, y)$	i th vector from the standard basis Identity matrix Proximal operator of ψ : $\text{prox}_{\alpha\psi}(x) \stackrel{\text{def}}{=} \arg \min_{u \in \mathbb{R}^d} \{\alpha\psi(u) + \frac{1}{2}\ u - x\ ^2\}$ Trace inner product of matrices \mathbf{X} and \mathbf{Y} : $\langle \mathbf{X}, \mathbf{Y} \rangle \stackrel{\text{def}}{=} \text{Tr}(\mathbf{X}^\top \mathbf{Y})$ Frobenius norm of matrix \mathbf{X} : $\ \mathbf{X}\ = \langle \mathbf{X}, \mathbf{X} \rangle^{\frac{1}{2}}$ Hadamard product: $(\mathbf{X} \circ \mathbf{Y})_{ij} = \mathbf{X}_{ij} \mathbf{Y}_{ij}$ Kronecker product Trace Diagonal matrix with vector x on the diagonal Set $\{1, 2, \dots, n\}$ Bregman distance $D_h(x, y) \stackrel{\text{def}}{=} h(x) - h(y) - \langle \nabla h(y), x - y \rangle$

Table A.1: Summary of frequently used notation.

Chapter 3	
Basic	
\mathcal{D}	Distribution over sketch matrices \mathbf{S}
\mathbf{S}	Sketch matrix from (3.3)
b	Random variable such that $\mathbf{S} \in \mathbb{R}^{n \times b}$
$\zeta(\mathbf{S}, x)$	Sketched gradient at x from (3.2)
\mathbf{Z}	$\mathbf{S}(\mathbf{S}^\top \mathbf{S})^\dagger \mathbf{S}^\top$
θ	Random variable for which $\mathbb{E}[\theta \mathbf{Z}] = \mathbf{I}$ from (3.5)
\mathbf{C}	$\mathbb{E}[\theta^2 \mathbf{Z}]$ from Theorem 3.4.2
h, g	Biased and unbiased gradient estimators from (3.4), (3.6)
Φ	Lyapunov function from Theorem 3.4.2,
σ	Parameter for Lyapunov function from Theorems 3.4.2, 3.5.2
Extra Notation for Section 3.5	
p, \mathbf{P}	Probability vector and matrix
v	vector of ESO parameters from (3.13)
$\hat{\mathbf{P}}, \hat{\mathbf{V}}$	$\text{Diag}(p), \text{Diag}(v)$
γ	$\alpha - \alpha^2 \max_i \{\frac{v_i}{p_i}\} - \sigma$ from Theorem 3.5.2
y, z	Extra sequences of iterates for ASEGA
τ, β	Parameters for ASEGA
Ψ, Υ	Lyapunov functions from Theorems 3.5.2, C.2.5
$\eta(v, p)$	$\max_i \frac{\sqrt{v_i}}{p_i}$

Table A.2: Summary of frequently used notation specific to Chapter 3.

Chapter 4	
General	
n	Number of parallel workers/machines
τ	Ratio of coordinate blocks to be sampled by each machine
m	Number of coordinate blocks
f_i	Part of the objective owned by machine i from (4.1)
L	Each f_i is L smooth (Assumption 4.4.1)
U_i^t	Subset of blocks sampled at iteration t and worker i
g	Unbiased gradient estimator
ISAGA	
\mathbf{J}_j	Delayed estimate of j th gradient from (4.6), (4.5)
N	Finite sum size for shared data problem from (4.4)
l	Number of datapoints per machine in distributed setup from (4.3)
\mathcal{L}^t	Lyapunov function from (D.7)
ISGD	
g_i^t	Unbiased stochastic gradient; $\mathbb{E}[g_i^t] = \nabla f_i(x^t)$
σ^2	An upper bound on the variance of stochastic gradients from Assumption 4.6.1
ISEGA	
h_i^t	Sequence of biased estimators for $\nabla f_i(x^t)$ from (4.12)
g_i^t	Sequence of unbiased estimators for $\nabla f_i(x^t)$ from (4.13)
Φ^t	Lyapunov function from Theorem 4.8.1

Table A.3: Summary of frequently used notation specific to Chapter 4.

Chapter 5	
Sets	
R	random subset ("sampling") of $[n]$
R^k	random subset ("sampling") of $[n]$ drawn at iteration k
L	a random subset ("sampling") of $[d]$
L^k	random subset ("sampling") of $[d]$ drawn at iteration k
p_j	probability that $j \in R$
p_i	probability that $i \in L$
Spaces \mathbb{R}^n and \mathbb{R}^d	
$\mathbf{e} \in \mathbb{R}^n$	vector of all ones in \mathbb{R}^n
$\mathbf{e} \in \mathbb{R}^d$	vector of all ones in \mathbb{R}^d
$\mathbf{e}_j \in \mathbb{R}^n$	j th standard unit basis vector in \mathbb{R}^n
$\mathbf{e}_i \in \mathbb{R}^n$	i th standard unit basis vector in \mathbb{R}^d
$\mathbf{x}^k \in \mathbb{R}^d$	the k th iterate produced by GJS
$\mathbf{p} \in \mathbb{R}^d$	the vector (p_1, \dots, p_d)
$\mathbf{p}^t \in \mathbb{R}^d$	the vector (p_1^t, \dots, p_d^t)
$\mathbf{p} \in \mathbb{R}^n$	the vector (p_1, \dots, p_n)
$\mathbf{p}^t \in \mathbb{R}^n$	the vector (p_1^t, \dots, p_n^t)
$\mathbf{q} \in \mathbb{R}^n$	the vector (q_1, \dots, q_n)
$\mathbf{q}^t \in \mathbb{R}^n$	the vector (q_1^t, \dots, q_n^t)
$\mathbf{v} \in \mathbb{R}^n$	any vector for which (E.18) holds
\mathbf{x}^{-1}	elementwise inverse of \mathbf{x}
\mathbf{g}^k	estimator of the gradient $\nabla f(\mathbf{x}^k)$ produced by GJS
Matrices in $\mathbb{R}^{d \times d}$, $\mathbb{R}^{d \times n}$ and $\mathbb{R}^{n \times n}$	
$\mathbf{I}_d \in \mathbb{R}^{d \times d}$	$d \times d$ identity matrix
$\mathbf{I}_n \in \mathbb{R}^{n \times n}$	$n \times n$ identity matrix
$\mathbf{G}(\mathbf{x}) \in \mathbb{R}^{d \times n}$	the Jacobian matrix, i.e., $\mathbf{G}(\mathbf{x}) = [\nabla f_1(\mathbf{x}), \dots, \nabla f_n(\mathbf{x})]$
$\mathbf{J}^k \in \mathbb{R}^{d \times n}$	estimator of the Jacobian produced by GJS
$\mathbf{M}_j \in \mathbb{R}^{d \times d}$	smoothness matrix of f_j (if $\mathbf{M}_j = m^j \mathbf{I}_d$, then this specializes to m^j -smoothness)
$\mathbf{R} \in \mathbb{R}^{n \times n}$	a random matrix we use to multiply \mathbf{J} or \mathbf{G} from the right
$\mathbf{R}_R \in \mathbb{R}^{n \times n}$	the random matrix $\mathbf{R}_R \stackrel{\text{def}}{=} \sum_{j \in R} \mathbf{e}_j \mathbf{e}_j^\top$
$\mathbf{L} \in \mathbb{R}^{d \times d}$	a random matrix we use to multiply \mathbf{J} or \mathbf{G} from the left
$\mathbf{L}_L \in \mathbb{R}^{d \times d}$	the random matrix $\mathbf{L}_L \stackrel{\text{def}}{=} \sum_{i \in L} \mathbf{e}_i \mathbf{e}_i^\top$
$\mathbf{P} \in \mathbb{R}^{n \times n}$	Matrix defined by $\mathbf{P}_{jj'} = \mathbb{P}j \in R, j' \in R$
$\mathbf{P}^t \in \mathbb{R}^{n \times n}$	Matrix defined by $\mathbf{P}^t_{jj'} = \mathbb{P}j \in R_t, j' \in R_t$
Linear operators $\mathbb{R}^{d \times n} \rightarrow \mathbb{R}^{d \times n}$	
\mathcal{U}	any unbiased operator: $\mathbb{E}[\mathcal{U}\mathbf{X}] \equiv \mathbf{X}$, i.e., $\mathbb{E}[\mathcal{U}] \equiv \mathcal{I}$
\mathcal{S}	any random projection operator
\mathcal{M}	operator defined via $(\mathcal{M}\mathbf{X})_{:j} = \mathbf{M}_j \mathbf{X}_{:j}$
\mathcal{B}	(a technical) operator used to define the Lyapunov function (5.11)
\mathcal{R}	(a technical) operator such that $\mathbf{J}^k - \mathbf{G}(\mathbf{x}^*) \in \text{Range}(\mathcal{R})$
Miscellaneous	
Γ	Random operator $\Gamma : \mathbb{R}^{d \times n} \rightarrow \mathbb{R}^d$ defined by $\Gamma\mathbf{X} = \mathcal{U}\mathbf{X}\mathbf{e}$

Table A.4: Summary of frequently used notation specific to Chapter 5.

Chapter 9	
From main body of the chapter	
$\mathbf{S} \in \mathbb{R}^{d, \tau(\mathbf{S})}$	Random matrix sampled from distribution \mathcal{D} from (9.2)
S	Random subset of $\{1, \dots, d\}$ from (9.2)
$M_{\mathbf{S}}$	Lipschitz constant of $\nabla^2 f(x)$ on the range of \mathbf{S} from (9.4)
M	Lipschitz constant of $\nabla^2 f(x)$ on \mathbb{R}^d ; $M = M_{\mathbf{I}^d}$
L	Lipschitz constant of $\nabla f(x)$ on \mathbb{R}^d
$\mathbf{A}_{\mathbf{S}}$	$\stackrel{\text{def}}{=} \mathbf{S}^\top \mathbf{A} \mathbf{S} \in \mathbb{R}^{\tau(\mathbf{S}) \times \tau(\mathbf{S})}$, for a given matrix $\mathbf{A} \in \mathbb{R}^{d \times d}$
$\nabla_{\mathbf{S}} f(x)$	$\stackrel{\text{def}}{=} \mathbf{S}^\top \nabla f(x)$
$\nabla_{\mathbf{S}}^2 f(x)$	$\stackrel{\text{def}}{=} (\nabla^2 f(x))_{\mathbf{S}} = \mathbf{S}^\top \nabla^2 f(x) \mathbf{S}$
$\mathbf{H}_{\mathbf{S}}(x)$	$\stackrel{\text{def}}{=} \nabla_{\mathbf{S}}^2 f(x) + \sqrt{\frac{M_{\mathbf{S}}}{2}} \ \nabla_{\mathbf{S}} f(x)\ ^{\frac{1}{2}} \mathbf{I}^{\tau(\mathbf{S})}$ from Lemma 9.7.1
ζ	$\stackrel{\text{def}}{=} \lambda_{\min} \left((\nabla^2 f(x^*))^{\frac{1}{2}} \mathbb{E} [\mathbf{S} (\nabla_{\mathbf{S}}^2 f(x^*))^{-1} \mathbf{S}^\top] (\nabla^2 f(x^*))^{\frac{1}{2}} \right)$ from (9.13)
\mathbf{Z}	$\stackrel{\text{def}}{=} \mathbf{S} (\mathbf{S}^\top \mathbf{S})^{-1} \mathbf{S}^\top$, the projection onto range of \mathbf{S} from Section 9.6.1
R	$\stackrel{\text{def}}{=} \sup_{x \in \mathbb{R}^d} \left\{ \ x - x^*\ : F(x) \leq F(x^0) \right\}$ from (9.9)
$\lambda_f(x)$	$\stackrel{\text{def}}{=} \left(\nabla f(x)^\top (\nabla^2 f(x))^{-1} \nabla f(x) \right)^{\frac{1}{2}}$, Newton decrement from (I.4)
χ^0	$\stackrel{\text{def}}{=} \{x; f(x) \leq f(x^0)\}$, sublevel set

Table A.5: Summary of frequently used notation specific to Chapter 9.

Appendix B

Appendix for Chapter 2

B.1 Proof of Theorem 2.3.2

Before starting the proof, we mention that the proof technique we use is inspired by the work [6, 7], which takes the advantage of the coupling of gradient descent with mirror descent, resulting in a relatively simple proof.

B.1.1 Proof of inequality (2.14)

By comparing (2.12) and (2.6) for $h = e_i$, we get $\mu_w w_i \leq \mathbf{M}_{ii}$, and the first inequality in (2.14) follows. Using (2.7) it follows that $e_i^\top (\mathbf{P} \circ \mathbf{M}) e_i \leq e_i^\top \mathbf{Diag}(p \circ v) e_i$, which in turn implies $\mathbf{M}_{ii} \leq v_i$ and the second inequality in (2.14) follows.

B.1.2 Descent lemma

The following lemma is a consequence of \mathbf{M} -smoothness of f , and ESO inequality (2.7).

Lemma B.1.1. *Under the assumptions of Theorem 2.3.2, for all $k \geq 0$ we have the bound*

$$f(x^{k+1}) - \mathbb{E} [f(y^{k+1}) | x^{k+1}] \geq \frac{1}{2} \|\nabla f(x^{k+1})\|_{v^{-1}op}^2. \quad (\text{B.1})$$

Proof. We have

$$\begin{aligned} & \mathbb{E} [f(y^{k+1}) | x^{k+1}] \\ \stackrel{(6)}{=} & \mathbb{E} \left[f \left(x^{k+1} - \sum_{i \in S^k} \frac{1}{v_i} \nabla_i f(x^{k+1}) e_i \right) \mid x^{k+1} \right] \\ \stackrel{(2.6)}{\leq} & f(x^{k+1}) - \|\nabla f(x^{k+1})\|_{v^{-1}op}^2 + \frac{1}{2} \mathbb{E} \left[\left\| \sum_{i \in S^k} \frac{1}{v_i} \nabla_i f(x^{k+1}) e_i \right\|_{\mathbf{M}}^2 \mid x^{k+1} \right] \\ \stackrel{(2.7)}{\leq} & f(x^{k+1}) - \|\nabla f(x^{k+1})\|_{v^{-1}op}^2 + \frac{1}{2} \|\nabla f(x^{k+1})\|_{v^{-1}op}^2. \end{aligned}$$

□

B.1.3 Key technical inequality

We first establish a lemma which will play a key part in the analysis.

Lemma B.1.2. *For every u we have*

$$\begin{aligned} \eta \sum_{i \in S^k} \left\langle \frac{1}{p_i} \nabla_i f(x^{k+1}) e_i, z^{k+1} - u \right\rangle - \frac{\eta \mu_w}{2} \|x^{k+1} - u\|_w^2 \\ \leq -\frac{1}{2} \|z^k - z^{k+1}\|_w^2 + \frac{1}{2} \|z^k - u\|_w^2 - \frac{1 + \eta \mu_w}{2} \|z^{k+1} - u\|_w^2. \end{aligned}$$

Proof. The proof is a direct generalization of the proof of analogous lemma of [7]. We include it for completeness. Notice that (7) is equivalent to

$$z^{k+1} = \arg \min_z h^k(z) \stackrel{\text{def}}{=} \arg \min_z \frac{1}{2} \|z - z^k\|_w^2 + \eta \sum_{i \in S^k} \left\langle \frac{1}{p_i} \nabla_i f(x^{k+1}) e_i, z \right\rangle + \frac{\eta \mu_w}{2} \|z - x^{k+1}\|_w^2.$$

Therefore, we have for every u

$$\begin{aligned} 0 &= \langle \nabla h^k(z^{k+1}), z^{k+1} - u \rangle_w \\ &= \langle z^{k+1} - z^k, z^{k+1} - u \rangle_w + \eta \sum_{i \in S^k} \left\langle \frac{1}{p_i} \nabla_i f(x^{k+1}) e_i, z^{k+1} - u \right\rangle \\ &\quad + \eta \mu_w \langle z^{k+1} - x^{k+1}, z^{k+1} - u \rangle_w. \end{aligned} \tag{B.2}$$

Next, by generalized Pythagorean theorem we have

$$\langle z^{k+1} - z^k, z^{k+1} - u \rangle_w = \frac{1}{2} \|z^k - z^{k+1}\|_w^2 - \frac{1}{2} \|z^k - u\|_w^2 + \frac{1}{2} \|u - z^{k+1}\|_w^2 \tag{B.3}$$

and

$$\langle z^{k+1} - x^{k+1}, z^{k+1} - u \rangle_w = \frac{1}{2} \|x^{k+1} - z^{k+1}\|_w^2 - \frac{1}{2} \|x^{k+1} - u\|_w^2 + \frac{1}{2} \|u - z^{k+1}\|_w^2. \tag{B.4}$$

It remains to put (B.3) and (B.4) into (B.2). \square

B.1.4 Proof of the theorem

To mitigate notational burden, consider all expectations in this proof to be taken with respect to the choice of the random subset of coordinates S^k . Using Lemma B.1.2 we

have

$$\begin{aligned}
& \eta \sum_{i \in S^k} \left\langle \frac{1}{p_i} \nabla_i f(x^{k+1}) e_i, z^k - u \right\rangle - \frac{\eta \mu_w}{2} \|x^{k+1} - u\|_w^2 \\
& \leq \eta \sum_{i \in S^k} \left\langle \frac{1}{p_i} \nabla_i f(x^{k+1}) e_i, z^k - z^{k+1} \right\rangle - \frac{1}{2} \|z^k - z^{k+1}\|_w^2 + \frac{1}{2} \|z^k - u\|_w^2 \\
& \quad - \frac{1 + \eta \mu_w}{2} \|z^{k+1} - u\|_w^2 \\
& \leq \frac{\eta^2}{2} \left\| \sum_{i \in S^k} \frac{1}{p_i} \nabla_i f(x^{k+1}) e_i \right\|_{w^{-1}}^2 + \frac{1}{2} \|z^k - u\|_w^2 - \frac{1 + \eta \mu_w}{2} \|z^{k+1} - u\|_w^2 \\
& = \frac{\eta^2}{2} \left\| \sum_{i \in S^k} \nabla_i f(x^{k+1}) e_i \right\|_{w^{-1} \circ p^{-2}}^2 + \frac{1}{2} \|z^k - u\|_w^2 - \frac{1 + \eta \mu_w}{2} \|z^{k+1} - u\|_w^2.
\end{aligned}$$

Taking the expectation over the choice of S^k , we get

$$\begin{aligned}
& \eta \langle \nabla f(x^{k+1}), z^k - u \rangle - \frac{\eta \mu_w}{2} \|x^{k+1} - u\|_w^2 \\
& \leq \frac{\eta^2}{2} \|\nabla f(x^{k+1})\|_{w^{-1} \circ p^{-1}}^2 + \frac{1}{2} \|z^k - u\|_w^2 - \frac{1 + \eta \mu_w}{2} \mathbb{E} [\|z^{k+1} - u\|_w^2] \\
& \stackrel{(2.13)}{=} \frac{\eta^2}{2} \|\nabla f(x^{k+1})\|_{v^{-1} \circ p}^2 + \frac{1}{2} \|z^k - u\|_w^2 - \frac{1 + \eta \mu_w}{2} \mathbb{E} [\|z^{k+1} - u\|_w^2] \\
& \stackrel{(B.1)}{\leq} \eta^2 (f(x^{k+1}) - \mathbb{E} [f(y^{k+1})]) + \frac{1}{2} \|z^k - u\|_w^2 - \frac{1 + \eta \mu_w}{2} \mathbb{E} [\|z^{k+1} - u\|_w^2].
\end{aligned}$$

Next, we have the following bounds

$$\begin{aligned}
\eta (f(x^{k+1}) - f(x^*)) & \stackrel{(2.12)}{\leq} \eta \langle \nabla f(x^{k+1}), x^{k+1} - x^* \rangle - \frac{\eta \mu_w}{2} \|x^* - x^{k+1}\|_w^2 \\
& = \eta \langle \nabla f(x^{k+1}), x^{k+1} - z^k \rangle + \eta \langle \nabla f(x^{k+1}), z^k - x^* \rangle \\
& \quad - \frac{\eta \mu_w}{2} \|x^* - x^{k+1}\|_w^2 \\
& \stackrel{(4)}{=} \frac{(1 - \theta) \eta}{\theta} \langle \nabla f(x^{k+1}), y^k - x^{k+1} \rangle + \eta \langle \nabla f(x^{k+1}), z^k - x^* \rangle \\
& \quad - \frac{\eta \mu_w}{2} \|x^* - x^{k+1}\|_w^2 \\
& \stackrel{(B.5)}{\leq} \frac{(1 - \theta) \eta}{\theta} (f(y^k) - f(x^{k+1})) + \eta^2 (f(x^{k+1}) - \mathbb{E} [f(y^{k+1})]) \\
& \quad + \frac{1}{2} \|z^k - x^*\|_w^2 - \frac{1 + \eta \mu_w}{2} \mathbb{E} [\|z^{k+1} - x^*\|_w^2].
\end{aligned}$$

Choosing $\eta = \frac{1}{\theta}$ and rearranging the above we obtain

$$\begin{aligned} & \frac{1}{\theta^2} (\mathbb{E} [f(y^{k+1})] - f(x^*)) + \frac{1 + \frac{\mu_w}{\theta}}{2} \mathbb{E} [\|z^{k+1} - x^*\|_w^2] \\ & \leq \frac{(1 - \theta)}{\theta^2} (f(y^k) - f(x^*)) + \frac{1}{2} \|z^k - x^*\|_w^2 \end{aligned}$$

Finally, setting θ such that $1 + \frac{\mu_w}{\theta} = \frac{1}{1 - \theta}$, which coincides with (2.16), we get

$$\mathbb{E} [P^{k+1}] \leq (1 - \theta)P^k,$$

as desired.

B.2 Better rates for minibatch CD (without acceleration)

In this section we establish better rates for minibatch CD method than the current state of the art. Our starting point is the following complexity theorem.

Theorem B.2.1. *Choose any proper sampling and let \mathbf{P} be its probability matrix and p its probability vector. Let*

$$c(S, \mathbf{M}) \stackrel{\text{def}}{=} \lambda_{\max}(\mathbf{P}'' \circ \mathbf{M}),$$

where $\mathbf{P}'' \stackrel{\text{def}}{=} \mathbf{D}^{-1}\mathbf{P}\mathbf{D}^{-1}$ and $\mathbf{D} \stackrel{\text{def}}{=} \text{Diag}(p)$. Then the vector v defined by $v_i = c(S, \mathbf{M})p_i$ satisfies the ESO inequality (2.7). Moreover, if we run the non-accelerated CD method (2.5) with this sampling and stepsizes $\alpha_i = \frac{1}{c(S, \mathbf{M})p_i}$, then the iteration complexity of the method is

$$\frac{c(S, \mathbf{M})}{\mu} \log \frac{1}{\epsilon}. \quad (\text{B.5})$$

Proof. Let $v_i = cp_i$ for all i . The ESO inequality holds for this choice of v if $\mathbf{P} \circ \mathbf{M} \preceq c\mathbf{D}^2$. This is equivalent to Since $\mathbf{D}^{-1}(\mathbf{P} \circ \mathbf{M})\mathbf{D}^{-1} = \mathbf{P}'' \circ \mathbf{M}$, the above inequality is equivalent to $\mathbf{P}'' \circ \mathbf{M} \preceq c\mathbf{I}$, which is equivalent to $c \geq \lambda_{\max}(\mathbf{P}'' \circ \mathbf{M})$. So, choosing $c = c(S, \mathbf{M})$ works. Plugging this choice of v into the complexity result (2.8) gives (B.5). \square

B.2.1 Two uniform samplings and one new importance sampling

In the next theorem we compute now consider several special samplings. All of them choose in expectation a minibatch of size τ and are hence directly comparable.

Theorem B.2.2. *The following statements hold:*

(i) *Let S_1 be the τ -nice sampling. Then*

$$c_1 \stackrel{\text{def}}{=} c(S_1, \mathbf{M}) = \frac{d}{\tau} \lambda_{\max} \left(\frac{\tau - 1}{d - 1} \mathbf{M} + \frac{d - \tau}{d - 1} \text{Diag}(\mathbf{M}) \right). \quad (\text{B.6})$$

(ii) *Let S_2 be the independent uniform sampling with minibatch size τ . That is, for all i we independently decide whether $i \in S$, and do so by picking i with probability*

$p_i = \frac{\tau}{d}$. Then

$$c_2 \stackrel{\text{def}}{=} c(S_2, \mathbf{M}) = \lambda_{\max} \left(\mathbf{M} + \frac{d - \tau}{\tau} \text{Diag}(\mathbf{M}) \right). \quad (\text{B.7})$$

(iii) Let S_3 be an independent sampling where we choose $p_i \propto \frac{\mathbf{M}_{ii}}{\delta + \mathbf{M}_{ii}}$ where $\delta > 0$ is chosen so that $\sum_i p_i = \tau$. Then

$$c_3 \stackrel{\text{def}}{=} c(S_3, \mathbf{M}) = \lambda_{\max}(\mathbf{M}) + \delta. \quad (\text{B.8})$$

Moreover,

$$\delta \leq \frac{\text{Tr}(\mathbf{M})}{\tau}. \quad (\text{B.9})$$

Proof. We will deal with each case separately:

(i) The probability matrix of S_1 is $\mathbf{P} = \frac{\tau}{d} (\beta \mathbf{E} + (1 - \beta) \mathbf{I})$, where $\beta = \frac{\tau - 1}{d - 1}$, and $\mathbf{D} = \frac{\tau}{d} \mathbf{I}$. Hence,

$$\begin{aligned} \mathbf{P}'' \circ \mathbf{M} &= (\mathbf{D}^{-1} \mathbf{P} \mathbf{D}^{-1}) \circ \mathbf{M} \\ &= \frac{\tau}{d} (\beta \mathbf{D}^{-1} \mathbf{E} \mathbf{D}^{-1} + (1 - \beta) \mathbf{D}^{-2}) \circ \mathbf{M} \\ &= \frac{\tau}{d} \left(\frac{\tau - 1}{d - 1} \mathbf{E} + \frac{d - \tau}{d - 1} \mathbf{I} \right) \circ \mathbf{M} \\ &= \frac{\tau}{d} \left(\frac{\tau - 1}{d - 1} \mathbf{M} + \frac{d - \tau}{d - 1} \text{Diag}(\mathbf{M}) \right). \end{aligned}$$

(ii) The probability matrix of S_2 is $\mathbf{P} = \frac{\tau}{d} (\frac{\tau}{d} \mathbf{E} + (1 - \frac{\tau}{d}) \mathbf{I})$, and $\mathbf{D} = \frac{\tau}{d} \mathbf{I}$. Hence,

$$\begin{aligned} \mathbf{P}'' \circ \mathbf{M} &= (\mathbf{D}^{-1} \mathbf{P} \mathbf{D}^{-1}) \circ \mathbf{M} \\ &= \frac{\tau}{d} \left(\frac{\tau}{d} \mathbf{D}^{-1} \mathbf{E} \mathbf{D}^{-1} + \left(1 - \frac{\tau}{d} \right) \mathbf{D}^{-2} \right) \circ \mathbf{M} \\ &= \left(\mathbf{E} + \frac{d - \tau}{\tau} \mathbf{I} \right) \circ \mathbf{M} \\ &= \mathbf{M} + \frac{d - \tau}{\tau} \text{Diag}(\mathbf{M}). \end{aligned}$$

(iii) The probability matrix of S_3 is $\mathbf{P} = p p^\top + \mathbf{D} - \mathbf{D}^2$. Therefore,

$$\begin{aligned} \mathbf{P}'' \circ \mathbf{M} &= (\mathbf{D}^{-1} \mathbf{P} \mathbf{D}^{-1}) \circ \mathbf{M} \\ &= (\mathbf{D}^{-1} p p^\top \mathbf{D}^{-1} + \mathbf{D}^{-1} - \mathbf{I}) \circ \mathbf{M} \\ &= (\mathbf{E} + \mathbf{D}^{-1} - \mathbf{I}) \circ \mathbf{M} \\ &= (\mathbf{E} + \delta (\text{Diag}(\mathbf{M}))^{-1}) \circ \mathbf{M} \\ &= \mathbf{M} + \delta \mathbf{I}. \end{aligned}$$

To establish the bound on δ , it suffices to note that

$$\tau = \sum_i p_i = \sum_i \frac{\mathbf{M}_{ii}}{\delta + \mathbf{M}_{ii}} \leq \sum_i \frac{\mathbf{M}_{ii}}{\delta} = \frac{\text{Tr}(\mathbf{M})}{\delta}.$$

□

B.2.2 Comparing the samplings

In the next result we show that sampling S_3 is at most twice worse than S_2 , which is at most twice worse than S_1 . Note that S_1 is uniform; and it is the standard minibatch sampling used in the literature and applications. Our novel sampling S_3 is *non-uniform*, and is at most four times worse than S_1 in the worst case. However, *it can be substantially better*, as we shall show later by giving an example.

Theorem B.2.3. *The leading complexity terms c_1, c_2 , and c_3 of CD (Algorithm (2.5)) with samplings S_1, S_2 , and S_3 , respectively, defined in Theorem B.2.2, compare as follows:*

$$(i) \quad c_3 \leq \frac{2d-\tau}{d-\tau} c_2$$

$$(ii) \quad c_2 \leq \frac{(d-1)\tau}{d(\tau-1)} c_1 \leq 2c_1$$

Proof. We have:

(i)

$$\begin{aligned} c_3 &\stackrel{(B.8)}{=} \lambda_{\max}(\mathbf{M}) + \delta \\ &\leq \lambda_{\max}\left(\mathbf{M} + \frac{d-\tau}{\tau} \text{Diag}(\mathbf{M})\right) + \delta \\ &\stackrel{(B.7)}{=} c_2 + \delta \\ &\stackrel{(B.9)}{\leq} c_2 + \frac{\text{Tr}(\mathbf{M})}{\tau} \\ &\leq c_2 + \frac{d \max_i \mathbf{M}_{ii}}{\tau} \\ &= c_2 + \frac{d}{d-\tau} \frac{d-\tau}{\tau} \max_i \mathbf{M}_{ii} \\ &= c_2 + \frac{d}{d-\tau} \lambda_{\max}\left(\frac{d-\tau}{\tau} \text{Diag}(\mathbf{M})\right) \\ &\leq c_2 + \frac{d}{d-\tau} \lambda_{\max}\left(\mathbf{M} + \frac{d-\tau}{\tau} \text{Diag}(\mathbf{M})\right) \\ &\stackrel{(B.7)}{=} \frac{2d-\tau}{d-\tau} c_2. \end{aligned}$$

(ii)

$$\begin{aligned}
c_2 &\stackrel{(B.7)}{=} \lambda_{\max} \left(\mathbf{M} + \frac{d-\tau}{\tau} \mathbf{Diag}(\mathbf{M}) \right) \\
&= \lambda_{\max} \left(\frac{d(\tau-1)}{\tau(d-1)} \mathbf{M} + \frac{d-\tau}{\tau} \mathbf{Diag}(\mathbf{M}) + \left(1 - \frac{d(\tau-1)}{\tau(d-1)} \right) \mathbf{M} \right) \\
&\stackrel{(\dagger)}{\leq} \lambda_{\max} \left(\frac{d(\tau-1)}{\tau(d-1)} \mathbf{M} + \frac{d-\tau}{\tau} \mathbf{Diag}(\mathbf{M}) \right) + \lambda_{\max} \left(\left(1 - \frac{d(\tau-1)}{\tau(d-1)} \right) \mathbf{M} \right) \\
&\leq \lambda_{\max} \left(\frac{d(\tau-1)}{\tau(d-1)} \mathbf{M} + \frac{d(d-\tau)}{\tau(d-1)} \mathbf{Diag}(\mathbf{M}) \right) + \frac{d-\tau}{(d-1)\tau} \lambda_{\max}(\mathbf{M}) \\
&\stackrel{(B.6)}{=} c_1 + \frac{d-\tau}{(d-1)\tau} \lambda_{\max}(\mathbf{M}) \\
&\stackrel{(B.7)}{\leq} c_1 + \frac{d-\tau}{(d-1)\tau} c_2.
\end{aligned}$$

The statement follows by reshuffling the final inequality. In step (\dagger) we have used subadditivity of the function $\mathbf{A} \mapsto \lambda_{\max}(\mathbf{A})$.

□

The next simple example shows that sampling S_3 can be arbitrarily better than sampling S_1 .

Example 12. Consider $d \gg 1$, and choose any τ and

$$\mathbf{M} \stackrel{\text{def}}{=} \begin{pmatrix} n & 0^\top \\ 0 & \mathbf{I} \end{pmatrix}$$

for $\mathbf{I} \in \mathbb{R}^{(d-1) \times (d-1)}$. Then, it is easy to verify that $c_1 \stackrel{(B.6)}{=} \frac{d^2}{\tau}$ and $c_3 \stackrel{(B.8)+(B.9)}{\leq} n + \frac{2n-1}{\tau} = \mathcal{O}(\frac{d}{\tau})$. Thus, convergence rate of CD with S_3 sampling can be up to $\mathcal{O}(d)$ times better than convergence rate of CD with τ -nice sampling.

Remark 19. Looking only at diagonal elements of \mathbf{M} , an intuition tells us that one should sample a coordinate corresponding to larger diagonal entry of \mathbf{M} with higher probability. However, this might lead to worse convergence, comparing to τ -nice sampling. Therefore the results we provide in this section cannot be qualitatively better, i.e. there are examples of smoothness matrix, for which assigning bigger probability to bigger diagonal elements leads to worse rate. It is an easy exercise to verify that for $\mathbf{M} \in \mathbb{R}^{10 \times 10}$ such that

$$\mathbf{M} \stackrel{\text{def}}{=} \begin{pmatrix} 2 & 0^\top \\ 0 & 11^\top \end{pmatrix},$$

and $\tau \geq 2$ we have $c(S_{\text{nice}}, \mathbf{M}) \leq c(S', \mathbf{M})$ for any S' satisfying $p(S')_i \geq p(S')_j$ if and only if $\mathbf{M}_{ii} \geq \mathbf{M}_{jj}$.

B.3 Proofs for Section 2.4

B.3.1 Proof of Theorem 2.4.1

We start with a lemma which allows us to focus on ESO parameters v_i which are proportional to the squares of the probabilities p_i .

Lemma B.3.1. *Assume that the ESO inequality (2.7) holds. Let $j = \arg \max_i \frac{v_i}{p_i^2}$, $c = \frac{v_j}{p_j^2}$ and $v' = cp^2$ (i.e., $v'_i = cp_i^2$ for all i). Then the following statements hold:*

- (i) $v' \geq v$.
- (ii) ESO inequality (2.7) holds for v' also.
- (iii) Assuming f is μ -convex, Theorem 2.3.2 holds if we replace v by v' , and the rate (2.19) is unchanged if we replace v by v' .

Proof. (i) $v'_i = cp_i^2 = \frac{v_j}{p_j^2} p_i^2 = \left(\frac{v_j}{p_j^2} \frac{p_i^2}{v_i} \right) v_i \geq v_i$.

(ii) This follows directly from (i).

(iii) Theorem 2.3.2 holds with v replaced by v' because ESO holds. To show that the rates are unchanged first note that $\max_i \frac{v_i}{p_i^2} = \frac{v_j}{p_j^2} = c$. On the other hand, by construction, we have $c = \frac{v'_i}{p_i^2}$ for all i . So, in particular, $c = \max_i \frac{v'_i}{p_i^2}$. □

In view of the above lemma, we can assume without loss of generality that $v = cp^2$. Hence, the rate in (2.19) can be written in the form

$$\sqrt{\max_i \frac{v_i}{p_i^2 \mu}} = \sqrt{\frac{c}{\mu}}. \quad (\text{B.10})$$

In what follows, we will establish a lower bound on c , which will lead to the lower bound on the rate expressed as inequality (2.19). As a starting point, note that directly from (2.7) we get the bound

$$\mathbf{P} \circ \mathbf{M} \preceq \mathbf{Diag}(p \circ v) = c \mathbf{Diag}(p^3). \quad (\text{B.11})$$

Let $\mathbf{D}_1 = \mathbf{Diag}(p)^{-1/2}$ and $\mathbf{D}_2 = \mathbf{Diag}(p)^{-1}$. From (B.11) we get $\mathbf{D}_1 \mathbf{D}_2 (\mathbf{P} \circ \mathbf{M}) \mathbf{D}_2 \mathbf{D}_1 \preceq c \mathbf{I}$ and hence

$$c \geq c(S, \mathbf{M}) \stackrel{\text{def}}{=} \lambda_{\max}(\mathbf{D}_1 \mathbf{D}_2 (\mathbf{P} \circ \mathbf{M}) \mathbf{D}_2 \mathbf{D}_1). \quad (\text{B.12})$$

At this point, the following identity will be useful.

Lemma B.3.2. *Let $\mathbf{A}, \mathbf{B}, \mathbf{D}_1, \mathbf{D}_2 \in \mathbb{R}^{d \times d}$, with $\mathbf{D}_1, \mathbf{D}_2$ being diagonal. Then*

$$\mathbf{D}_1 (\mathbf{A} \circ \mathbf{B}) \mathbf{D}_2 = (\mathbf{D}_1 \mathbf{A} \mathbf{D}_2) \circ \mathbf{B} = \mathbf{A} \circ (\mathbf{D}_1 \mathbf{B} \mathbf{D}_2). \quad (\text{B.13})$$

Proof. The proof is straightforward, and hence we do not include it. The identity is formulated as an exercise in [221]. \square

Repeatedly applying Lemma B.3.2, we get

$$\mathbf{D}_1 \mathbf{D}_2 (\mathbf{P} \circ \mathbf{M}) \mathbf{D}_2 \mathbf{D}_1 = \underbrace{(\mathbf{D}_1 \mathbf{P} \mathbf{D}_1)}_{\mathbf{P}'} \circ \underbrace{(\mathbf{D}_2 \mathbf{M} \mathbf{D}_2)}_{\mathbf{M}'}.$$

Plugging this back into (B.12), and since $\mathbf{P}'_{ii} = 1$ for all i , we get the bound

$$\begin{aligned} c &\geq c(S, \mathbf{M}) = \lambda_{\max}(\mathbf{P}' \circ \mathbf{M}') \geq \max_i (\mathbf{P}' \circ \mathbf{M}')_{ii} = \max_i \mathbf{P}'_{ii} \mathbf{M}'_{ii} = \max_i \mathbf{M}'_{ii} \\ &= \max_i \frac{\mathbf{M}_{ii}}{p_i^2} \geq \frac{\left(\sum_{i=1}^d \mathbf{M}_{ii}^{1/2}\right)^2}{\tau^2}. \end{aligned} \quad (\text{B.14})$$

The last inequality follows by observing that the optimal solution of the optimization problem

$$\min_p \left\{ \max_i \frac{\mathbf{M}_{ii}}{p_i^2} \mid p_1, \dots, p_d > 0, \sum_i p_i = \tau \right\}$$

is $p_i = \tau \frac{\mathbf{M}_{ii}^{1/2}}{\sum_j \mathbf{M}_{jj}^{1/2}}$. Inequality (2.19) now follows by substituting the lower bound on c obtained in (B.14) into (B.10).

B.3.2 Proof of Lemma 2.4.2

$$\begin{aligned} \text{Diag}(p_1 v_1, \dots, p_d v_d) &= c(S, \mathbf{M}) \text{Diag}(p_1^3, \dots, p_d^3) \\ &= c(S, \mathbf{M}) \mathbf{D}^3 \\ &= \lambda_{\max}((\mathbf{D}^{-1/2} \mathbf{P} \mathbf{D}^{-1/2}) \circ (\mathbf{D}^{-1} \mathbf{M} \mathbf{D}^{-1})) \mathbf{D}^3 \\ &\succeq \mathbf{D}^{\frac{3}{2}} ((\mathbf{D}^{-1/2} \mathbf{P} \mathbf{D}^{-1/2}) \circ (\mathbf{D}^{-1} \mathbf{M} \mathbf{D}^{-1})) \mathbf{D}^{\frac{3}{2}} \\ &\stackrel{(\text{B.13})}{=} \mathbf{P} \circ \mathbf{M}. \end{aligned}$$

The last inequality came from the fact that \mathbf{D} is diagonal.

B.3.3 Bound on $c(S_1, \mathbf{M})$

Lemma B.3.3. $c(S_1, \mathbf{M}) \leq \frac{d^2}{\tau^2}((1 - \beta) \max_i \mathbf{M}_{ii} + \beta L)$.

Proof. Recall that the probability matrix of S_1 is $\mathbf{P} = \frac{\tau}{d}((1 - \beta)\mathbf{I} + \beta\mathbf{E})$. Since $p_i = \frac{\tau}{d}$

and $\mathbf{M} \preceq L\mathbf{I}$, we have

$$\begin{aligned}
c(S_1, \mathbf{M}) &= \lambda_{\max}(\mathbf{P}' \circ \mathbf{M}') \\
&= \lambda_{\max}((\mathbf{D}^{-1/2}\mathbf{P}\mathbf{D}^{-1/2}) \circ (\mathbf{D}^{-1}\mathbf{M}\mathbf{D}^{-1})) \\
&= \lambda_{\max}\left(\frac{\tau}{d}((1-\beta)\mathbf{D}^{-1} + \beta\mathbf{D}^{-1/2}\mathbf{E}\mathbf{D}^{-1/2}) \circ \mathbf{D}^{-1}\mathbf{M}\mathbf{D}^{-1}\right) \\
&= \frac{\tau}{d}\lambda_{\max}(((1-\beta)\mathbf{D}^{-1} + \beta\mathbf{D}^{-1/2}\mathbf{E}\mathbf{D}^{-1/2}) \circ \mathbf{D}^{-1}\mathbf{M}\mathbf{D}^{-1}) \\
&= \frac{\tau}{d}\lambda_{\max}((1-\beta)\mathbf{Diag}(\mathbf{M}_{ii}/p_i^3) + \beta\mathbf{D}^{-3/2}\mathbf{M}\mathbf{D}^{-3/2}) \\
&\preceq \frac{\tau}{d}\lambda_{\max}((1-\beta)\mathbf{Diag}(\mathbf{M}_{ii}/p_i^3) + \beta L\mathbf{D}^{-3}) \\
&= \frac{\tau}{d}\lambda_{\max}\left((1-\beta)\frac{d^3}{\tau^3}\max_i \mathbf{M}_{ii} + \beta L\frac{d^3}{\tau^3}\right) \\
&= \frac{d^2}{\tau^2}\left((1-\beta)\max_i \mathbf{M}_{ii} + \beta L\right).
\end{aligned}$$

□

B.3.4 Proof of Theorem 2.4.3

For the purpose of this proof, let S_2 be the independent uniform sampling with minibatch size τ . That is, for all i we independently decide whether $i \in S$, and do so by picking i with probability $p_i = \frac{\tau}{d}$. Recall that S_3 is the independent importance sampling.

For simplicity, let \mathbf{P}_i be the probability matrix of sampling S_i , $\mathbf{D}_i \stackrel{\text{def}}{=} \mathbf{Diag}(\mathbf{P}_i)$, and $\mathbf{M}'_i \stackrel{\text{def}}{=} \mathbf{D}_i^{-1/2}\mathbf{M}\mathbf{D}_i^{-1/2}$, for $i = 1, 3$. Next, we have

$$\begin{aligned}
c(S_i, \mathbf{M}) &= \lambda_{\max}\left((\mathbf{D}_i^{-1/2}\mathbf{P}_i\mathbf{D}_i^{-1/2}) \circ (\mathbf{D}_i^{-1}\mathbf{M}\mathbf{D}_i^{-1})\right) \\
&\stackrel{\text{(B.13)}}{=} \lambda_{\max}\left((\mathbf{D}_i^{-1}\mathbf{P}_i\mathbf{D}_i^{-1}) \circ (\mathbf{D}_i^{-1/2}\mathbf{M}\mathbf{D}_i^{-1/2})\right) \\
&= \lambda_{\max}((\mathbf{E} + \mathbf{D}_i^{-1} - \mathbf{I}) \circ \mathbf{M}'_i) \\
&= \lambda_{\max}(\mathbf{M}'_i + \mathbf{Diag}(\mathbf{M}'_i) \circ (\mathbf{D}_i^{-1} - \mathbf{I})), \tag{B.15}
\end{aligned}$$

where the third identity holds since both S_i is an independent sampling, which means that $(\mathbf{D}_i^{-1}\mathbf{P}_i\mathbf{D}_i^{-1})_{kl} = \frac{p_{kl}}{p_k p_l}$, where $p = \mathbf{Diag}(\mathbf{D}_i)$.

Denote $c_i \stackrel{\text{def}}{=} c(S_i, \mathbf{M})$. Thus for S_2 we have

$$c_2 = \frac{\tau}{d}\lambda_{\max}\left(\mathbf{M} + \frac{d-\tau}{\tau}\mathbf{Diag}(\mathbf{M})\right). \tag{B.16}$$

Let us now establish a technical lemma.

Lemma B.3.4.

$$\lambda_{\max}(\mathbf{M}'_3 + \mathbf{Diag}(\mathbf{M}'_3) \circ (\mathbf{D}_3^{-1} - \mathbf{I})) \leq \frac{2d - \tau}{d - \tau} \lambda_{\max} \left(\mathbf{M}'_3 + \frac{d - \tau}{\tau} \mathbf{Diag}(\mathbf{M}'_3) \right). \quad (\text{B.17})$$

Proof. The statement follows immediately repeating the steps of the proof of (i) from Theorem B.2.3 using the fact that for sampling S_3 we have $p_i/\mathbf{M}_{ii} \propto p_i^{-1} - 1$. \square

We can now proceed with comparing c_2 to c_3 .

$$\begin{aligned} c_3 &= \lambda_{\max}(\mathbf{M}'_3 + \mathbf{Diag}(\mathbf{M}'_3) \circ (\mathbf{D}_3^{-1} - \mathbf{I})) \\ &\stackrel{(\text{B.17})}{\leq} \frac{2d - \tau}{d - \tau} \lambda_{\max} \left(\mathbf{M}'_3 + \frac{d - \tau}{\tau} \mathbf{Diag}(\mathbf{M}'_3) \right) \\ &\stackrel{(*)}{\leq} \frac{2d - \tau}{d - \tau} \lambda_{\max} \left(d \mathbf{Diag}(\mathbf{M}'_3) + \frac{d - \tau}{\tau} \mathbf{Diag}(\mathbf{M}'_3) \right) \\ &= \frac{2d - \tau}{d - \tau} \frac{d\tau + d - \tau}{\tau} \lambda_{\max}(\mathbf{Diag}(\mathbf{M}'_3)) \\ &\stackrel{(**)}{\leq} \frac{2d - \tau}{d - \tau} \frac{d\tau + d - \tau}{\tau} \frac{\tau}{d} \lambda_{\max}(\mathbf{Diag}(\mathbf{M})) \\ &\leq \frac{2d - \tau}{d - \tau} \frac{d\tau + d - \tau}{\tau} \frac{\tau}{d} \frac{\tau}{d - \tau} \lambda_{\max} \left(\mathbf{M} + \frac{d - \tau}{\tau} \mathbf{Diag}(\mathbf{M}) \right) \\ &\stackrel{(\text{B.15})}{=} \frac{2d - \tau}{d - \tau} \frac{d\tau + d - \tau}{\tau} \frac{\tau}{d - \tau} c_2. \end{aligned} \quad (\text{B.18})$$

Above, inequality $(*)$ holds since for any $d \times d$ matrix $\mathbf{Q} \succ 0$ we have $\mathbf{Q} \preceq d \mathbf{Diag}(\mathbf{Q})$ and inequality $(**)$ holds since $(\mathbf{D}_3)_{ii} \geq (\mathbf{D}_3)_{jj}$ if and only if $\mathbf{M}_{ii} \geq \mathbf{M}_{jj}$ due to choice of p .

Let us now compare to c_2 and c_1 . We have

$$\begin{aligned} c_1 &= \lambda_{\max} \left(\left(\mathbf{D}_1^{-1/2} \mathbf{P}_1 \mathbf{D}_1^{-1/2} \right) \circ \left(\mathbf{D}_1^{-1} \mathbf{M} \mathbf{D}_1^{-1} \right) \right) \\ &\stackrel{(\text{B.13})}{=} \lambda_{\max} \left(\left(\mathbf{D}_1^{-1} \mathbf{P}_1 \mathbf{D}_1^{-1} \right) \circ \left(\mathbf{D}_1^{-1/2} \mathbf{M} \mathbf{D}_1^{-1/2} \right) \right) \\ &= \lambda_{\max} \left(\left(\left(\frac{\tau - 1}{d - 1} \frac{d}{\tau} \mathbf{E} + \frac{d}{\tau} \mathbf{I} - \frac{\tau - 1}{d - 1} \frac{d}{\tau} \mathbf{I} \right) \circ \mathbf{M}_1'' \right) \right) \\ &= \frac{d}{\tau} \lambda_{\max} \left(\frac{\tau - 1}{d - 1} \mathbf{M}_1'' + \frac{d - \tau}{d - 1} \mathbf{Diag}(\mathbf{M}_1'') \right) \\ &= \left(\frac{d}{\tau} \right)^2 \lambda_{\max} \left(\frac{\tau - 1}{d - 1} \mathbf{M} + \frac{d - \tau}{d - 1} \mathbf{Diag}(\mathbf{M}) \right). \end{aligned} \quad (\text{B.19})$$

As (B.16) and (B.19) are established, following the proof of (ii) from Theorem B.2.3, we arrive at

$$c_2 \leq \frac{(d - 1)\tau}{d(\tau - 1)} c_1 \leq 2c_1. \quad (\text{B.20})$$

It remains to combine (B.18) and (B.20) to establish (2.25).

An example with $c_3 \approx \left(\frac{\tau}{d}\right)^2 c_2$ follows.

Example 13. Consider $d \geq 1$, choose any $d \geq \tau \geq 1$ and

$$\mathbf{M} \stackrel{\text{def}}{=} \begin{pmatrix} N & 0^\top \\ 0 & \mathbf{I} \end{pmatrix}$$

for $\mathbf{I} \in \mathbb{R}^{(d-1) \times (d-1)}$. Then, it is easy to verify that $c_1 \stackrel{(\text{B.19})}{=} \left(\frac{d}{\tau}\right)^2 N$. Moreover, for large enough N we have

$$p \approx \left(1, \frac{\tau-1}{d-1}, \dots, \frac{\tau-1}{d-1}\right)^\top \quad \Rightarrow \quad \mathbf{M}'_3 \approx \mathbf{Diag} \left(N, \frac{d-1}{\tau-1}, \dots, \frac{d-1}{\tau-1}\right).$$

Therefore, using (B.15) and again for large enough N , we get $c_3 \approx N$. Thus, $c_3 \approx \left(\frac{\tau}{d}\right)^2 c_2$.

Appendix C

Appendix for Chapter 3

C.1 Proofs for Section 3.4

Lemma C.1.1. *Suppose that f is twice differentiable. Assumption 3.4.1 is equivalent to (2.6) for $\mathbf{Q} = \mathbf{M}^{-1}$.*

Proof. We first establish that Assumption 3.4.1 implies (2.6). Summing up (3.10) for (x, y) and (y, x) yields

$$\langle \nabla f(x) - \nabla f(y), x - y \rangle \geq \|\nabla f(x) - \nabla f(y)\|_{\mathbf{Q}}^2.$$

Using Cauchy Schwartz inequality we obtain

$$\|x - y\|_{\mathbf{Q}^{-1}} \geq \|\nabla f(x) - \nabla f(y)\|_{\mathbf{Q}}.$$

By the mean value theorem, there is $z \in [x, y]$ such that $\nabla f(x) - \nabla f(y) = \nabla^2 f(z)(x - y)$. Thus

$$\|x - y\|_{\mathbf{Q}^{-1}} \geq \|x - y\|_{\nabla^2 f(z) \mathbf{Q} \nabla^2 f(z)}.$$

The above is equivalent to

$$(\nabla^2 f(z))^{-\frac{1}{2}} \mathbf{Q}^{-1} (\nabla^2 f(z))^{-\frac{1}{2}} \succeq (\nabla^2 f(z))^{\frac{1}{2}} \mathbf{Q} (\nabla^2 f(z))^{\frac{1}{2}}$$

Note that for any $\mathbf{M}' \succ 0$ we have $\mathbf{M}' \succeq \mathbf{M}^{-1}$ if and only if $\mathbf{M} \succeq \mathbf{I}$. Thus

$$(\nabla^2 f(z))^{-\frac{1}{2}} \mathbf{Q}^{-1} (\nabla^2 f(z))^{-\frac{1}{2}} \succeq \mathbf{I},$$

which is equivalent to $\mathbf{Q}^{-1} \succeq \nabla^2 f(z)$. To establish the other direction, denote $\phi(y) = f(y) - \langle \nabla f(x), y \rangle$. Clearly, x is minimizer of ϕ and therefore we have

$$\phi(x) \leq \phi(x - \mathbf{M}^{-1} \nabla f(y)) \leq \phi(y) - \frac{1}{2} \|\nabla f(y)\|_{\mathbf{M}^{-1}}^2,$$

which is exactly (3.10) for $\mathbf{Q} = \mathbf{M}^{-1}$. □

Lemma C.1.2. *For $\mathbf{Z}_k \stackrel{\text{def}}{=} \mathbf{S}_k (\mathbf{S}_k^\top \mathbf{S}_k)^\dagger \mathbf{S}_k^\top$, then*

$$\mathbf{Z}_k^\top \mathbf{Z}_k = \mathbf{Z}_k. \tag{C.1}$$

Proof. It is a property of pseudo-inverse that for any matrices \mathbf{A}, \mathbf{I} it holds $((\mathbf{AI})^\dagger)^\top = (\mathbf{I}^\top \mathbf{A}^\top)^\dagger$, so $\mathbf{Z}_k^\top = \mathbf{Z}_k$. Moreover, we also know for any \mathbf{A} that $\mathbf{A}^\dagger \mathbf{A} \mathbf{A}^\dagger = \mathbf{A}^\dagger$ and, thus,

$$\mathbf{Z}_k^\top \mathbf{Z}_k = \mathbf{S}_k (\mathbf{S}_k^\top \mathbf{S}_k)^\dagger \mathbf{S}_k^\top \mathbf{S}_k (\mathbf{S}_k^\top \mathbf{S}_k)^\dagger \mathbf{S}_k^\top = \mathbf{Z}_k.$$

□

C.1.1 Proof of Theorem 3.4.2

We first state two lemmas which will be crucial for the analysis. They characterize key properties of the gradient learning process (3.4), (3.6) and will be used later to bound expected distances of both h^{k+1} and g^k from $\nabla f(x^*)$. The proofs are provided in Appendix C.1.2 and C.1.3 respectively

Lemma C.1.3. *For all $v \in \mathbb{R}^d$ we have*

$$\mathbb{E} [\|h^{k+1} - v\|^2] = \|h^k - v\|_{\mathbf{I} - \mathbb{E}[\mathbf{Z}]}^2 + \|\nabla f(x^k) - v\|_{\mathbb{E}[\mathbf{Z}]}^2. \quad (\text{C.2})$$

Lemma C.1.4. *Let $\mathbf{C} \stackrel{\text{def}}{=} \mathbb{E} [\theta^2 \mathbf{Z}]$. Then for all $v \in \mathbb{R}^d$ we have*

$$\mathbb{E} [\|g^k - v\|^2] \leq 2\|\nabla f(x^k) - v\|_{\mathbf{C}}^2 + 2\|h^k - v\|_{\mathbf{C} - \mathbf{I}}^2.$$

For notational simplicity, it will be convenient to define Bregman divergence between x and y :

$$D_f(x, y) \stackrel{\text{def}}{=} f(x) - f(y) - \langle \nabla f(y), x - y \rangle$$

We can now proceed with the proof of Theorem 3.4.2. Let us start with bounding the first term in the expression for Φ^{k+1} . From Lemma C.1.4 and strong convexity it follows that

$$\begin{aligned} \mathbb{E} [\|x^{k+1} - x^*\|^2] &= \mathbb{E} [\|\text{prox}_{\alpha\psi}(x^k - \alpha g^k) - \text{prox}_{\alpha\psi}(x^* - \alpha \nabla f(x^*))\|^2] \\ &\leq \mathbb{E} [\|x^k - \alpha g^k - (x^* - \alpha \nabla f(x^*))\|^2] \\ &= \|x^k - x^*\|^2 - 2\alpha \mathbb{E} [(g^k - \nabla f(x^*))^\top (x^k - x^*)] \\ &\quad + \alpha^2 \mathbb{E} [\|g^k - \nabla f(x^*)\|^2] \\ &\leq \|x^k - x^*\|^2 - 2\alpha (\nabla f(x^k) - \nabla f(x^*))^\top (x^k - x^*) \\ &\quad + 2\alpha^2 \|\nabla f(x^k) - \nabla f(x^*)\|_{\mathbf{C}}^2 + 2\alpha^2 \|h^k - \nabla f(x^*)\|_{\mathbf{C} - \mathbf{I}}^2 \\ &\leq \|x^k - x^*\|^2 - \alpha\mu \|x^k - x^*\|^2 - 2\alpha D_f(x^k, x^*) \\ &\quad + 2\alpha^2 \|\nabla f(x^k) - \nabla f(x^*)\|_{\mathbf{C}}^2 + 2\alpha^2 \|h^k - \nabla f(x^*)\|_{\mathbf{C} - \mathbf{I}}^2. \end{aligned}$$

Using Assumption 3.4.1 we get

$$-2\alpha D_f(x^k, x^*) \leq -\alpha \|\nabla f(x^k) - \nabla f(x^*)\|_{\mathbf{Q}}^2.$$

As for the second term in Φ^{k+1} , we have by Lemma C.1.3

$$\alpha\sigma \mathbb{E} [\|h^{k+1} - \nabla f(x^*)\|^2] = \alpha\sigma \|h^k - \nabla f(x^*)\|_{\mathbf{I} - \mathbb{E}[\mathbf{Z}]}^2 + \alpha\sigma \|\nabla f(x^k) - \nabla f(x^*)\|_{\mathbb{E}[\mathbf{Z}]}^2.$$

Combining it into Lyapunov function Φ^k ,

$$\begin{aligned}\Phi^{k+1} \leq & (1 - \alpha\mu)\|x^k - x^*\|^2 + \alpha\sigma\|h^k - \nabla f(x^*)\|_{\mathbf{I} - \mathbb{E}[\mathbf{Z}]}^2 + 2\alpha^2\|h^k - \nabla f(x^*)\|_{\mathbf{C} - \mathbf{I}}^2 \\ & + \alpha\sigma\|\nabla f(x^k) - \nabla f(x^*)\|_{\mathbb{E}[\mathbf{Z}]}^2 + 2\alpha^2\|\nabla f(x^k) - \nabla f(x^*)\|_{\mathbf{C}}^2 - \alpha\|\nabla f(x^k) \\ & - \nabla f(x^*)\|_{\mathbf{Q}}^2.\end{aligned}$$

To see that this gives us the theorem's statement, consider first

$$\alpha\sigma\mathbb{E}[\mathbf{Z}] + 2\alpha^2\mathbf{C} - \alpha\mathbf{Q} = 2\alpha\left(\alpha\mathbf{C} - \frac{1}{2}(\mathbf{Q} - \sigma\mathbb{E}[\mathbf{Z}])\right) \leq 0,$$

so we can drop norms related to $\nabla f(x^k) - \nabla f(x^*)$. Next, we have

$$\begin{aligned}\alpha\sigma(\mathbf{I} - \mathbb{E}[\mathbf{Z}]) + 2\alpha^2(\mathbf{C} - \mathbf{I}) &= \alpha(\alpha(2(\mathbf{C} - \mathbf{I}) + \sigma\mu\mathbf{I}) - \mathbb{E}[\mathbf{Z}]) + \sigma\alpha(1 - \alpha\mu)\mathbf{I} \\ &\leq \sigma\alpha(1 - \alpha\mu)\mathbf{I},\end{aligned}$$

which follows from our assumption on α .

C.1.2 Proof of Lemma C.1.3

Proof. Keeping in mind that $\mathbf{Z}_k^\top = \mathbf{Z}_k$, we first write

$$\begin{aligned}\mathbb{E}[\|h^{k+1} - v\|^2] &\stackrel{(3.8)}{=} \mathbb{E}\left[\|h^k + \mathbf{Z}_k(\nabla f(x^k) - h^k) - v\|^2\right] \\ &= \mathbb{E}\left[\|(\mathbf{I} - \mathbf{Z}_k)(h^k - v) + \mathbf{Z}_k(\nabla f(x^k) - v)\|^2\right] \\ &= \mathbb{E}\left[\|(\mathbf{I} - \mathbf{Z}_k)(h^k - v)\|^2\right] + \mathbb{E}\left[\|\mathbf{Z}_k(\nabla f(x^k) - v)\|^2\right] \\ &\quad + 2(h^k - v)^\top \mathbb{E}\left[(\mathbf{I} - \mathbf{Z}_k)^\top \mathbf{Z}_k\right](\nabla f(x^k) - v) \\ &= (h^k - v)^\top \mathbb{E}\left[(\mathbf{I} - \mathbf{Z}_k)^\top (\mathbf{I} - \mathbf{Z}_k)\right](h^k - v) \\ &\quad + (\nabla f(x^k) - v)^\top \mathbb{E}[\mathbf{Z}_k \mathbf{Z}_k](\nabla f(x^k) - v) \\ &\quad + 2(h^k - v)^\top \mathbb{E}[\mathbf{Z}_k - \mathbf{Z}_k \mathbf{Z}_k](\nabla f(x^k) - v).\end{aligned}$$

By Lemma C.1.2 we have $\mathbf{Z}_k \mathbf{Z}_k = \mathbf{Z}_k$, so the last term in the expression above is equal to 0. As for the other two, expanding the matrix factor in the first term leads to

$$\begin{aligned}\mathbb{E}\left[(\mathbf{I} - \mathbf{Z}_k)^\top (\mathbf{I} - \mathbf{Z}_k)\right] &= \mathbb{E}[(\mathbf{I} - \mathbf{Z}_k)(\mathbf{I} - \mathbf{Z}_k)] \\ &= \mathbb{E}[\mathbf{I} - \mathbf{Z}_k \mathbf{I} - \mathbf{I} \mathbf{Z}_k + \mathbf{Z}_k \mathbf{Z}_k] \\ &= \mathbf{I} - \mathbb{E}[\mathbf{Z}_k].\end{aligned}$$

We, thereby, have derived

$$\begin{aligned}\mathbb{E} [\|h^{k+1} - v\|^2] &= (h^k - v)^\top (\mathbf{I} - \mathbb{E} [\mathbf{Z}_k]) (h^k - v) \\ &\quad + (\nabla f(x^k) - v)^\top \mathbb{E} [\mathbf{Z}_k \mathbf{Z}_k] (\nabla f(x^k) - v) \\ &= \|h^k - v\|_{\mathbf{I} - \mathbb{E} [\mathbf{Z}]}^2 + \|\nabla f(x^k) - v\|_{\mathbb{E} [\mathbf{Z}]}^2.\end{aligned}$$

□

C.1.3 Proof of Lemma C.1.4

Proof. Throughout this proof, we will use without any mention that $\mathbf{Z}_k^\top = \mathbf{Z}_k$.

Writing $g^k - v = a + b$, where $a \stackrel{\text{def}}{=} (\mathbf{I} - \theta_k \mathbf{Z}_k)(h^k - v)$ and $b \stackrel{\text{def}}{=} \theta_k \mathbf{Z}_k(\nabla f(x^k) - v)$, we get $\|g^k\|^2 \leq 2(\|a\|^2 + \|b\|^2)$. Using Lemma C.1.2 and the definition of θ_k yields

$$\begin{aligned}\mathbb{E} [\|a\|^2] &= \mathbb{E} [\|(\mathbf{I} - \theta_k \mathbf{Z}_k)(h^k - v)\|^2] \\ &= (h^k - v)^\top \mathbb{E} [(\mathbf{I} - \theta_k \mathbf{Z}_k)(\mathbf{I} - \theta_k \mathbf{Z}_k)] (h^k - v) \\ &= (h^k - v)^\top \mathbb{E} [(\mathbf{I} - \theta_k \mathbf{Z}_k - \theta_k \mathbf{Z}_k + \theta_k^2 \mathbf{Z}_k \mathbf{I} \mathbf{Z}_k)] (h^k - v) \\ &= (h^k - v)^\top \mathbb{E} [(\mathbf{I} - 2\mathbf{I} + \theta_k^2 \mathbf{Z}_k)] (h^k - v) \\ &= \|h^k - v\|_{\mathbb{E} [\theta^2 \mathbf{Z}] - \mathbf{I}}^2.\end{aligned}$$

Similarly, the second term in the upper bound on g^k can be rewritten as

$$\begin{aligned}\mathbb{E} [\|b\|^2] &= \mathbb{E} [\|\theta_k \mathbf{Z}_k(\nabla f(x^k) - v)\|^2] \\ &= (\nabla f(x^k) - v)^\top \mathbb{E} [\theta_k^2 \mathbf{Z}_k \mathbf{Z}_k] (\nabla f(x^k) - v) \\ &= \|\nabla f(x^k) - v\|_{\mathbf{C}}^2.\end{aligned}$$

Combining the pieces, we get the claim. □

C.2 Proofs for Section 3.5

C.2.1 Technical lemmas

We first start with an analogue of Lemma C.1.4 allowing for a norm different from $\|\cdot\|$. We remark that matrix \mathbf{Q}' in the lemma is not to be confused with the smoothness matrix \mathbf{Q} from Assumption 3.4.1.

Lemma C.2.1. *Let $\mathbf{Q}' \succ 0$. The variance of g^k as an estimator of $\nabla f(x^k)$ can be bounded as follows:*

$$\frac{1}{2} \mathbb{E} [\|g^k\|_{\mathbf{Q}'}^2] \leq \|h^k\|_{\hat{\mathbf{P}}^{-1}(\mathbf{P} \circ \mathbf{Q}') \hat{\mathbf{P}}^{-1} - \mathbf{Q}'}^2 + \|\nabla f(x^k)\|_{\hat{\mathbf{P}}^{-1}(\mathbf{P} \circ \mathbf{Q}') \hat{\mathbf{P}}^{-1}}^2. \quad (\text{C.3})$$

Proof. Denote \mathbf{S}_k to be a matrix with columns e_i for $i \in \text{Range}(\mathbf{S}_k)$. We first write

$$g^k = \underbrace{h^k - \hat{\mathbf{P}}^{-1} \mathbf{S}_k \mathbf{S}_k^\top h^k}_a + \underbrace{\hat{\mathbf{P}}^{-1} \mathbf{S}_k \mathbf{S}_k^\top \nabla f(x^k)}_b.$$

Let us bound the expectation of each term individually. The first term is equal to

$$\begin{aligned}
\mathbb{E} [\|a\|_{\mathbf{Q}'}^2] &= \mathbb{E} \left[\left\| \left(\mathbf{I} - \hat{\mathbf{P}}^{-1} \mathbf{S}_k \mathbf{S}_k^\top \right) h^k \right\|_{\mathbf{Q}'}^2 \right] \\
&= (h^k)^\top \mathbb{E} \left[\left(\mathbf{I} - \hat{\mathbf{P}}^{-1} \mathbf{S}_k \mathbf{S}_k^\top \right)^\top \mathbf{Q}' \left(\mathbf{I} - \hat{\mathbf{P}}^{-1} \mathbf{S}_k \mathbf{S}_k^\top \right) \right] h^k \\
&= (h^k)^\top \mathbb{E} \left[\left(\mathbf{Q}' - \hat{\mathbf{P}}^{-1} \mathbf{S}_k \mathbf{S}_k^\top \mathbf{Q}' - \mathbf{Q}' \mathbf{S}_k \mathbf{S}_k^\top \hat{\mathbf{P}}^{-1} \right) \right] h^k \\
&\quad + (h^k)^\top \mathbb{E} \left[\left(\hat{\mathbf{P}}^{-1} \mathbf{S}_k \mathbf{S}_k^\top \mathbf{Q}' \mathbf{S}_k \mathbf{S}_k^\top \hat{\mathbf{P}}^{-1} \right) \right] h^k \\
&= (h^k)^\top \left(\hat{\mathbf{P}}^{-1} (\mathbf{P} \circ \mathbf{Q}') \hat{\mathbf{P}}^{-1} - \mathbf{Q}' \right) h^k.
\end{aligned}$$

The second term can be bounded as

$$\begin{aligned}
\mathbb{E} [\|b\|_{\mathbf{Q}'}^2] &= \mathbb{E} \left[\left\| \hat{\mathbf{P}}^{-1} \mathbf{S}_k^\top \nabla f(x^k) \mathbf{S}_k \right\|_{\mathbf{Q}'}^2 \right] = \mathbb{E} \left[\|\nabla f(x^k)\|_{\hat{\mathbf{P}}^{-1} \mathbf{S}_k \mathbf{S}_k^\top \mathbf{Q}' \mathbf{S}_k \mathbf{S}_k^\top \hat{\mathbf{P}}^{-1}}^2 \right] \\
&= \|\nabla f(x^k)\|_{\hat{\mathbf{P}}^{-1} (\mathbf{P} \circ \mathbf{Q}') \hat{\mathbf{P}}^{-1}}^2.
\end{aligned}$$

It remains to combine the two bounds. □

We also state the analogue of Lemma C.1.3, which allows for a different norm as well.

Lemma C.2.2. *For all diagonal $\mathbf{D} \succ 0$ we have*

$$\mathbb{E} [\|h^{k+1}\|_{\mathbf{D}}^2] = \|h^k\|_{\mathbf{D} - \hat{\mathbf{P}}\mathbf{D}}^2 + \|\nabla f(x^k)\|_{\hat{\mathbf{P}}\mathbf{D}}^2. \quad (\text{C.4})$$

Proof. Denote \mathbf{S}_k to be a matrix with columns e_i for $i \in \mathbf{S}_k$. We first write

$$h^{k+1} = h^k - \mathbf{S}_k \mathbf{S}_k^\top h^k + \mathbf{S}_k \mathbf{S}_k^\top \nabla f(x^k).$$

Therefore

$$\begin{aligned}
\mathbb{E} [\|h^{k+1}\|_{\mathbf{D}}^2] &= \mathbb{E} \left[\left\| (\mathbf{I} - \mathbf{S}_k \mathbf{S}_k^\top) h^k + \mathbf{S}_k \mathbf{S}_k^\top \nabla f(x^k) \right\|_{\mathbf{D}}^2 \right] \\
&= \mathbb{E} \left[\left\| (\mathbf{I} - \mathbf{S}_k \mathbf{S}_k^\top) h^k \right\|_{\mathbf{D}}^2 \right] + \mathbb{E} \left[\left\| \mathbf{S}_k \mathbf{S}_k^\top \nabla f(x^k) \right\|_{\mathbf{D}}^2 \right] \\
&\quad + 2\mathbb{E} \left[h^{k\top} (\mathbf{I} - \mathbf{S}_k \mathbf{S}_k^\top) \mathbf{D} \mathbf{S}_k \mathbf{S}_k^\top \nabla f(x^k) \right] \\
&= \|h^k\|_{\mathbf{D} - \hat{\mathbf{P}}\mathbf{D}}^2 + \|\nabla f(x^k)\|_{\hat{\mathbf{P}}\mathbf{D}}^2.
\end{aligned}$$

□

C.2.2 Proof of Theorem 3.5.2

Proof. Throughout the proof, we will use the following Lyapunov function:

$$\Psi^k \stackrel{\text{def}}{=} f(x^k) - f(x^*) + \sigma \|h^k\|_{\mathbf{P}^{-1}}^2.$$

Following similar steps to what we did before, we obtain

$$\begin{aligned}
\mathbb{E} [\Psi^{k+1}] &\stackrel{(2.6)}{\leq} f(x^k) - f(x^*) + \alpha \mathbb{E} [\langle \nabla f(x^k), g^k \rangle] + \frac{\alpha^2}{2} \mathbb{E} [\|g^k\|_{\mathbf{M}}^2] + \sigma \mathbb{E} [\|h^{k+1}\|_{\hat{\mathbf{P}}^{-1}}^2] \\
&= f(x^k) - f(x^*) - \alpha \|\nabla f(x^k)\|_2^2 + \frac{\alpha^2}{2} \mathbb{E} [\|g^k\|_{\mathbf{M}}^2] + \sigma \mathbb{E} [\|h^{k+1}\|_{\hat{\mathbf{P}}^{-1}}^2] \\
&\stackrel{(C.3)}{\leq} f(x^k) - f(x^*) - \alpha \|\nabla f(x^k)\|_2^2 + \alpha^2 \|\nabla f(x^k)\|_{\hat{\mathbf{P}}^{-1}(\mathbf{P} \circ \mathbf{M})\hat{\mathbf{P}}^{-1}}^2 \\
&\quad + \alpha^2 \|h^k\|_{\hat{\mathbf{P}}^{-1}(\mathbf{P} \circ \mathbf{M})\hat{\mathbf{P}}^{-1} - \mathbf{M}}^2 + \sigma \mathbb{E} [\|h^{k+1}\|_{\hat{\mathbf{P}}^{-1}}^2].
\end{aligned}$$

This is the place where the ESO assumption comes into play. By applying it to the right-hand side of the bound above, we obtain

$$\begin{aligned}
\mathbb{E} [\Psi^{k+1}] &\stackrel{(3.13)}{\leq} f(x^k) - f(x^*) - \alpha \|\nabla f(x^k)\|_2^2 + \alpha^2 \|\nabla f(x^k)\|_{\hat{\mathbf{V}}\hat{\mathbf{P}}^{-1}}^2 + \alpha^2 \|h^k\|_{\hat{\mathbf{V}}\hat{\mathbf{P}}^{-1} - \mathbf{M}}^2 \\
&\quad + \sigma \mathbb{E} [\|h^{k+1}\|_{\hat{\mathbf{P}}^{-1}}^2] \\
&\stackrel{(C.4)}{=} f(x^k) - f(x^*) - \alpha \|\nabla f(x^k)\|_2^2 + \alpha^2 \|\nabla f(x^k)\|_{\hat{\mathbf{V}}\hat{\mathbf{P}}^{-1}}^2 + \alpha^2 \|h^k\|_{\hat{\mathbf{V}}\hat{\mathbf{P}}^{-1} - \mathbf{M}}^2 \\
&\quad + \sigma \|\nabla f(x^k)\|_2^2 + \sigma \|h^k\|_{\hat{\mathbf{P}}^{-1} - \mathbf{I}}^2 \\
&= f(x^k) - f(x^*) - \left(\alpha - \alpha^2 \max_i \frac{v_i}{p_i} - \sigma \right) \|\nabla f(x^k)\|_2^2 \\
&\quad + \|h^k\|_{\alpha^2(\hat{\mathbf{V}}\hat{\mathbf{P}}^{-1} - \mathbf{M}) + \sigma(\hat{\mathbf{P}}^{-1} - \mathbf{I})}^2.
\end{aligned}$$

Due to Polyak-Łojasiewicz inequality, we can further upper bound the last expression by

$$\left(1 - \left(\alpha - \alpha^2 \max_i \frac{v_i}{p_i} - \sigma \right) \mu \right) (f(x^k) - f(x^*)) + \|h^k\|_{\alpha^2(\hat{\mathbf{V}}\hat{\mathbf{P}}^{-1} - \mathbf{M}) + \sigma(\hat{\mathbf{P}}^{-1} - \mathbf{I})}^2.$$

To finish the proof, it remains to use (3.14). □

C.2.3 Proof of Corollary 3.5.3

The claim was obtained by choosing carefully α and σ using numerical grid search. Note that by strong convexity we have $\mathbf{I} \succeq \mu \mathbf{Diag}(\mathbf{M})^{-1}$, so we can satisfy assumption (3.14). Then, the claim follows immediately noticing that we can also set $\hat{\mathbf{V}} = \mathbf{Diag}(\mathbf{M})$ while maintaining

$$\left(\alpha - \alpha^2 \max_i \frac{\mathbf{M}_{ii}}{p_i} - \sigma \right) \geq \frac{0.117}{\mathbf{Tr}(\mathbf{M})}.$$

C.2.4 Accelerated SEGA with arbitrary sampling

Before establishing the main theorem, we first state two technical lemmas which will be crucial for the analysis. First one, Lemma C.2.3 provides a key inequality following from (6). The second one, Lemma C.2.4, analyzes update (5) and was technically established throughout the proof of Theorem 3.5.2. We include a proof of lemmas in Appendix C.2.5 and C.2.6 respectively.

Lemma C.2.3. For every $u \in \mathbb{R}^d$ we have

$$\begin{aligned} & \beta \langle \nabla f(x^{k+1}), z^k - u \rangle - \frac{\beta\mu}{2} \|x^{k+1} - u\|_2^2 \\ & \leq \beta^2 \frac{1}{2} \mathbb{E} [\|g^k\|_2^2] + \frac{1}{2} \|z^k - u\|_2^2 - \frac{1 + \beta\mu}{2} \mathbb{E} [\|z^{k+1} - u\|_2^2] \end{aligned} \quad (\text{C.5})$$

Lemma C.2.4. Letting $\eta(v, p) \stackrel{\text{def}}{=} \max_i \frac{\sqrt{v_i}}{p_i}$, we have

$$f(x^{k+1}) - \mathbb{E} [f(y^{k+1})] + \|h^k\|_{\alpha^2(\hat{\mathbf{V}}\hat{\mathbf{P}}^{-3}-\hat{\mathbf{P}}^{-1}\mathbf{M}\hat{\mathbf{P}}^{-1})}^2 \geq (\alpha - \alpha^2\eta(v, p)^2) \|\nabla f(x^k)\|_{\hat{\mathbf{P}}^{-1}}^2 \quad (\text{C.6})$$

Now we state the main theorem of Section 3.5.3, providing a convergence rate of ASEGA (Algorithm 7) for arbitrary minibatch sampling. As we mentioned, the convergence rate is, up to a constant factor, same as state-of-the-art minibatch accelerated coordinate descent [78].

Theorem C.2.5. Assume \mathbf{M} -smoothness and μ -strong convexity and that v satisfies (3.13). Denote

$$\Upsilon^k \stackrel{\text{def}}{=} \frac{2}{75} \frac{\eta(v, p)^{-2}}{\tau^2} (\mathbb{E} [f(y^k)] - f(x^*)) + \frac{1 + \beta\mu}{2} \mathbb{E} [\|z^k - x^*\|_2^2] + \sigma \mathbb{E} [\|h^k\|_{\hat{\mathbf{P}}^{-2}}^2]$$

and choose

$$c_1 = \max \left(1, \eta(v, p)^{-1} \frac{\sqrt{\mu}}{\min_i p_i} \right), \quad (\text{C.7})$$

$$\alpha = \frac{1}{5\eta(v, p)^2}, \quad (\text{C.8})$$

$$\beta = \frac{2}{75\tau\eta(v, p)^2}, \quad (\text{C.9})$$

$$\sigma = 5\beta^2, \quad (\text{C.10})$$

$$\tau = \frac{\sqrt{\frac{4}{9.5^4} \eta(v, p)^{-4} \mu^2 + \frac{8}{75} \eta(v, p)^{-2} \mu - \frac{2}{75} \eta(v, p)^{-2} \mu}}{2}. \quad (\text{C.11})$$

Then, we have

$$\mathbb{E} [\Upsilon^k] \leq (1 - c_1^{-1} \tau)^k \Upsilon^0.$$

Proof. The proof technique is inspired by Allen-Zhu and Orecchia [6]. First of all, let us see what strong convexity of f gives us:

$$\beta (f(x^{k+1}) - f(x^*)) \leq \beta \langle \nabla f(x^{k+1}), x^{k+1} - x^* \rangle - \frac{\beta\mu}{2} \|x^* - x^{k+1}\|_2^2.$$

Thus, we are interested in finding an upper bound for the scalar product that appeared

above. We have

$$\begin{aligned} & \beta \langle \nabla f(x^{k+1}), z^k - u \rangle - \frac{\beta\mu}{2} \|x^{k+1} - u\|_2^2 + \sigma \mathbb{E} [\|h^{k+1}\|_{\hat{\mathbf{P}}^{-2}}^2] \\ & \stackrel{(C.5)}{\leq} \beta^2 \frac{1}{2} \mathbb{E} [\|g^k\|_2^2] + \frac{1}{2} \|z^k - u\|_2^2 - \frac{1 + \beta\mu}{2} \mathbb{E} [\|z^{k+1} - u\|_2^2] + \sigma \mathbb{E} [\|h^{k+1}\|_{\hat{\mathbf{P}}^{-2}}^2]. \end{aligned}$$

Using the Lemmas introduced above, we can upper bound the norms of g^k and h^{k+1} by using norms of h^k and $\nabla f(x^k)$ to get the following:

$$\begin{aligned} & \beta^2 \frac{1}{2} \mathbb{E} [\|g^k\|_2^2] + \sigma \mathbb{E} [\|h^{k+1}\|_{\hat{\mathbf{P}}^{-2}}^2] \\ & \stackrel{(C.4)}{\leq} \beta^2 \frac{1}{2} \mathbb{E} [\|g^k\|_2^2] + \sigma \|h^k\|_{\hat{\mathbf{P}}^{-2} - \hat{\mathbf{P}}^{-1}}^2 + \sigma \|\nabla f(x^k)\|_{\hat{\mathbf{P}}^{-1}}^2 \\ & \stackrel{(C.3)}{\leq} \beta^2 \|h^k\|_{\hat{\mathbf{P}}^{-1} - \mathbf{I}}^2 + \beta^2 \|\nabla f(x^k)\|_{\hat{\mathbf{P}}^{-1}}^2 + \sigma \|h^k\|_{\hat{\mathbf{P}}^{-2} - \hat{\mathbf{P}}^{-1}}^2 + \sigma \|\nabla f(x^k)\|_{\hat{\mathbf{P}}^{-1}}^2. \end{aligned}$$

Now, let us get rid of $\nabla f(x^k)$ by using the gradients property from Lemma C.2.4:

$$\begin{aligned} & \beta^2 \frac{1}{2} \mathbb{E} [\|g^k\|_2^2] + \sigma \mathbb{E} [\|h^{k+1}\|_{\hat{\mathbf{P}}^{-2}}^2] \\ & \stackrel{(C.6)}{\leq} \beta^2 \|h^k\|_{\hat{\mathbf{P}}^{-1} - \mathbf{I}}^2 + (\beta^2 + \sigma) \frac{f(x^{k+1}) - f(y^{k+1}) + \|h^k\|_{\alpha^2(\hat{\mathbf{V}}\hat{\mathbf{P}}^{-3} - \hat{\mathbf{P}}^{-1}\mathbf{M}\hat{\mathbf{P}}^{-1})}^2}{\alpha - \alpha^2\eta(v, p)^2} \\ & \quad + \sigma \|h^k\|_{\hat{\mathbf{P}}^{-2} - \hat{\mathbf{P}}^{-1}}^2 \\ & = \|h^k\|_{\beta^2(\hat{\mathbf{P}}^{-1} - \mathbf{I}) + \frac{(\beta^2 + \sigma)\alpha^2}{\alpha - \alpha^2\eta(v, p)^2}(\hat{\mathbf{V}}\hat{\mathbf{P}}^{-3} - \hat{\mathbf{P}}^{-1}\mathbf{M}\hat{\mathbf{P}}^{-1}) + \sigma(\hat{\mathbf{P}}^{-2} - \hat{\mathbf{P}}^{-1})}^2 \\ & \quad + \frac{\beta^2 + \sigma}{\alpha - \alpha^2\eta(v, p)^2} (f(x^{k+1}) - \mathbb{E} [f(y^{k+1})]) \\ & \leq \|h^k\|_{\beta^2\hat{\mathbf{P}}^{-1} + \frac{(\beta^2 + \sigma)\alpha^2}{\alpha - \alpha^2\eta(v, p)^2}\hat{\mathbf{V}}\hat{\mathbf{P}}^{-3} + \sigma(\hat{\mathbf{P}}^{-2} - \hat{\mathbf{P}}^{-1})}^2 \\ & \quad + \frac{\beta^2 + \sigma}{\alpha - \alpha^2\eta(v, p)^2} (f(x^{k+1}) - \mathbb{E} [f(y^{k+1})]). \end{aligned}$$

Plugging this into the bound with which we started the proof, we deduce

$$\begin{aligned} & \beta \langle \nabla f(x^{k+1}), z^k - u \rangle - \frac{\beta\mu}{2} \|x^{k+1} - u\|_2^2 + \sigma \mathbb{E} [\|h^{k+1}\|_{\hat{\mathbf{P}}^{-2}}^2] \\ & \leq \|h^k\|_{\beta^2\hat{\mathbf{P}}^{-1} + \frac{(\beta^2 + \sigma)\alpha^2}{\alpha - \alpha^2\eta(v, p)^2}\hat{\mathbf{V}}\hat{\mathbf{P}}^{-3} + \sigma(\hat{\mathbf{P}}^{-2} - \hat{\mathbf{P}}^{-1})}^2 \\ & \quad + \frac{\beta^2 + \sigma}{\alpha - \alpha^2\eta(v, p)^2} (f(x^{k+1}) - \mathbb{E} [f(y^{k+1})]) + \frac{1}{2} \|z^k - u\|_2^2 \\ & \quad - \frac{1 + \beta\mu}{2} \mathbb{E} [\|z^{k+1} - u\|_2^2]. \end{aligned}$$

Recalling our first step, we get with a few rearrangements

$$\begin{aligned}
& \beta (f(x^{k+1}) - f(x^*)) \\
& \leq \beta \langle \nabla f(x^{k+1}), x^{k+1} - x^* \rangle - \frac{\beta\mu}{2} \|x^* - x^{k+1}\|_2^2 \\
& = \beta \langle \nabla f(x^{k+1}), x^{k+1} - z^k \rangle + \beta \langle \nabla f(x^{k+1}), z^k - x^* \rangle - \frac{\beta\mu}{2} \|x^* - x^{k+1}\|_2^2 \\
& = \frac{(1-\tau)\beta}{\tau} \langle \nabla f(x^{k+1}), y^k - x^{k+1} \rangle + \beta \langle \nabla f(x^{k+1}), z^k - x^* \rangle - \frac{\beta\mu}{2} \|x^* - x^{k+1}\|_2^2 \\
& \leq \frac{(1-\tau)\beta}{\tau} (f(y^k) - f(x^{k+1})) + \|h^k\|_{\beta^2 \hat{\mathbf{P}}^{-1} + \frac{(\beta^2 + \sigma)\alpha^2}{\alpha - \alpha^2 \eta(v,p)^2} \hat{\mathbf{V}} \hat{\mathbf{P}}^{-3 + \sigma} (\hat{\mathbf{P}}^{-2} - \hat{\mathbf{P}}^{-1})}^2 \\
& \quad + \frac{\beta^2 + \sigma}{\alpha - \alpha^2 \eta(v,p)^2} (f(x^{k+1}) - \mathbb{E}[f(y^{k+1})]) + \frac{1}{2} \|z^k - x^*\|_2^2 \\
& \quad - \frac{1 + \beta\mu}{2} \mathbb{E}[\|z^{k+1} - x^*\|_2^2] - \sigma \mathbb{E}[\|h^{k+1}\|_{\hat{\mathbf{P}}^{-2}}^2].
\end{aligned}$$

Let us choose σ, β such that for some constant c_2 (which we choose at the end) we have

$$c_2 \sigma = \beta^2, \quad \beta = \frac{\alpha - \alpha^2 \eta(v,p)^2}{(1 + c_2^{-1})\tau}.$$

Consequently, we have

$$\begin{aligned}
& \frac{\alpha - \alpha^2 \eta(v,p)^2}{(1 + c_2^{-1})\tau^2} (\mathbb{E}[f(y^{k+1})] - f(x^*)) + \frac{1 + \beta\mu}{2} \mathbb{E}[\|z^{k+1} - x^*\|_2^2] + \sigma \mathbb{E}[\|h^{k+1}\|_{\hat{\mathbf{P}}^{-2}}^2] \\
& \leq (1 - \tau) \frac{\alpha - \alpha^2 \eta(v,p)^2}{(1 + c_2^{-1})\tau^2} (f(y^k) - f(x^*)) + \frac{1}{2} \|z^k - x^*\|_2^2 \\
& \quad + \|h^k\|_{\left(\hat{\mathbf{P}}^{-1} - (1 - c_2)\mathbf{I} + \frac{(1 + c_2)\alpha^2}{\alpha - \alpha^2 \eta(v,p)^2} \hat{\mathbf{V}} \hat{\mathbf{P}}^{-2}\right) \sigma \hat{\mathbf{P}}^{-1}}^2.
\end{aligned}$$

Let us make a particular choice of α , so that for some constant c_3 (which we choose at the end) we can obtain the equations below:

$$\begin{aligned}
\alpha = \frac{1}{c_3 \eta(v,p)^2} \quad \Rightarrow \quad \alpha - \alpha^2 \eta(v,p)^2 &= \frac{c_3 - 1}{c_3^2} \eta(v,p)^{-2}, \\
\frac{\alpha^2}{\alpha - \alpha^2 \eta(v,p)^2} &= \frac{1}{(c_3 - 1) \eta(v,p)^2}.
\end{aligned}$$

Thus

$$\begin{aligned}
& \frac{\frac{c_3-1}{c_2^2}\eta(v,p)^{-2}}{(1+c_2^{-1})\tau^2} (\mathbb{E}[f(y^{k+1})] - f(x^*)) + \frac{1+\beta\mu}{2}\mathbb{E}[\|z^{k+1} - x^*\|_2^2] + \sigma\mathbb{E}[\|h^{k+1}\|_{\hat{\mathbf{P}}^{-2}}^2] \\
& \leq (1-\tau)\frac{\frac{c_3-1}{c_2^2}\eta(v,p)^{-2}}{(1+c_2^{-1})\tau^2} (f(y^k) - f(x^*)) + \frac{1}{2}\|z^k - x^*\|_2^2 \\
& \quad + \|h^k\|_{\left(\hat{\mathbf{P}}^{-1} - (1-c_2)\mathbf{I} + \frac{(1+c_2)}{(c_3-1)\eta(v,p)^2}\hat{\mathbf{V}}\hat{\mathbf{P}}^{-2}\right)\sigma\hat{\mathbf{P}}^{-1}}^2.
\end{aligned}$$

Using the definition of $\eta(v, p)$, one can see that the above gives

$$\begin{aligned}
& \frac{\frac{c_3-1}{c_2^2}\eta(v,p)^{-2}}{(1+c_2^{-1})\tau^2} (\mathbb{E}[f(y^{k+1})] - f(x^*)) + \frac{1+\beta\mu}{2}\mathbb{E}[\|z^{k+1} - x^*\|_2^2] + \sigma\mathbb{E}[\|h^{k+1}\|_{\hat{\mathbf{P}}^{-2}}^2] \\
& \leq (1-\tau)\frac{\frac{c_3-1}{c_2^2}\eta(v,p)^{-2}}{(1+c_2^{-1})\tau^2} (f(y^k) - f(x^*)) + \frac{1}{2}\|z^k - x^*\|_2^2 \\
& \quad + \|h^k\|_{\left(\hat{\mathbf{P}}^{-1} - (1-c_2)\mathbf{I} + \frac{1+c_2}{c_3-1}\mathbf{I}\right)\sigma\hat{\mathbf{P}}^{-1}}^2.
\end{aligned}$$

To get the convergence rate, we shall establish

$$\left(1 - c_2 - \frac{1+c_2}{c_3-1}\right)c_1\mathbf{I} \succeq \tau\hat{\mathbf{P}}^{-1} \quad (\text{C.12})$$

and

$$1 + \beta\mu \geq \frac{1}{1-\tau}. \quad (\text{C.13})$$

To this end, let us recall that

$$\beta = \frac{c_3-1}{c_2^2}\eta(v,p)^{-2}\tau^{-1}\frac{1}{1+c_2^{-1}}.$$

Now we would like to set equality in (C.13), which yields

$$0 = \tau^2 + \frac{c_3-1}{c_2^2}\eta(v,p)^{-2}\frac{1}{1+c_2^{-1}}\mu\tau - \frac{c_3-1}{c_2^2}\eta(v,p)^{-2}\frac{1}{1+c_2^{-1}}\mu = 0.$$

This, in turn, implies

$$\begin{aligned}
\tau &= \frac{\sqrt{\left(\frac{c_3-1}{c_2^2}\right)^2\eta(v,p)^{-4}\frac{1}{(1+c_2^{-1})^2}\mu^2 + 4\frac{c_3-1}{c_2^2}\eta(v,p)^{-2}\frac{1}{1+c_2^{-1}}\mu} - \frac{c_3-1}{c_2^2}\eta(v,p)^{-2}\frac{1}{1+c_2^{-1}}\mu}{2} \\
&= \mathcal{O}\left(\sqrt{\frac{c_3-1}{c_2^2}\frac{1}{\sqrt{1+c_2^{-1}}}\eta(v,p)^{-1}\sqrt{\mu}}\right).
\end{aligned}$$

Notice that for any $c \leq 1$ we have $\frac{\sqrt{c^2+4c}-c}{2} \leq \sqrt{c}$ and therefore

$$\tau \leq \sqrt{\frac{c_3 - 1}{c_2^2}} \eta(v, p)^{-1} \frac{1}{\sqrt{1 + c_2^{-1}}} \sqrt{\mu}. \quad (\text{C.14})$$

Using this inequality and a particular choice of constants, we can upper bound \mathbf{P}^{-1} by a matrix proportional to identity as shown below:

$$\begin{aligned} \tau \hat{\mathbf{P}}^{-1} &\stackrel{(\text{C.14})}{\preceq} \sqrt{\frac{c_3 - 1}{c_2^2}} \eta(v, p)^{-1} \frac{1}{\sqrt{1 + c_2^{-1}}} \sqrt{\mu} \hat{\mathbf{P}}^{-1} \\ &\preceq \sqrt{\frac{c_3 - 1}{c_2^2}} \eta(v, p)^{-1} \frac{1}{\sqrt{1 + c_2^{-1}}} \frac{\sqrt{\mu}}{\min_i p_i} \mathbf{I} \\ &\stackrel{(\text{C.7})}{\preceq} \sqrt{\frac{c_3 - 1}{c_2^2}} \frac{1}{\sqrt{1 + c_2^{-1}}} c_1 \mathbf{I} \\ &\stackrel{(*)}{\preceq} \left(1 - c_2 - \frac{1 + c_2}{c_3 - 1} \right) c_1 \mathbf{I}, \end{aligned}$$

which is exactly (C.12). Above, $(*)$ holds for choice $c_3 = 5$ and $c_2 = \frac{1}{5}$. It remains to verify that (C.8), (C.9), (C.10) and (C.11) indeed correspond to our derivations. \square

We also mention, without a proof, that acceleration parameters can be chosen in general such that c_1 can be lower bounded by constant and therefore the rate from Theorem C.2.5 coincides with the rate from Table 3.1. Corollary 3.5.4 is in fact a weaker result of that type.

Proof of Corollary 3.5.4

It suffices to verify that one can choose $v = \text{Diag}(\mathbf{M})$ in (3.13) and that due to $p_i \propto \sqrt{\mathbf{M}_{ii}}$ we have $c_1 = 1$.

C.2.5 Proof of Lemma C.2.3

Proof. Firstly (6), is equivalent to

$$z^{k+1} = \arg \min_z \psi^k(z) \stackrel{\text{def}}{=} \frac{1}{2} \|z - z^k\|_2^2 + \beta \langle g^k, z \rangle + \frac{\beta \mu}{2} \|z - x^{k+1}\|_2^2.$$

Therefore, we have for every u

$$\begin{aligned} 0 &= \langle \nabla \psi^k(z^{k+1}), z^{k+1} - u \rangle \\ &= \langle z^{k+1} - z^k, z^{k+1} - u \rangle + \beta \langle g^k, z^{k+1} - u \rangle + \beta \mu \langle z^{k+1} - x^{k+1}, z^{k+1} - u \rangle. \end{aligned} \quad (\text{C.15})$$

Next, by generalized Pythagorean theorem we have

$$\langle z^{k+1} - z^k, z^{k+1} - u \rangle = \frac{1}{2} \|z^k - z^{k+1}\|_2^2 - \frac{1}{2} \|z^k - u\|_2^2 + \frac{1}{2} \|u - z^{k+1}\|_2^2 \quad (\text{C.16})$$

and

$$\langle z^{k+1} - x^{k+1}, z^{k+1} - u \rangle = \frac{1}{2} \|x^{k+1} - z^{k+1}\|_2^2 - \frac{1}{2} \|x^{k+1} - u\|_2^2 + \frac{1}{2} \|u - z^{k+1}\|_2^2. \quad (\text{C.17})$$

Plugging (C.16) and (C.17) into (C.15) we obtain

$$\begin{aligned} \beta \langle g^k, z^k - u \rangle - \frac{\beta\mu}{2} \|x^{k+1} - u\|_2^2 \\ \leq \beta \langle g^k, z^k - z^{k+1} \rangle - \frac{1}{2} \|z^k - z^{k+1}\|_2^2 + \frac{1}{2} \|z^k - u\|_2^2 - \frac{1 + \beta\mu}{2} \|z^{k+1} - u\|_2^2 \\ \stackrel{(*)}{\leq} \frac{\beta^2}{2} \|g^k\|_2^2 + \frac{1}{2} \|z^k - u\|_2^2 - \frac{1 + \beta\mu}{2} \|z^{k+1} - u\|_2^2. \end{aligned}$$

The step marked by (*) holds due to Cauchy-Schwartz inequality. It remains to take the expectation conditioned on x^{k+1} and use (3.7). □

C.2.6 Proof of Lemma C.2.4

Proof. The shortest, although not the most intuitive, way to write the proof is to put matrix factor into norms. Apart from this trick, the proof is quite simple consists of applying smoothness followed by ESO:

$$\begin{aligned} \mathbb{E} [f(y^{k+1})] - f(x^{k+1}) &\stackrel{(2.6)}{\leq} -\alpha \mathbb{E} [\langle \nabla f(x^k), \hat{\mathbf{P}}^{-1} g^k \rangle] + \frac{\alpha^2}{2} \mathbb{E} [\|\hat{\mathbf{P}}^{-1} g^k\|_{\mathbf{M}}^2] \\ &= -\alpha \|\nabla f(x^k)\|_{\hat{\mathbf{P}}^{-1}}^2 + \frac{\alpha^2}{2} \mathbb{E} [\|g^k\|_{\hat{\mathbf{P}}^{-1} \mathbf{M} \hat{\mathbf{P}}^{-1}}^2] \\ &\stackrel{(\text{C.3})}{\leq} -\alpha \|\nabla f(x^k)\|_{\hat{\mathbf{P}}^{-1}}^2 + \alpha^2 \|\nabla f(x^k)\|_{\hat{\mathbf{P}}^{-1} (\mathbf{P} \circ \hat{\mathbf{P}}^{-1} \mathbf{M} \hat{\mathbf{P}}^{-1}) \hat{\mathbf{P}}^{-1}}^2 \\ &\quad + \alpha^2 \|h^k\|_{\hat{\mathbf{P}}^{-1} (\mathbf{P} \circ \hat{\mathbf{P}}^{-1} \mathbf{M} \hat{\mathbf{P}}^{-1}) \hat{\mathbf{P}}^{-1} - \hat{\mathbf{P}}^{-1} \mathbf{M} \hat{\mathbf{P}}^{-1}}^2 \\ &= -\alpha \|\nabla f(x^k)\|_{\hat{\mathbf{P}}^{-1}}^2 + \alpha^2 \|\nabla f(x^k)\|_{\hat{\mathbf{P}}^{-2} (\mathbf{P} \circ \mathbf{M}) \hat{\mathbf{P}}^{-2}}^2 \\ &\quad + \alpha^2 \|h^k\|_{\hat{\mathbf{P}}^{-2} (\mathbf{P} \circ \mathbf{M}) \hat{\mathbf{P}}^{-2} - \hat{\mathbf{P}}^{-1} \mathbf{M} \hat{\mathbf{P}}^{-1}}^2 \\ &\stackrel{(3.13)}{\leq} -\alpha \|\nabla f(x^k)\|_{\hat{\mathbf{P}}^{-1}}^2 + \alpha^2 \|\nabla f(x^k)\|_{\hat{\mathbf{V}} \hat{\mathbf{P}}^{-3}}^2 \\ &\quad + \alpha^2 \|h^k\|_{\hat{\mathbf{V}} \hat{\mathbf{P}}^{-3} - \hat{\mathbf{P}}^{-1} \mathbf{M} \hat{\mathbf{P}}^{-1}}^2 \\ &\leq -\left(\alpha - \alpha^2 \max_i \frac{v_i}{p_i^2}\right) \|\nabla f(x^k)\|_{\hat{\mathbf{P}}^{-1}}^2 + \alpha^2 \|h^k\|_{\hat{\mathbf{V}} \hat{\mathbf{P}}^{-3} - \hat{\mathbf{P}}^{-1} \mathbf{M} \hat{\mathbf{P}}^{-1}}^2. \end{aligned}$$

□

C.3 Subspace SEGA: a more aggressive approach

In this section we describe a *more aggressive* variant of SEGA, one that exploits the fact that the gradients of f lie in a lower dimensional subspace if this is indeed the case.

In particular, assume that $F(x) = f(x) + \psi(x)$ and

$$f(x) = \phi(\mathbf{A}x),$$

where $\mathbf{A} \in \mathbb{R}^{m \times d}$.¹ Note that $\nabla f(x)$ lies in $\text{Range}(\mathbf{A}^\top)$. There are situations where the dimension of $\text{Range}(\mathbf{A}^\top)$ is much smaller than n . For instance, this happens when $m \ll d$. However, standard coordinate descent methods still move around in directions $e_i \in \mathbb{R}^d$ for all i . We can modify the gradient sketch method to force our gradient estimate to lie in $\text{Range}(\mathbf{A}^\top)$, hoping that this will lead to faster convergence.

C.3.1 The algorithm

Let x^k be the current iterate, and let h^k be the current estimate of the gradient of f . Assume that the sketch $\mathbf{S}_k^\top \nabla f(x^k)$ is available. We can now define h^{k+1} through the following modified sketch-and-project process:

$$\begin{aligned} h^{k+1} &= \arg \min_{h \in \mathbb{R}^d} \|h - h^k\|^2 \\ \text{subject to } & \mathbf{S}_k^\top h = \mathbf{S}_k^\top \nabla f(x^k), \\ & h \in \text{Range}(\mathbf{A}^\top). \end{aligned} \quad (\text{C.18})$$

Standard arguments reveal that the closed-form solution of (C.18) is

$$h^{k+1} = \mathbf{H} (h^k - \mathbf{S}_k (\mathbf{S}_k^\top \mathbf{H} \mathbf{S}_k)^\dagger \mathbf{S}_k^\top (\mathbf{H} h^k - \nabla f(x^k))), \quad (\text{C.19})$$

where

$$\mathbf{H} \stackrel{\text{def}}{=} \mathbf{A}^\top (\mathbf{A} \mathbf{A}^\top)^\dagger \mathbf{A} \quad (\text{C.20})$$

is the projector onto $\text{Range}(\mathbf{A}^\top)$. A quick sanity check reveals that this gives the same formula as (3.4) in the case where $\text{Range}(\mathbf{A}^\top) = \mathbb{R}^d$. We can also write

$$h^{k+1} = \mathbf{H} h^k - \mathbf{H} \mathbf{Z}_k (\mathbf{H} h^k - \nabla f(x^k)) = (\mathbf{I} - \mathbf{H} \mathbf{Z}_k) \mathbf{H} h^k + \mathbf{H} \mathbf{Z}_k \nabla f(x^k), \quad (\text{C.21})$$

where

$$\mathbf{Z}_k \stackrel{\text{def}}{=} \mathbf{S}_k (\mathbf{S}_k^\top \mathbf{H} \mathbf{S}_k)^\dagger \mathbf{S}_k^\top. \quad (\text{C.22})$$

¹Strong convexity is not compatible with the assumption that \mathbf{A} does not have full rank, so a different type of analysis using Polyak-Łojasiewicz inequality is required to give a formal justification. However, we proceed with the analysis anyway to build the intuition why this approach leads to better rates.

Assume that θ_k is chosen in such a way that

$$\mathbb{E}[\theta_k \mathbf{Z}_k] = \mathbf{I}.$$

Then, the following estimate of $\nabla f(x^k)$

$$g^k \stackrel{\text{def}}{=} \mathbf{H}h^k + \theta_k \mathbf{H}\mathbf{Z}_k(\nabla f(x^k) - \mathbf{H}h^k) \quad (\text{C.23})$$

is unbiased, i.e. $\mathbb{E}[g^k] = \nabla f(x^k)$. After evaluating g^k , we perform the same step as in SEGA:

$$x^{k+1} = \text{prox}_{\alpha\psi}(x^k - \alpha g^k).$$

By inspecting (C.18), (C.20) and (C.23), we get the following simple observation.

Lemma C.3.1. *If $h^0 \in \text{Range}(\mathbf{A}^\top)$, then $h^k, g^k \in \text{Range}(\mathbf{A}^\top)$ for all k .*

Consequently, if $h^0 \in \text{Range}(\mathbf{A}^\top)$, (C.19) simplifies to

$$h^{k+1} = h^k - \mathbf{H}\mathbf{S}_k(\mathbf{S}_k^\top \mathbf{H}\mathbf{S}_k)^\dagger \mathbf{S}_k^\top (h^k - \nabla f(x^k)) \quad (\text{C.24})$$

and (C.23) simplifies to

$$g^k \stackrel{\text{def}}{=} h^k + \theta_k \mathbf{H}\mathbf{Z}_k(\nabla f(x^k) - h^k). \quad (\text{C.25})$$

Example 14 (Coordinate sketch). Consider \mathcal{D} given by $\mathbf{S} = e_i$ with probability $p_i > 0$. Then we can choose the bias-correcting random variable as $\theta = \theta(s) = \frac{w_i}{p_i}$, where $w_i \stackrel{\text{def}}{=} \|\mathbf{H}e_i\|_2^2 = e_i^\top \mathbf{H}e_i$. Indeed, with this choice, (3.5) is satisfied. For simplicity, further choose $p_i = 1/n$ for all i . We then have

$$h^{k+1} = h^k - \frac{e_i^\top h^k - e_i^\top \nabla f(x^k)}{w_i} \mathbf{H}e_i = \left(\mathbf{I} - \frac{\mathbf{H}e_i e_i^\top}{w_i} \right) h^k + \frac{\mathbf{H}e_i e_i^\top}{w_i} \nabla f(x^k) \quad (\text{C.26})$$

and (C.25) simplifies to

$$g^k \stackrel{\text{def}}{=} (1 - \theta_k)h^k + \theta_k h^{k+1} = h^k + n \mathbf{H}e_i e_i^\top (\nabla f(x^k) - h^k). \quad (\text{C.27})$$

C.3.2 Lemmas

All theory provided in this subsection is, in fact, a straightforward generalization of our non-subspace results. The reader can recognize similarities in both statements and proofs with that of previous sections.

Lemma C.3.2. *Define \mathbf{Z}_k and \mathbf{H} as in equations (C.22) and (C.20). Then \mathbf{Z}_k is symmetric, $\mathbf{Z}_k \mathbf{H} \mathbf{Z}_k = \mathbf{Z}_k$, $\mathbf{H}^2 = \mathbf{H}$ and $\mathbf{H} = \mathbf{H}^\top$.*

Proof. The symmetry of \mathbf{Z}_k follows from its definition. The second statement is a corollary of the equations $((\mathbf{A}_1 \mathbf{A}_2)^\dagger)^\top = (\mathbf{A}_2^\top \mathbf{A}_1^\top)^\dagger$ and $\mathbf{A}_1^\dagger \mathbf{A}_1 \mathbf{A}_1^\dagger = \mathbf{A}_1^\dagger$, which are true for any matrices $\mathbf{A}_1, \mathbf{A}_2$. Finally, the last two rules follow directly from the definition of \mathbf{H} and the property $\mathbf{A}_1^\dagger \mathbf{A}_1 \mathbf{A}_1^\dagger = \mathbf{A}_1^\dagger$. \square

Lemma C.3.3. Assume $h^k \in \text{Range}(\mathbf{A}^\top)$. Then

$$\mathbb{E} [\|h^{k+1} - v\|^2] = \|h^k - v\|_{\mathbf{I} - \mathbb{E}[\mathbf{Z}]}^2 + \|\nabla f(x^k) - v\|_{\mathbb{E}[\mathbf{Z}]}^2$$

for any vector $v \in \text{Range}(\mathbf{A}^\top)$.

Proof. By Lemma C.3.2 we can rewrite \mathbf{H} as \mathbf{H}^\top , so

$$\begin{aligned} \mathbb{E} [\|h^{k+1} - v\|^2] &\stackrel{(\text{C.21})}{=} \mathbb{E} [\|h^k - \mathbf{H}\mathbf{Z}_k(h^k - \nabla f(x^k)) - v\|^2] \\ &= \mathbb{E} [\|(\mathbf{I} - \mathbf{H}\mathbf{Z}_k)(h^k - v) + \mathbf{H}\mathbf{Z}_k(\nabla f(x^k) - v)\|^2] \\ &= \mathbb{E} [\|(\mathbf{I} - \mathbf{H}^\top \mathbf{Z}_k)(h^k - v) + \mathbf{H}\mathbf{Z}_k(\nabla f(x^k) - v)\|^2] \\ &= \mathbb{E} [\|(\mathbf{I} - \mathbf{H}^\top \mathbf{Z}_k)(h^k - v)\|^2] + \mathbb{E} [\|\mathbf{H}\mathbf{Z}_k(\nabla f(x^k) - v)\|^2] \\ &\quad + 2(h^k - v)^\top \mathbb{E} [(\mathbf{I} - \mathbf{H}^\top \mathbf{Z}_k)^\top \mathbf{H}\mathbf{Z}_k] (\nabla f(x^k) - v) \\ &= (h^k - v)^\top \mathbb{E} [(\mathbf{I} - \mathbf{H}^\top \mathbf{Z}_k)^\top (\mathbf{I} - \mathbf{H}\mathbf{Z}_k)] (h^k - v) \\ &\quad + (\nabla f(x^k) - v)^\top \mathbb{E} [\mathbf{Z}_k \mathbf{H}^\top \mathbf{H}\mathbf{Z}_k] (\nabla f(x^k) - v) \\ &\quad + 2(h^k - v)^\top \mathbb{E} [\mathbf{H}\mathbf{Z}_k - \mathbf{Z}_k \mathbf{H}\mathbf{H}\mathbf{Z}_k] (\nabla f(x^k) - v). \quad (\text{C.28}) \end{aligned}$$

By Lemma C.3.2 we have

$$\mathbf{Z}_k \mathbf{H}\mathbf{H}\mathbf{Z}_k = \mathbf{Z}_k \mathbf{H}\mathbf{Z}_k = \mathbf{Z}_k,$$

so the last term in (C.28) is equal to 0. As for the other two, expanding the matrix factor in the first term leads to

$$\begin{aligned} (\mathbf{I} - \mathbf{H}^\top \mathbf{Z}_k)^\top (\mathbf{I} - \mathbf{H}\mathbf{Z}_k) &= (\mathbf{I} - \mathbf{Z}_k \mathbf{H}) (\mathbf{I} - \mathbf{H}\mathbf{Z}_k) \\ &= \mathbf{I} - \mathbf{Z}_k \mathbf{H} - \mathbf{H}^\top \mathbf{Z}_k + \mathbf{Z}_k \mathbf{H}\mathbf{H}\mathbf{Z}_k \\ &= \mathbf{I} - \mathbf{Z}_k \mathbf{H} - \mathbf{H}^\top \mathbf{Z}_k + \mathbf{Z}_k. \end{aligned}$$

Let us mention that $\mathbf{H}(h^k - v) = h^k - v$ and $(h^k - v)^\top \mathbf{H}^\top = (h^k - v)^\top$ as both vectors h^k and v belong to $\text{Range}(\mathbf{A}^\top)$. Therefore,

$$(h^k - v)^\top \mathbb{E} [\mathbf{I} - \mathbf{Z}_k \mathbf{H} - \mathbf{H}^\top \mathbf{Z}_k + \mathbf{Z}_k] (h^k - v) = (h^k - v)^\top (\mathbf{I} - \mathbb{E}[\mathbf{Z}_k]) (h^k - v).$$

It remains to consider

$$\mathbb{E} [\mathbf{Z}_k \mathbf{H}^\top \mathbf{H}\mathbf{Z}_k] = \mathbb{E} [\mathbf{Z}_k \mathbf{H}\mathbf{H}\mathbf{Z}_k] = \mathbb{E} [\mathbf{Z}_k].$$

We, thereby, have derived

$$\begin{aligned}\mathbb{E} [\|h^{k+1} - v\|^2] &= (h^k - v)^\top (\mathbf{I} - \mathbb{E} [\mathbf{Z}_k]) (h^k - v) \\ &\quad + (\nabla f(x^k) - v)^\top \mathbb{E} [\mathbf{Z}_k \mathbf{Z}_k] (\nabla f(x^k) - v) \\ &= \|h^k - v\|_{\mathbf{I} - \mathbb{E} [\mathbf{Z}_k]}^2 + \|\nabla f(x^k) - v\|_{\mathbb{E} [\mathbf{Z}_k]}^2.\end{aligned}$$

□

Lemma C.3.4. Suppose $h^k \in \text{Range}(\mathbf{A}^\top)$ and g^k is defined by (C.23). Then

$$\mathbb{E} [\|g^k - v\|^2] \leq \|h^k - v\|_{\mathbf{C} - \mathbf{I}}^2 + \|\nabla f(x^k) - v\|_{\mathbf{C}}^2 \quad (\text{C.29})$$

for any $v \in \text{Range}(\mathbf{A}^\top)$, where

$$\mathbf{C} \stackrel{\text{def}}{=} \mathbb{E} [\theta^2 \mathbf{Z}]. \quad (\text{C.30})$$

Proof. Writing $g^k - v = a + b$, where $a \stackrel{\text{def}}{=} (\mathbf{I} - \theta_k \mathbf{H} \mathbf{Z}_k)(h^k - v)$ and $b \stackrel{\text{def}}{=} \theta_k \mathbf{H} \mathbf{Z}_k(\nabla f(x^k) - v)$, we get $\|g^k\|^2 \leq 2(\|a\|^2 + \|b\|^2)$. By definition of θ_k ,

$$\begin{aligned}\mathbb{E} [\|a\|^2] &= \mathbb{E} [\|(\mathbf{I} - \theta_k \mathbf{H} \mathbf{Z}_k)(h^k - v)\|^2] \\ &= (h^k - v)^\top \mathbb{E} [(\mathbf{I} - \theta_k \mathbf{Z}_k \mathbf{H})(\mathbf{I} - \theta_k \mathbf{H} \mathbf{Z}_k)] (h^k - v) \\ &= \|h^k - v\|_{\mathbb{E}[(\mathbf{I} - \theta_k \mathbf{Z}_k \mathbf{H} \mathbf{I} - \theta_k \mathbf{H} \mathbf{Z}_k + \theta_k^2 \mathbf{Z}_k \mathbf{H} \mathbf{H} \mathbf{Z}_k)]}^2.\end{aligned}$$

According to Lemma C.3.2, $\mathbf{H} = \mathbf{H}$ and $\mathbf{Z}_k \mathbf{H} \mathbf{Z}_k = \mathbf{Z}_k$, so

$$\begin{aligned}\mathbb{E} [\|a\|^2] &= (h^k - v)^\top \mathbb{E} [(\mathbf{I} - \theta_k \mathbf{Z}_k \mathbf{H} - \theta_k \mathbf{H}^\top \mathbf{Z}_k + \theta_k^2 \mathbf{Z}_k)] (h^k - v) \\ &= \|h^k - v\|_{\mathbb{E}[\theta^2 \mathbf{Z}] - \mathbf{I}}^2,\end{aligned}$$

where in the last step we used the assumption that h^k and v are from $\text{Range}(\mathbf{A}^\top)$ and \mathbf{H} is the projector operator onto $\text{Range}(\mathbf{A}^\top)$.

Similarly, the second term in the upper bound on g^k can be rewritten as

$$\begin{aligned}\mathbb{E} [\|b\|^2] &= \mathbb{E} [\|\theta_k \mathbf{H} \mathbf{Z}_k(\nabla f(x^k) - v)\|^2] \\ &= (\nabla f(x^k) - v)^\top \mathbb{E} [\theta_k^2 \mathbf{Z}_k \mathbf{H}^\top \mathbf{H} \mathbf{Z}_k] (\nabla f(x^k) - v) \\ &= \|\nabla f(x^k) - v\|_{\mathbb{E}[\theta_k^2 \mathbf{Z}_k]}^2.\end{aligned}$$

Combining the pieces, we get the claim. □

C.3.3 Main result

The main result of this section is:

Theorem C.3.5. Assume that f is \mathbf{Q} -smooth, μ -strongly convex, and that $\alpha > 0$ is such that

$$\alpha(2(\mathbf{C} - \mathbf{I}) + \sigma\mu\mathbf{I}) \leq \sigma\mathbb{E} [\mathbf{Z}], \quad \alpha\mathbf{C} \leq \frac{1}{2}(\mathbf{Q} - \sigma\mathbb{E} [\mathbf{Z}]). \quad (\text{C.31})$$

If we define $\Phi^k \stackrel{\text{def}}{=} \|x^k - x^*\|^2 + \sigma\alpha\|h^k - \nabla f(x^k)\|^2$, then $\mathbb{E}[\Phi^k] \leq (1 - \alpha\mu)^k \Phi^0$.

Proof. Having established Lemmas C.3.2, C.3.3 and C.3.4, the proof follows the same steps as the proof of Theorem 3.4.2. \square

C.3.4 The conclusion of subspace SEGA

Let us recall that $g^k = h^k + \theta_k \mathbf{Z}_k(\nabla f(x^k) - h^k)$. A careful examination shows that when we reduce θ_k from $\mathcal{O}(n)$ to $\mathcal{O}(d)$, we put more trust in the value of h^k with the benefit of reducing the variance of g^k . This insight points out that a practical implementation of the algorithm may exploit the fact that h^k learns the gradient of f by using smaller θ_k .

It is also worth noting that SEGA is a stationary point algorithm regardless of the value of θ_k . Indeed, if one has $x^k = x^*$ and $h^k = \nabla f(x^*)$, then $g^k = \nabla f(x^*)$ for any θ_k . Therefore, once we get a reasonable h^k , it is well grounded to choose g^k to be closer to h^k . This argument is also supported by our experiments.

Finally, the ability to take bigger stepsizes is also of high interest. One can think of extending other methods in this direction, especially if interested in applications with a small rank of matrix \mathbf{A} .

C.4 Simplified analysis of SEGA

In this section we consider the setup from Example 3 with uniform probabilities: $p_i = 1/d$ for all i . We now state the main complexity result.

Theorem C.4.1. *Choose \mathcal{D} to be the uniform distribution over unit basis vectors in \mathbb{R}^d . For any $\sigma > 0$ define*

$$\Phi^k \stackrel{\text{def}}{=} \|x^k - x^*\|_2^2 + \sigma\alpha\|h^k\|_2^2,$$

where $\{x^k, h^k\}_{k \geq 0}$ are the iterates of the gradient sketch method. If the stepsize satisfies

$$0 < \alpha \leq \min \left\{ \frac{1 - \frac{L\sigma}{n}}{2Ld}, \frac{1}{d \left(\mu + \frac{2(d-1)}{\sigma} \right)} \right\}, \quad (\text{C.32})$$

then $\mathbb{E}[\Phi^{k+1}] \leq (1 - \alpha\mu)\Phi^k$. This means that

$$k \geq \frac{1}{\alpha\mu} \log \frac{1}{\epsilon} \quad \Rightarrow \quad \mathbb{E}[\Phi^k] \leq \epsilon \Phi^0.$$

In particular, if we let $\sigma = \frac{d}{2L}$, then $\alpha = \frac{1}{(4L+\mu)d}$ satisfies (C.32), and we have the iteration complexity

$$d \left(4 + \frac{1}{\kappa} \right) \kappa \log \frac{1}{\epsilon} = \tilde{\mathcal{O}}(d\kappa),$$

where $\kappa \stackrel{\text{def}}{=} \frac{L}{\mu}$ is the condition number.

This is the same complexity as NSync [175] under the same assumptions on f . NSync also needs just access to partial derivatives. However, NSync uses variable stepsizes, while

SEGA can do the same with *fixed* stepsizes. This is because SEGA *learns* the direction g^k using past information.

C.4.1 Technical lemmas

Since f is L -smooth, we have

$$\|\nabla f(x^k)\|_2^2 \leq 2L(f(x^k) - f(x^*)). \quad (\text{C.33})$$

On the other hand, by μ -strong convexity of f we have

$$f(x^*) \geq f(x^k) + \langle \nabla f(x^k), x^* - x^k \rangle + \frac{\mu}{2} \|x^* - x^k\|_2^2. \quad (\text{C.34})$$

Lemma C.4.2. *The variance of g^k as an estimator of $\nabla f(x^k)$ can be bounded as follows:*

$$\mathbb{E} [\|g^k\|_2^2] \leq 4Ln(f(x^k) - f(x^*)) + 2(d-1)\|h^k\|_2^2. \quad (\text{C.35})$$

Proof. In view of (3.9), we first write

$$g^k = \underbrace{h^k - \frac{1}{p_i} e_i^\top h^k e_i}_a + \underbrace{\frac{1}{p_i} e_i^\top \nabla f(x^k) e_i}_b,$$

and note that $p_i = 1/n$ for all i . Let us bound the expectation of each term individually. The first term is equal to

$$\begin{aligned} \mathbb{E} [\|a\|_2^2] &= \mathbb{E} [\|h^k - de_i^\top h^k e_i\|_2^2] \\ &= \mathbb{E} [\|(\mathbf{I} - de_i e_i^\top) h^k\|_2^2] \\ &= (h^k)^\top \mathbb{E} [(\mathbf{I} - de_i e_i^\top)^\top (\mathbf{I} - de_i e_i^\top)] h^k \\ &= (d-1)\|h^k\|_2^2. \end{aligned}$$

The second term can be bounded as

$$\begin{aligned} \mathbb{E} [\|b\|_2^2] &= \mathbb{E} [\|de_i^\top \nabla f(x^k) e_i\|_2^2] \\ &= d^2 \sum_{i=1}^d \frac{1}{d} (e_i^\top \nabla f(x^k))^2 \\ &= d \|\nabla f(x^k)\|_2^2 \\ &= d \|\nabla f(x^k) - \nabla f(x^*)\|_2^2 \\ &\stackrel{(\text{C.33})}{\leq} 2Ld(f(x^k) - f(x^*)), \end{aligned}$$

where in the last step we used L -smoothness of f . It remains to combine the two bounds. \square

Lemma C.4.3. For all $v \in \mathbb{R}^d$ we have

$$\mathbb{E} [\|h^{k+1}\|_2^2] = \left(1 - \frac{1}{d}\right) \|h^k\|_2^2 + \frac{1}{d} \|\nabla f(x^k) - v\|_2^2. \quad (\text{C.36})$$

Proof. We have

$$\begin{aligned} \mathbb{E} [\|h^{k+1}\|_2^2] &\stackrel{(3.8)}{=} \mathbb{E} [\|h^k + e_{i_k}^\top (\nabla f(x^k) - h^k) e_{i_k}\|_2^2] \\ &= \mathbb{E} [\|(\mathbf{I} - e_{i_k} e_{i_k}^\top) h^k + e_{i_k} e_{i_k}^\top \nabla f(x^k)\|_2^2] \\ &= \mathbb{E} [\|(\mathbf{I} - e_{i_k} e_{i_k}^\top) h^k\|_2^2] + \mathbb{E} [\|e_{i_k} e_{i_k}^\top \nabla f(x^k)\|_2^2] \\ &= (h^k)^\top \mathbb{E} [(\mathbf{I} - e_{i_k} e_{i_k}^\top)^\top (\mathbf{I} - e_{i_k} e_{i_k}^\top)] h^k \\ &\quad + (\nabla f(x^k))^\top \mathbb{E} [(e_{i_k} e_{i_k}^\top)^\top e_{i_k} e_{i_k}^\top] \nabla f(x^k) \\ &= (h^k)^\top \mathbb{E} [\mathbf{I} - e_{i_k} e_{i_k}^\top] h^k + (\nabla f(x^k))^\top \mathbb{E} [e_{i_k} e_{i_k}^\top] \nabla f(x^k) \\ &= \left(1 - \frac{1}{d}\right) \|h^k\|_2^2 + \frac{1}{d} \|\nabla f(x^k)\|_2^2. \end{aligned}$$

□

C.4.2 Proof of Theorem C.4.1

We can now write

$$\begin{aligned} \mathbb{E} [\|x^{k+1} - x^*\|_2^2] &= \mathbb{E} [\|x^k - \alpha g^k - x^*\|_2^2] \\ &= \|x^k - x^*\|_2^2 + \alpha^2 \mathbb{E} [\|g^k\|_2^2] - 2\alpha \langle \mathbb{E} [g^k], x^k - x^* \rangle \\ &\stackrel{(3.7)}{=} \|x^k - x^*\|_2^2 + \alpha^2 \mathbb{E} [\|g^k\|_2^2] - 2\alpha \langle \nabla f(x^k), x^k - x^* \rangle \\ &\stackrel{(\text{C.34})}{\leq} (1 - \alpha\mu) \|x^k - x^*\|_2^2 + \alpha^2 \mathbb{E} [\|g^k\|_2^2] - 2\alpha (f(x^k) - f(x^*)). \end{aligned}$$

Using Lemma C.4.2, we can further estimate

$$\begin{aligned} \mathbb{E} [\|x^{k+1} - x^*\|_2^2] &\leq (1 - \alpha\mu) \|x^k - x^*\|_2^2 \\ &\quad + 2\alpha(2Ld\alpha - 1)(f(x^k) - f(x^*)) + 2(d-1)\alpha^2 \|h^k\|_2^2. \end{aligned}$$

Let us now add $\sigma\alpha\mathbb{E}[\|h^{k+1}\|_2^2]$ to both sides of the last inequality. Recalling the definition of the Lyapunov function, and applying Lemma C.1.3, we get

$$\begin{aligned}
\mathbb{E}[\Phi^{k+1}] &\leq (1 - \alpha\mu)\|x^k - x^*\|_2^2 + 2\alpha(2Ld\alpha - 1)(f(x^k) - f(x^*)) \\
&\quad + 2(d-1)\alpha^2\|h^k\|_2^2 + \sigma\alpha\left(1 - \frac{1}{d}\right)\|h^k\|_2^2 + \frac{\sigma\alpha}{d}\|\nabla f(x^k)\|_2^2 \\
&\stackrel{\text{(C.33)}}{\leq} (1 - \alpha\mu)\|x^k - x^*\|_2^2 + 2\alpha\underbrace{\left(2Ld\alpha + \frac{L\sigma}{d} - 1\right)}_{\text{I}}(f(x^k) - f(x^*)) \\
&\quad + \underbrace{\left(1 - \frac{1}{d} + \frac{2(d-1)\alpha}{\sigma}\right)}_{\text{II}}\sigma\alpha\|h^k\|_2^2.
\end{aligned}$$

Let us choose α so that I ≤ 0 and II $\leq 1 - \alpha\mu$. This leads to the bound (C.32). For any $\alpha > 0$ satisfying this bound we therefore have $\mathbb{E}[\Phi^{k+1}] \leq (1 - \alpha\mu)\Phi^k$, as desired. Lastly, as we have freedom to choose σ , let us pick it so as to maximize the upper bound on the stepsize.

Appendix D

Appendix for Chapter 4

D.1 IBGD: Bernoulli alternative to IBCD

As an alternative to computing a random block of partial derivatives of size τm , it is possible to compute the whole gradient with probability τ , and attain the same complexity result. While this can be inserted in all algorithms we propose, we only present an alternative to IBCD, which we call IBGD.

Algorithm 25 Independent Bernoulli Gradient Descent (IBGD)

```

1: Input:  $x^0 \in \mathbb{R}^d$ , probability of computing the whole gradient  $\tau$ , stepsize  $\alpha$ , # of
   parallel units  $n$ 
2: for  $k = 0, 1, 2, \dots$  do
3:   for  $i = 1, \dots, n$  in parallel do
4:     Set  $g_i^k = \begin{cases} \nabla f_i(x^k) & \text{with probability } \tau \\ 0 & \text{with probability } 1 - \tau \end{cases}$  independently
5:      $x_i^{k+1} = x^k - \alpha g_i^k$ 
6:   end for
7:    $x^{k+1} = \frac{1}{n} \sum_{i=1}^n x_i^{k+1}$ 
8: end for

```

Theorem D.1.1. *Suppose that Assumptions 4.4.1 holds and $\nabla f_i(x^*) = 0$ for all i . For Algorithm 25 with $\alpha = \frac{n}{\tau n + 2(1-\tau)} \frac{1}{2L}$ we have*

$$\mathbb{E} [\|x^k - x^*\|^2] \leq \left(1 - \frac{\mu}{2L} \frac{\tau n}{\tau n + 2(1-\tau)}\right)^k \|x^0 - x^*\|^2.$$

Note that IBGD does not perform sparse updates to the server; it is either full (dense), or none. This resembles the most naive asynchronous setup – where each iteration, a random subset of machines communicates with the server¹. Our findings thus show that we can expect perfect linear scaling for such unreal asynchronous setup. In the honest asynchronous setup, we shall still expect good parallel scaling once the sequence of machines that communicate with server somewhat resembles a fixed uniform distribution.

¹In reality, there subset is not drawn from a fixed distribution

D.2 Asynchronous ISGD

In this section we extend ISGD algorithm to the asynchronous setup. In particular, we revisit the method that was considered by [70], extend its convergence to stochastic oracle and show better dependency on quantization noise.

Algorithm 26 Asynchronous ISGD

- 1: **Input:** $x^0 \in \mathbb{R}^d$, partition of \mathbb{R}^d into m blocks u_1, \dots, u_m , ratio of blocks to be sampled τ , stepsize α , # parallel units n
 - 2: **for** $k = 0, 1, 2, \dots$ **do**
 - 3: Worker $i = i_t$ is making update
 - 4: $w^{t-d_i^k} = \frac{1}{n} \sum_{j=1}^n x_j^{t-d_i^k}$
 - 5: $z_i^{k+1} = \text{prox}_{\alpha\psi}(w^{t-d_i^k})$
 - 6: Sample independently and uniformly a subset of τm blocks $U_i^k \subseteq \{u_1, \dots, u_m\}$
 - 7: Sample blocks of stochastic gradient $(g_i^k)_{U_i^k}$ such that $\mathbb{E}[g_i^k | x^k] = \nabla f_i(z_i^{k+1})$
 - 8: $x_i^k = \dots = x_i^{t-d_i^k}$
 - 9: $x_i^{k+1} = x_i^k + \frac{1}{\tau + \frac{1}{n}}(z_i^{k+1} - \alpha(g_i^k) - x_i^k)_{U_i^k}$
 - 10: Send $x_i^{k+1} - x_i^k$ and receive $w^{k+1} = w^k + (x_i^{k+1} - x_i^k)$
 - 11: **end for**
 - 12: **Output:** $x^k = \text{prox}_{\alpha\psi}(w^k)$
-

Let us denote the delay of worker i at moment t by d_i^k .

Theorem D.2.1. Assume f_1, \dots, f_n are L -smooth and μ -strongly convex and let Assumption 4.6.2 be satisfied. Let us run Algorithm 26 for t iterations and assume that delays are bounded: $d_i^k \leq M$ for any i and t . If $\alpha \leq \frac{1}{2L(\tau + \frac{2}{n})}$, then

$$\mathbb{E}\|x^k - x^*\|^2 \leq (1 - \tau\alpha\mu)^{\lfloor t/M \rfloor} C + 4\alpha \frac{\sigma^2}{n},$$

where $C \stackrel{\text{def}}{=} \max_{i=1, \dots, n} \|x^0 - x_i^*\|^2$, $x_i^* \stackrel{\text{def}}{=} x^* - \tau\alpha \nabla f_i(x^*)$ and $\lfloor \cdot \rfloor$ is the floor operator.

Plugging $\alpha = \frac{1}{2L(\tau + \frac{2}{n})}$ gives complexity that will be significantly improving from increasing τ until $\tau = \frac{1}{n}$, and then only if τ jumps from $\frac{1}{n}$ to 1. In contrast, doubling τ from $\frac{2}{n}$ to $\frac{4}{n}$ would make little difference.

We note that if ℓ_1 penalty is used, in practice z_i^k should be rather computed on the parameter server side because it will sparsify the vector for communication back.

D.3 Proofs for Section 4.4

D.3.1 Key techniques

The most important equality used many times to prove the results of this chapter is a simple decomposition of expected distances into the distance of expectation and variance:

$$\mathbb{E}\|X - a\|^2 = \|\mathbb{E}X - a\|^2 + \mathbb{E}\|X - \mathbb{E}X\|^2, \quad (\text{D.1})$$

where X is any random vector with finite variance and a is an arbitrary vector from \mathbb{R}^d .

As almost every algorithm we propose average all updates coming from workers, it will be useful to bound the expected distance of mean of n random variables from the optimum. Lemma D.3.1 provides the result.

Lemma D.3.1. *Suppose that $x^{k+1} = \frac{1}{n} \sum_{i=1}^n x_i^k$. Then, we have*

$$\mathbb{E}\|x^{k+1} - x^*\|^2 \leq \left\| \frac{1}{n} \sum_{i=1}^n \mathbb{E}x_i^{k+1} - x^* \right\|^2 + \frac{1}{n^2} \sum_{i=1}^n \mathbb{E}\|x_i^{k+1} - \mathbb{E}x_i^{k+1}\|^2.$$

Proof. First of all, we have

$$\|x^{t+1} - x^*\| = \left\| \frac{1}{n} \sum_{i=1}^n x_i^k - (x^* - \alpha \nabla f(x^*)) \right\|.$$

Now let us proceed to expectations. Note that for any random vector X we have

$$\mathbb{E}\|X\|^2 = \|\mathbb{E}X\|^2 + \mathbb{E}\|X - \mathbb{E}X\|^2.$$

Applying this to random vector $X \stackrel{\text{def}}{=} \frac{1}{n} \sum_{i=1}^n x_i^k - x^*$, we get

$$\mathbb{E} \left\| \frac{1}{n} \sum_{i=1}^n x_i^k - x^* \right\|^2 = \left\| \mathbb{E} \frac{1}{n} \sum_{i=1}^n x_i^k - x^* \right\|^2 + \mathbb{E} \left\| \frac{1}{n} \sum_{i=1}^n x_i^k - \mathbb{E} \frac{1}{n} \sum_{i=1}^n x_i^k \right\|^2.$$

In addition, in all minibatching schemes x_i^{k+1} are conditionally independent given x^k . Therefore, for the variance term we get

$$\mathbb{E} \left\| \frac{1}{n} \sum_{i=1}^n x_i^k - \mathbb{E} \frac{1}{n} \sum_{i=1}^n x_i^k \right\|^2 = \frac{1}{n^2} \sum_{i=1}^n \mathbb{E}\|x_i^{k+1} - \mathbb{E}x_i^{k+1}\|^2. \quad (\text{D.2})$$

Plugging it into our previous bounds concludes the proof. \square

D.3.2 Proof of Theorem 4.4.2

Proof. From Lemma D.5.4, using $\sigma = 0$ and $\nabla f_i(x^*) = 0$, we immediately obtain

$$\mathbb{E} [\|x^{k+1} - x^*\|^2 \mid x^k] \leq (1 - \mu\alpha\tau) \|x^k - x^*\|^2 = \left(1 - \frac{\mu}{2L} \frac{\tau n}{\tau n + 2(1 - \tau)}\right) \|x^k - x^*\|^2.$$

It remains to apply the above inequality recursively. \square

D.3.3 Proof of Theorem D.1.1

Proof. Clearly,

$$\mathbb{E} x_i^{k+1} = x^k - \alpha\tau \nabla f_i(x^k).$$

Let us now elaborate on the second moments.

Thus,

$$\mathbb{E} \|x_i^{k+1} - \mathbb{E} x_i^{k+1}\|^2 = \alpha^2 \mathbb{E} [\|g_i^k - \tau \nabla f_i(x^k)\|^2] = \tau(1 - \tau) \|\nabla f_i(x)\|^2.$$

Therefore, the conditional variance of x^{k+1} variance is equal to

$$\mathbb{E} \|x^{k+1} - \mathbb{E} x^{k+1}\|^2 = \frac{1}{n^2} \sum_{i=1}^n \mathbb{E} \|x_i^{k+1} - \mathbb{E} x_i^{k+1}\|^2 = \frac{\tau(1 - \tau)}{n^2} \sum_{i=1}^n \|\nabla f_i(x)\|^2.$$

Note that the above equality is exactly (D.10) with $\sigma = 0$. Thus, one can use Lemma D.5.4 (with using $\sigma = 0$ and $\nabla f_i(x^*) = 0$) obtaining

$$\mathbb{E} [\|x^{k+1} - x^*\|^2 \mid x^k] \leq (1 - \mu\alpha\tau) \|x^k - x^*\|^2 = \left(1 - \frac{\mu}{2L} \frac{\tau n}{\tau n + 2(1 - \tau)}\right) \|x^k - x^*\|^2.$$

It remains to apply the above inequality recursively. \square

D.4 Missing parts from Sections 4.5 and 4.5.2

D.4.1 Useful lemmata

Let us start with a variance bound, which will be useful for both Algorithm 10 and Algorithm 9. Define $\Phi(x) = \frac{1}{t} \sum_{i=1}^t \phi_i(x)$, and define $x^+ = x - \alpha(\nabla f_j(x) - \mathbf{J}_j + \bar{\mathbf{J}})_U$ for (uniformly) randomly chosen index $1 \leq j \leq t$ and subset of blocks U of size τm . Define also $\bar{\mathbf{J}} = \frac{1}{t} \sum_{i=1}^t \mathbf{J}_i$.

Lemma D.4.1 (Variance bound). *Assume ϕ is μ -strongly convex and ϕ_j is L -smooth and convex for all j . Suppose that $x^* = \arg \min \Phi(x)$. Then, for any x we have*

$$\mathbb{E} \|x^+ - \mathbb{E} x^+\|^2 \leq 2\alpha^2\tau \left(2L(\phi(x) - \phi(x^*)) + \frac{1}{t} \sum_{j=1}^t \|\mathbf{J}_j - \nabla \phi_j(x^*)\|^2 \right). \quad (\text{D.3})$$

Proof. Since $x^+ = x - \alpha(\nabla\phi_j(x) - \mathbf{J}_j + \bar{\mathbf{J}})_U$ and $\mathbb{E}x^+ = x - \alpha\tau\nabla\phi(x)$, we get

$$\begin{aligned} \mathbb{E}\|x^+ - \mathbb{E}x^+\|^2 &= \alpha^2\mathbb{E}\|\tau\nabla\phi(x) - (\nabla\phi_j(x) - \mathbf{J}_j + \bar{\mathbf{J}})_U\|^2 \\ &= \alpha^2\mathbb{E}\|(\tau\nabla\phi(x) - (\nabla\phi_j(x) - \mathbf{J}_j + \bar{\mathbf{J}}))_U\|^2 + \alpha^2\mathbb{E}\|\tau\nabla\phi(x) - (\tau\nabla\phi(x))_U\|^2 \\ &= \alpha^2\tau\mathbb{E}\|\tau\nabla\phi(x) - (\nabla\phi_j(x) - \mathbf{J}_j + \bar{\mathbf{J}})\|^2 + \alpha^2(1-\tau)\tau^2\|\nabla\phi(x)\|^2. \end{aligned}$$

We will leave the second term as is for now and obtain a bound for the first one. Note that the expression inside the norm is now biased: $\mathbb{E}[\tau\nabla\phi(x) - (\nabla\phi_j(x) - \mathbf{J}_j + \bar{\mathbf{J}})] = (\tau - 1)\nabla\phi(x)$. Therefore,

$$\mathbb{E}\|\tau\nabla\phi(x) - (\nabla\phi_j(x) - \mathbf{J}_j + \bar{\mathbf{J}})\|^2 = (1-\tau)^2\|\nabla\phi(x)\|^2 + \mathbb{E}\|\nabla\phi(x) - (\nabla\phi_j(x) - \mathbf{J}_j + \bar{\mathbf{J}})\|^2.$$

Now, since $\nabla\phi_j(x)$ and \mathbf{J}_j are not independent, we shall decouple them using inequality $\|a + b\|^2 \leq 2\|a\|^2 + 2\|b\|^2$. In particular,

$$\begin{aligned} \mathbb{E}\|\nabla\phi(x) - (\nabla\phi_j(x) - \mathbf{J}_j + \bar{\mathbf{J}})\|^2 &= \mathbb{E}\|\nabla\phi(x) - \nabla\phi_j(x) + \nabla\phi_j(x^*) - \nabla\phi_j(x^*) + \mathbf{J}_j - \bar{\mathbf{J}}\|^2 \\ &\leq 2\mathbb{E}\|\nabla\phi(x) - \nabla\phi_j(x) + \nabla\phi_j(x^*)\|^2 + 2\mathbb{E}\|\mathbf{J}_j - \nabla\phi_j(x^*) - \bar{\mathbf{J}}\|^2. \end{aligned}$$

Both terms can be simplified by expanding the squares. For the first one we have:

$$\begin{aligned} \mathbb{E}\|\nabla\phi(x) - \nabla\phi_j(x) + \nabla\phi_j(x^*)\|^2 &= \|\nabla\phi(x)\|^2 - 2\langle\nabla\phi(x), \mathbb{E}[\nabla\phi_j(x) - \nabla\phi_j(x^*)]\rangle \\ &\quad + \mathbb{E}\|\nabla\phi_j(x) - \nabla\phi_j(x^*)\|^2 \\ &= -\|\nabla\phi(x)\|^2 + \frac{1}{t} \sum_{j=1}^t \|\nabla\phi_j(x) - \nabla\phi_j(x^*)\|^2. \end{aligned}$$

Similarly,

$$\begin{aligned} \mathbb{E}\|\mathbf{J}_j - \nabla\phi_j(x^*) - \bar{\mathbf{J}}\|^2 &= \frac{1}{t} \sum_{j=1}^t \|\mathbf{J}_j - \nabla\phi_j(x^*)\|^2 - 2\mathbb{E}\langle\mathbf{J}_j - \nabla\phi_j(x^*), \bar{\mathbf{J}}\rangle + \|\bar{\mathbf{J}}\|^2 \\ &= \frac{1}{t} \sum_{j=1}^t \|\mathbf{J}_j - \nabla\phi_j(x^*)\|^2 - \|\bar{\mathbf{J}}\|^2 \\ &\leq \frac{1}{t} \sum_{j=1}^t \|\mathbf{J}_j - \nabla\phi_j(x^*)\|^2. \end{aligned}$$

Coming back to the first bound that we obtained for this lemma, we deduce

$$\begin{aligned} & \mathbb{E} \|x^+ - \mathbb{E}x^+\|^2 \\ & \leq \alpha^2 \tau \left((1-\tau)^2 \|\nabla\phi(x)\|^2 - 2\|\nabla\phi(x)\|^2 + \frac{2}{k} \sum_{j=1}^t \|\nabla\phi_j(x) - \nabla\phi_j(x^*)\|^2 \right) \\ & \quad + \alpha^2 \tau \frac{2}{k} \sum_{j=1}^t \|\mathbf{J}_j - \nabla\phi_j(x^*)\|^2 + \alpha^2 (1-\tau) \tau^2 \|\nabla\phi(x)\|^2. \end{aligned}$$

The coefficient before $\|\nabla\phi(x)\|^2$ is equal to $\alpha^2 \tau ((1-\tau)^2 - 2 + (1-\tau)\tau) = \alpha^2 \tau (1-\tau-2) < 0$, so we can drop this term. By smoothness of each ϕ_j ,

$$\begin{aligned} \frac{1}{t} \sum_{j=1}^t \|\nabla\phi_j(x) - \nabla\phi(x^*)\|^2 & \leq \frac{2L}{t} \sum_{j=1}^t (\phi_j(x) - \phi(x^*) - \langle \nabla\phi_j(x^*), x - x^* \rangle) \\ & = 2L(\phi(x) - \phi(x^*)), \end{aligned} \tag{D.4}$$

where in the last step we used $\frac{1}{t} \sum_{j=1}^t \nabla\phi_j(x^*) = 0$. \square

Lemma D.4.2. *For ISAGA with shared data we have (given the setting from Theorem 4.5.1)*

$$\mathbb{E} \left[\sum_{j=1}^N \|\mathbf{J}_j^{k+1} - \nabla f_j(x^*)\|^2 \mid x^k \right] \leq 2\tau L n(f(x^k) - f(x^*)) + \left(1 - \frac{\tau n}{N}\right) \sum_{j=1}^N \|\mathbf{J}_j^k - \nabla f_j(x^*)\|^2.$$

On the other hand, for distributed ISAGA we have for all i (given the setting from Theorem 4.5.3):

$$\mathbb{E} \left[\sum_{j=1}^l \|\mathbf{J}_{ij}^{k+1} - \nabla f_{ij}(x^*)\|^2 \mid x^k \right] \leq 2\tau L(f_i(x^k) - f_i(x^*)) + \left(1 - \frac{\tau}{l}\right) \sum_{j=1}^l \|\mathbf{J}_{ij}^k - \nabla f_{ij}(x^*)\|^2.$$

Proof. Consider all expectations throughout this proof to be conditioned on x^k . Let j^k be the function index used to obtain x_i^{k+1} from x^k . Then we have $(\mathbf{J}_{j^k}^{k+1})_{U_i^k} = (\nabla f_{j^k}(x^k))_{U_i^k}$. In the rest of the blocks, $\mathbf{J}_{j^k}^{k+1}$ coincides with its previous value. This implies

$$\mathbb{E} \left[\|\mathbf{J}_{j^k}^{k+1} - \nabla f_{j^k}(x^*)\|^2 \mid j^k \right] = \tau \|\nabla f_{j^k}(x^k) - \nabla f_{j^k}(x^*)\|^2 + (1-\tau) \|\mathbf{J}_{j^k}^k - \nabla f_{j^k}(x^*)\|^2. \tag{D.5}$$

Taking expectation with respect to sampling of j^k we obtain for shared data setup:

$$\begin{aligned}
& \mathbb{E} \left[\sum_{j=1}^N \|\mathbf{J}_j^{k+1} - \nabla f_j(x^*)\|^2 \right] \\
&= \sum_{j=1}^N \mathbb{E} [\|\mathbf{J}_j^{k+1} - \nabla f_j(x^*)\|^2] \\
&\stackrel{(D.5)}{=} \sum_{j=1}^N \frac{n}{N} (\tau \|\nabla f_j(x^k) - \nabla f_j(x^*)\|^2 + (1 - \tau) \|\mathbf{J}_j^k - \nabla f_j(x^*)\|^2) \\
&\quad + \sum_{j=1}^N \left(1 - \frac{n}{N}\right) ((1 - \tau) \|\mathbf{J}_j^k - \nabla f_j(x^*)\|^2) \\
&= \tau \frac{n}{N} \sum_{j=1}^N \|\nabla f_j(x^k) - \nabla f_j(x^*)\|^2 + \left(1 - \frac{\tau n}{N}\right) \sum_{j=1}^N \|\mathbf{J}_j^k - \nabla f_j(x^*)\|^2.
\end{aligned}$$

Similarly, for distributed setup we get

$$\mathbb{E} \left[\sum_{j=1}^l \|\mathbf{J}_{ij}^{k+1} - \nabla f_{ij}(x^*)\|^2 \right] \leq \tau \frac{1}{l} \sum_{j=1}^l \|\nabla f_{ij}(x^k) - \nabla f_{ij}(x^*)\|^2 + \left(1 - \frac{\tau}{l}\right) \sum_{j=1}^l \|\mathbf{J}_{ij}^k - \nabla f_{ij}(x^*)\|^2$$

Using (D.4), the first sum of right hand side can be bounded by $2LN(f(x^k) - f(x^*))$ or $2Ll(f(x^k) - f(x^*))$. \square

D.4.2 Proof of Theorem 4.5.3

Proof. First of all, let us verify that it indeed holds $c > 0$ and $\rho \geq 0$. As $\alpha \leq \frac{1}{L(\frac{3}{n} + \tau)}$, we have $c = \frac{1}{n} \left(\frac{1}{\alpha L} - \frac{1}{n} - \tau\right) \geq \frac{1}{n} \left(\frac{3}{n} + \tau - \frac{1}{n} - \tau\right) > 0$. Furthermore, $\alpha\mu \geq 0$, so to show $\rho \geq 0$, it is enough to mention that

$$\frac{1}{l} - \frac{2}{n^2lc} = \frac{1}{l} - \frac{2}{n^2l \left(\frac{1}{\alpha L} - \frac{1}{n} - \tau\right)} \geq \frac{1}{l} - \frac{2}{n^2l \left(\frac{3}{n} + \tau - \frac{1}{n} - \tau\right)} = 0.$$

Now we proceed to the proof of convergence. We are going to decompose the expected distance from x^{k+1} to x^* into its variance and the distance of expected iterates, so let us analyze them separately. The variance can be bounded as follows:

$$\begin{aligned}
& \mathbb{E} [\|x^{k+1} - \mathbb{E}[x^{k+1} \mid x^k]\|^2 \mid x^k] \\
&\stackrel{(D.2)+(D.3)}{\leq} 2 \frac{\alpha^2 \tau}{n} \left(2L(f(x) - f(x^*)) + \frac{1}{ln} \sum_{j=1}^n \sum_{j=1}^l \|\mathbf{J}_{ij} - \nabla f_{ij}(x^*)\|^2 \right). \quad (D.6)
\end{aligned}$$

For the distance of the expected iterates we write

$$\begin{aligned}\|\mathbb{E}[x^{k+1} | x^k] - x^*\|^2 &= \|x^k - \alpha\tau\nabla f(x^k) - x^*\|^2 \\ &\leq (1 - \alpha\tau\mu)\|x^k - x^*\|^2 - 2\alpha\tau(f(x^k) - f(x^*)) \\ &\quad + 2\alpha^2\tau^2L(f(x^k) - f(x^*)).\end{aligned}$$

As is usually done for SAGA, we are going to prove convergence using a Lyapunov function. Namely, let us define

$$\mathcal{L}^k \stackrel{\text{def}}{=} \mathbb{E} \left[\|x^k - x^*\|^2 + c\alpha^2 \sum_{i=1}^n \sum_{j=1}^l \|\mathbf{J}_{ij}^k - \nabla f_{ij}(x^*)\|^2 \right], \quad (\text{D.7})$$

where $c = \frac{1}{n} \left(\frac{1}{\alpha L} - \frac{1}{n} - \tau \right)$. Using Lemma D.4.2 together with the bounds above, we get

$$\begin{aligned}\mathcal{L}^{k+1} &\leq \mathbb{E} \left[(1 - \alpha\tau\mu)\|x^k - x^*\|^2 + \left(\frac{2\alpha^2\tau}{n^2l} + c\alpha^2 \left(1 - \frac{\tau}{l} \right) \right) \sum_{i=1}^n \sum_{j=1}^l \|\mathbf{J}_{ij}^k - \nabla f_{ij}(x^*)\|^2 \right] \\ &\quad + 2\alpha\tau\mathbb{E} \left[\underbrace{\left(\alpha\tau L + \frac{\alpha L}{n} + c\alpha Ln - 1 \right)}_{=0 \text{ by our choice of } c} (f(x^k) - f(x^*)) \right].\end{aligned}$$

In fact, we chose c exactly to make the last expression equal to zero. After dropping it, we reduce the bound to

$$\mathcal{L}^{k+1} \leq (1 - \rho)\mathbb{E} \left[\|x^k - x^*\|^2 + c\alpha^2 \sum_{i=1}^n \sum_{j=1}^l \|\mathbf{J}_{ij}^k - \nabla f_{ij}(x^*)\|^2 \right] = (1 - \rho)\mathcal{L}^k,$$

where $\rho = \min \left\{ \alpha\tau\mu, \frac{\tau}{l} - \frac{2\tau}{n^2lc} \right\}$. Note that $\mathbb{E}\|x^k - x^*\|^2 \leq \mathcal{L}^k \leq (1 - \rho)^k \mathcal{L}^0$ by induction, so we have the stated linear rate. \square

D.4.3 Proof of Theorem 4.5.1

Proof. First of all, let us verify that it indeed holds $c > 0$ and $\rho \geq 0$. As $\alpha \leq \frac{1}{L(\frac{3}{n} + \tau)}$, we have $c = \frac{1}{n} \left(\frac{1}{\alpha L} - \frac{1}{n} - \tau \right) \geq \frac{1}{n} \left(\frac{3}{n} + \tau - \frac{1}{n} - \tau \right) > 0$. Furthermore, $\alpha\mu \geq 0$, so to show $\rho \geq 0$, it is enough to mention that

$$\frac{n}{N} - \frac{2}{nNc} = \frac{n}{N} - \frac{2}{N \left(\frac{1}{\alpha L} - \frac{1}{n} - \tau \right)} \geq \frac{n}{N} - \frac{2}{N \left(\frac{3}{n} + \tau - \frac{1}{n} - \tau \right)} = 0.$$

Now we proceed to the proof of convergence. We are going to decompose the expected distance from x^{k+1} to x^* into its variance and the distance of expected iterates, so let us

analyze them separately. The variance term can be bounded as follows

$$\begin{aligned} & \mathbb{E} [\|x^{k+1} - \mathbb{E}[x^{k+1} \mid x^k]\|^2 \mid x^k] \\ & \stackrel{(D.2)+(D.3)}{\leq} \frac{2\alpha^2\tau}{n} \left(2L(f(x^k) - f(x^*)) + \frac{1}{N} \sum_{j=1}^N \|\mathbf{J}_j^k - \nabla f_j(x^*)\|^2 \right) \quad (D.8) \end{aligned}$$

For the distance of the expected iterates we write

$$\begin{aligned} \|\mathbb{E}[x^{k+1} \mid x^k] - x^*\|^2 &= \|x^k - \alpha\tau\nabla f(x^k) - x^*\|^2 \\ &\leq (1 - \alpha\tau\mu)\|x^k - x^*\|^2 - 2\alpha\tau(f(x^k) - f(x^*)) + 2\alpha^2\tau^2L(f(x^k) - f(x^*)). \end{aligned}$$

As is usually done for SAGA, we are going to prove convergence using a Lyapunov function. Namely, let us define

$$\mathcal{L}^k \stackrel{\text{def}}{=} \mathbb{E} \left[\|x^k - x^*\|^2 + c\alpha^2 \sum_{j=1}^N \|\mathbf{J}_j^k - \nabla f_j(x^*)\|^2 \right],$$

where $c = \frac{1}{n} \left(\frac{1}{\alpha L} - \frac{1}{n} - \tau \right)$. Using Lemma D.4.2 together with the bounds above, we get

$$\begin{aligned} \mathcal{L}^{k+1} &\leq \mathbb{E} \left[(1 - \alpha\tau\mu)\|x^k - x^*\|^2 + \left(\frac{2\alpha^2\tau}{nN} + c\alpha^2 \left(1 - \frac{\tau n}{N} \right) \right) \sum_{j=1}^N \|\mathbf{J}_j^k - \nabla f_j(x^*)\|^2 \right] \\ &\quad + 2\alpha\tau\mathbb{E} \left[\underbrace{\left(\alpha\tau L + \frac{\alpha L}{n} + c\alpha L n - 1 \right)}_{=0 \text{ by our choice of } c} (f(x^k) - f(x^*)) \right]. \end{aligned}$$

In fact, we chose c exactly to make the last expression equal to zero. After dropping it, we reduce the bound to

$$\mathcal{L}^{k+1} \leq (1 - \rho)\mathbb{E} \left[\|x^k - x^*\|^2 + c\alpha^2 \sum_{j=1}^N \|\mathbf{J}_j^k - \nabla f_j(x^*)\|^2 \right] = (1 - \rho)\mathcal{L}^k,$$

where $\rho = \min \left\{ \alpha\tau\mu, \frac{\tau n}{N} - \frac{2\tau}{nNc} \right\}$. Note that $\mathbb{E}\|x^k - x^*\|^2 \leq \mathcal{L}^k \leq (1 - \rho)^k \mathcal{L}^0$ by induction, so we have the stated linear rate. \square

D.5 Proofs for Section 4.6

D.5.1 Useful lemmas

The next lemma is a key technical tool to analyze Algorithm 11. It provides a better expression for first and second moments of algorithm iterates.

Lemma D.5.1 (SGD moments). *Consider the randomness of the update of Algorithm 11 at moment t . The first moments of the generated iterates are simply $\mathbb{E}x_i^{k+1} = x^k -$*

$\alpha\tau\nabla f_i(x^k)$ and $\mathbb{E}x^{k+1} = x^k - \alpha\tau\nabla f(x^k)$, while their second moments are:

$$\begin{aligned}\mathbb{E}[\|x_i^{k+1} - \mathbb{E}x_i^{k+1}\|^2 | x^k] &= \alpha^2\tau \left((1-\tau) \|\nabla f_i(x^k)\|^2 + \mathbb{E}\|g_i^k - \nabla f_i(x^k)\|^2 \right), \quad (\text{D.9}) \\ \mathbb{E}[\|x^{k+1} - \mathbb{E}x^{k+1}\|^2 | x^k] &= \alpha^2\frac{\tau}{n^2} \sum_{i=1}^n \left((1-\tau) \|\nabla f_i(x^k)\|^2 + \mathbb{E}\|g_i^k - \nabla f_i(x^k)\|^2 \right).\end{aligned}\quad (\text{D.10})$$

Proof. Clearly,

$$\mathbb{E}x_i^{k+1} = x^k - \alpha\mathbb{E}[(g_i^k)_{U_i^k}] = x^k - \alpha\mathbb{E}[(\nabla f_i(x^k))_{U_i^k}] = x^k - \alpha\tau\nabla f_i(x^k)$$

and, therefore, $\mathbb{E}x^{k+1} = x^k - \alpha\tau\nabla f(x^k)$. Let us now elaborate on the second moments. Using the obtained formula for $\mathbb{E}x_i^{k+1}$, we get $(x_i^{k+1} - \mathbb{E}x_i^{k+1})_{U_i^k} = -\alpha(g_i^k - \tau\nabla f_i(x^k))_{U_i^k}$ and $(x_i^{k+1} - \mathbb{E}x_i^{k+1})_{\bar{U}_i^k} = \alpha\tau(\nabla f_i(x^k))_{\bar{U}_i^k}$ where \bar{U}_i^k is a set of blocks not contained in U_i^k . Thus,

$$\begin{aligned}\mathbb{E}\|x_i^{k+1} - \mathbb{E}x_i^{k+1}\|^2 &= \alpha^2\mathbb{E}\left[\|(g_i^k - \tau\nabla f_i(x^k))_{U_i^k}\|^2 + \tau^2\|(\nabla f_i(x^k))_{\bar{U}_i^k}\|^2\right] \\ &= \alpha^2(\tau\mathbb{E}\|g_i^k - \tau\nabla f_i(x^k)\|^2 + \tau^2(1-\tau)\|\nabla f_i(x^k)\|^2).\end{aligned}$$

Note that $\mathbb{E}g_i^k - \tau\nabla f_i(x^k) = (1-\tau)\nabla f_i(x^k)$, so we can use decomposition (D.1) to write $\mathbb{E}\|g_i^k - \tau\nabla f_i(x^k)\|^2 = (1-\tau)^2\|\nabla f_i(x^k)\|^2 + \mathbb{E}\|g_i^k - \nabla f_i(x^k)\|^2$. This develops our previous statement into

$$\begin{aligned}\mathbb{E}\|x_i^{k+1} - \mathbb{E}x_i^{k+1}\|^2 &= \alpha^2(\tau((1-\tau)^2\|\nabla f_i(x^k)\|^2 + \mathbb{E}\|g_i^k - \nabla f_i(x^k)\|^2) + \tau^2(1-\tau)\|\nabla f_i(x^k)\|^2) \\ &= \alpha^2\tau((1-\tau)\|\nabla f_i(x^k)\|^2 + \mathbb{E}\|g_i^k - \nabla f_i(x^k)\|^2),\end{aligned}$$

which coincides with what we wanted to prove for x_i^{k+1} . As for x^{k+1} , it is merely the average of independent random variables conditioned on x^k . Therefore, its variance is equal to

$$\mathbb{E}\|x^{k+1} - \mathbb{E}x^{k+1}\|^2 = \frac{1}{n^2} \sum_{i=1}^n \mathbb{E}\|x_i^{k+1} - \mathbb{E}x_i^{k+1}\|^2.$$

This concludes the proof. □

Lemma D.5.2. *Let f_i be L -smooth and convex for all i . Then,*

$$\frac{1}{n} \sum_{i=1}^n \|\nabla f_i(x^k)\|^2 \leq 4L(f(x^k) - f(x^*)) + \frac{2}{n} \sum_{i=1}^n \|\nabla f_i(x^*)\|^2. \quad (\text{D.11})$$

Proof. If $\nabla f_i(x^*) = 0$ for all i , we can simply write $\|\nabla f_i(x^k)\|^2 = \|\nabla f_i(x^k) - \nabla f_i(x^*)\|^2 \leq 2L(f(x^k) - f(x^*) - \langle \nabla f_i(x^*), x^k - x^* \rangle) = 2L(f(x^k) - f(x^*))$. Otherwise, we have to use inequality $\|a+b\|^2 \leq 2\|a\|^2 + 2\|b\|^2$ with $a = \nabla f_i(x^k) - \nabla f_i(x^*)$ and $b = \nabla f_i(x^*)$.

We get

$$\begin{aligned}
\sum_{i=1}^n \|\nabla f_i(x^k)\|^2 &\leq 2 \sum_{i=1}^n \|\nabla f_i(x^k) - \nabla f_i(x^*)\|^2 + 2 \sum_{i=1}^n \|\nabla f_i(x^*)\|^2 \\
&\leq 4L \sum_{i=1}^n (f_i(x^k) - f_i(x^*) - \langle \nabla f_i(x^*), x^k - x^* \rangle) + 2 \sum_{i=1}^n \|\nabla f_i(x^*)\|^2 \\
&= 4Ln(f(x^k) - f(x^*)) + 2 \sum_{i=1}^n \|\nabla f_i(x^*)\|^2.
\end{aligned}$$

□

Lemma D.5.3. *Let $f = \mathbb{E}f(\cdot; \xi)$ be μ -strongly and $f(\cdot; \xi)$ be L -smooth and convex almost surely. Then, for any x and y*

$$\mathbb{E}\|\nabla f(x; \xi)\|^2 \leq 4L(f(x) - f(y) - \langle \nabla f(y), x - y \rangle) + 2\mathbb{E}\|\nabla f(y; \xi)\|^2.$$

Proof. The proof proceeds exactly the same way as that of Lemma D.5.2. □

Lemma D.5.4. *Suppose that Assumption 4.4.1 holds. Then, if we have*

$$\begin{aligned}
&2\alpha\tau \left(1 - \alpha\tau L - \frac{2\alpha L(1-\tau)}{n}\right) \mathbb{E}[f(x^k) - f(x^*)] \\
&\leq (1 - \alpha\tau\mu) \mathbb{E}\|x^k - x^*\|^2 - \mathbb{E}\|x^{k+1} - x^*\|^2 + \alpha^2 \frac{\tau}{n} \left(\sigma^2 + 2 \frac{1-\tau}{n} \sum_{i=1}^n \|\nabla f_i(x^*)\|^2 \right).
\end{aligned}$$

Proof. Substituting Assumption 4.6.1 into (D.10), we obtain

$$\mathbb{E} [\|x^{k+1} - \mathbb{E}[x^{k+1} | x^k]\|^2 | x^k] \leq \alpha^2 \frac{\tau}{n^2} \sum_{i=1}^n ((1-\tau)\|\nabla f_i(x^k)\|^2 + \sigma^2). \quad (\text{D.12})$$

We use it together with decomposition (D.1) to write

$$\begin{aligned}
&\mathbb{E} [\|x^{k+1} - x^*\|^2 | x^k] \\
&= \mathbb{E} [\|x^{k+1} | x^k - x^*\|^2] + \mathbb{E} [\|x^{k+1} - \mathbb{E}[x^{k+1} | x^k]\|^2] \\
&\stackrel{(\text{D.12})}{\leq} \|x^k - \alpha\tau \nabla f(x^k) - x^*\|^2 + \alpha^2 \frac{\tau}{n^2} \sum_{i=1}^n ((1-\tau)\|\nabla f_i(x^k)\|^2 + \sigma^2) \\
&\stackrel{(\text{D.11})}{\leq} \|x^k - \alpha\tau \nabla f(x^k) - x^*\|^2 \\
&\quad + \alpha^2 \frac{\tau}{n} \left((1-\tau) \left(4L(f(x^k) - f(x^*)) + \frac{2}{n} \sum_{i=1}^n \|\nabla f_i(x^*)\|^2 \right) + \sigma^2 \right).
\end{aligned}$$

Let us expand the first square:

$$\begin{aligned} & \|x^k - \alpha\tau\nabla f(x^k) - x^*\|^2 \\ &= \|x^k - x^*\|^2 - 2\alpha\tau \langle x^k - x^*, \nabla f(x^k) \rangle + \alpha^2\tau^2 \|\nabla f(x^k)\|^2 \\ &\leq \|x^k - x^*\|^2 - 2\alpha\tau \langle x^k - x^*, \nabla f(x^k) \rangle + \alpha^2\tau 2L(f(x^k) - f(x^*)). \end{aligned}$$

The scalar product gives

$$\langle \nabla f(x^k), x^k - x^* \rangle \geq f(x^k) - f(x^*) + \frac{\mu}{2} \|x^k - x^*\|^2.$$

Combining the produced bounds, we show that

$$\begin{aligned} & \mathbb{E} [\|x^{k+1} - x^*\|^2 \mid x^k] \\ &\leq (1 - \alpha\tau\mu) \|x^k - x^*\|^2 + \left(2\alpha^2\tau L - 2\alpha\tau + \alpha^2(1 - \tau) \frac{4\tau L}{n} \right) (f(x^k) - f(x^*)) \\ &\quad + \alpha^2 \frac{\tau}{n} \left(2(1 - \tau) \frac{1}{n} \sum_{i=1}^n \|\nabla f_i(x^*)\|^2 + \sigma^2 \right). \end{aligned}$$

This is equivalent to our claim. \square

D.5.2 Proof of Theorem 4.6.3

Proof. Only for the purpose of this proof, denote $\alpha_k \stackrel{\text{def}}{=} \alpha^k$ in order to not confuse superscript with power. From the choice of α_k we deduce that $2\alpha_k\tau \left(1 - \alpha_k\tau L - \frac{2\alpha_k L(1-\tau)}{n} \right) \geq \alpha_k\tau$. Therefore, the result of Lemma D.5.4 simplifies to

$$\mathbb{E}[f(x^t) - f(x^*)] \leq \frac{1}{\alpha_k\tau} (1 - \alpha_k\tau\mu) \mathbb{E}\|x^t - x^*\|^2 - \frac{1}{\alpha_k\tau} \mathbb{E}\|x^{t+1} - x^*\|^2 + \alpha_k \frac{E}{n}, \quad (\text{D.13})$$

where $E \stackrel{\text{def}}{=} \sigma^2 + (1 - \tau) \frac{2}{n} \sum_{i=1}^n \|\nabla f_i(x^*)\|^2$. Dividing (D.13) by α_k and summing it for $k = 0, \dots, t$ we obtain

$$\begin{aligned} \sum_{k=0}^t \frac{1}{\alpha_k} \mathbb{E}[f(x^t) - f(x^*)] &\leq \frac{1}{\alpha_0^2\tau} (1 - \alpha_0\tau\mu) \|x^0 - x^*\|^2 - \frac{1}{\alpha_k^2\tau} \mathbb{E}\|x^{k+1} - x^*\|^2 + t \frac{E}{n} \\ &\quad + \frac{1}{\tau} \sum_{k=1}^{t-1} \left(\frac{1}{\alpha_k^2} (1 - \alpha_k\tau\mu) - \frac{1}{\alpha_{k-1}^2} \right) \mathbb{E}\|x^t - x^*\|^2. \end{aligned}$$

Next, notice that

$$\begin{aligned}
\frac{1}{\alpha_k^2} - \frac{1}{\alpha_{k+1}^2} (1 - \alpha_{k+1} \tau \mu) &= \frac{1}{\alpha_k^2} \left(1 - \frac{\alpha_k^2}{\alpha_{k+1}^2} (1 - \alpha_{k+1} \tau \mu) \right) \\
&= \frac{1}{\alpha_k^2} \left(1 - \left(1 + \frac{c}{a + ck} \right)^2 \left(1 - \frac{\tau \mu}{a + c(k+1)} \right) \right) \\
&\stackrel{(*)}{\geq} \frac{1}{\alpha_k^2} \left(1 - \left(1 + \frac{2.125c}{a + ck} \right) \left(1 - \frac{\tau \mu}{a + c(k+1)} \right) \right) \\
&= \frac{1}{\alpha_k^2} \left(1 - \left(1 + \frac{2.125}{4} \frac{1}{\frac{a}{\tau \mu} + \frac{1}{4}k} \right) \left(1 - \frac{1}{\frac{a}{\tau \mu} + \frac{1}{4}k + \frac{1}{4}} \right) \right) \\
&\stackrel{(**)}{\geq} 0.
\end{aligned}$$

Above $(*)$ holds since $\frac{c}{a+ck} \leq \frac{1}{8}$ and $(1 + \epsilon)^2 \leq (1 + 2.125\epsilon)$ for $\epsilon \leq \frac{1}{8}$. Next, inequality $(**)$ holds since function $\varphi(y) = (1 + \frac{2.125}{4y}) \left(1 - \frac{1}{y + \frac{1}{4}} \right)$ is upper bounded by 1 on $[0, \infty)$. Thus, we have

$$\sum_{t=0}^k \frac{1}{\alpha_k} \mathbb{E}[f(x^t) - f(x^*)] \leq \frac{a^2}{\tau} \left(1 - \frac{\tau \mu}{a} \right) \|x^0 - x^*\|^2 + t \frac{E}{n}.$$

All that remains is to mention that by Jensen's inequality $\mathbb{E}f(\hat{x}^k) \leq \frac{1}{(k+1)a + \frac{c}{2}k(k+1)} \sum_{k=0}^k (a + ck) \mathbb{E}f(x^t) = \frac{1}{\sum_{k=0}^k \alpha_k^{-1}} \sum_{t=0}^k \alpha_k^{-1} \mathbb{E}f(x^t)$. □

D.5.3 Proof of Theorem 4.6.5

It will be useful to establish a technical lemma first.

Lemma D.5.5. *Let f be L -smooth and assume that $\frac{1}{n} \sum_{i=1}^n \|\nabla f_i(x) - \nabla f(x)\|^2 \leq \nu^2$ for all x . Then, considering only randomness from iteration t of Algorithm 11,*

$$\mathbb{E}f(x^{k+1}) \leq f(x^k) - \alpha \tau \left(1 - \frac{\alpha \tau L}{2} - \alpha L (1 - \tau) \frac{1}{n} \right) \|\nabla f(x^k)\|^2 + \alpha^2 L \tau \frac{(1 - \tau) \nu^2 + \frac{1}{2} \sigma^2}{n}.$$

Proof. Using smoothness of f and assuming x^k is fixed, we write

$$\begin{aligned}
\mathbb{E}f(x^{k+1}) &\leq f(x^k) + \langle \nabla f(x^k), \mathbb{E} x^{k+1} - x^k \rangle + \frac{L}{2} \mathbb{E} \|x^{k+1} - x^k\|^2 \\
&= f(x^k) - \alpha \tau \|\nabla f(x^k)\|^2 + \frac{L}{2} \mathbb{E} \|x^{k+1} - x^k\|^2.
\end{aligned}$$

It holds

$$\begin{aligned}
& \mathbb{E} \|x^{k+1} - x^k\|^2 \\
&= \mathbb{E} \|\mathbb{E} x^{k+1} - x^k\|^2 + \mathbb{E} \|x^{k+1} - \mathbb{E} [x^{k+1} \mid x^k]\|^2 \\
&\stackrel{(D.10)}{=} \alpha^2 \tau^2 \|\nabla f(x^k)\|^2 + \alpha^2 \tau \frac{1}{n^2} \sum_{i=1}^n ((1-\tau) \|\nabla f_i(x^k)\|^2 + \mathbb{E} \|g_i^k - \nabla f_i(x^k)\|^2) \\
&\stackrel{\text{As. 4.6.1}}{\leq} \alpha^2 \tau^2 \|\nabla f(x^k)\|^2 + \alpha^2 \tau \frac{1}{n^2} \sum_{i=1}^n ((1-\tau) \|\nabla f_i(x^k)\|^2 + \sigma^2).
\end{aligned}$$

Using inequality $\|a + b\|^2 \leq \|a\|^2 + \|b\|^2$ with $a = \nabla f_i(x^k) - \nabla f(x^k)$ and $b = \nabla f(x^k)$ yields

$$\frac{1}{n} \sum_{i=1}^n \|\nabla f_i(x^k)\|^2 \leq \frac{2}{n} \sum_{i=1}^n \|\nabla f_i(x^k) - \nabla f(x^k)\|^2 + 2\|\nabla f(x^k)\|^2 \leq 2\nu^2 + 2\|\nabla f(x^k)\|^2.$$

Putting the pieces together, we prove the claim. \square

We now proceed with Proof of Theorem 4.6.5.

Proof. Taking full expectation in Lemma D.5.5 and telescoping this inequality from 0 to t , we obtain

$$\begin{aligned}
0 &\leq \mathbb{E} f(x^{k+1}) - f^* \\
&\leq f(x^0) - f^* - \alpha\tau \left(1 - \frac{\alpha\tau L}{2} - \alpha L(1-\tau) \frac{1}{n}\right) \sum_{k=0}^t \|\nabla f(x^k)\|^2 \\
&\quad + t\alpha^2 L\tau \frac{(1-\tau)\nu^2 + \frac{1}{2}\sigma^2}{n}.
\end{aligned}$$

Rearranging the gradients and dividing by the coefficient before it, we get the result. \square

D.6 Missing parts from Section 4.7

D.6.1 Proof of Lemma 4.7.3

Proof. Let us first bound a variance of $\frac{1}{\tau}(g_i)_{U_i}$ – an unbiased estimate of $\nabla f_i(x)$, as it will appear later in the derivations:

$$\begin{aligned}
 & \mathbb{E} \left[\left\| \frac{1}{\tau}(g_i)_{U_i} - \nabla f_i(x) \right\|^2 \right] \tag{D.14} \\
 = & \mathbb{E}_g \left[\mathbb{E}_U \left[\left\| \frac{1}{\tau}(g_i)_{U_i} - \nabla f_i(x) \right\|^2 \right] \right] \\
 = & \mathbb{E}_g \left[(1 - \tau) \|\nabla f_i(x)\|^2 + \tau \left\| \frac{1}{\tau}g_i - \nabla f_i(x) \right\|^2 \right] \\
 \stackrel{(D.1)}{=} & (1 - \tau) \|\nabla f_i(x)\|^2 + \tau \left\| \left(\frac{1}{\tau} - 1 \right) \nabla f_i(x) \right\|^2 + \tau \mathbb{E}_g \left[\left\| \frac{1}{\tau}(g_i - \nabla f_i(x)) \right\|^2 \right] \\
 = & (1 - \tau) \|\nabla f_i(x)\|^2 + \tau \left(\frac{1}{\tau} - 1 \right)^2 \|\nabla f_i(x)\|^2 + \frac{1}{\tau} \|g_i - \nabla f_i(x)\|^2 \\
 \stackrel{(4.8)}{\leq} & (1 - \tau) \|\nabla f_i(x)\|^2 + \tau \left(\frac{1}{\tau} - 1 \right)^2 \|\nabla f_i(x)\|^2 + \frac{\bar{\rho}}{\tau} \|\nabla f_i(x)\|^2 + \frac{\bar{\sigma}^2}{\tau} \\
 = & \left(\frac{1}{\tau} - 1 + \frac{\bar{\rho}}{\tau} \right) \|\nabla f_i(x)\|^2 + \frac{\bar{\sigma}^2}{\tau}. \tag{D.15}
 \end{aligned}$$

Next we proceed with bounding the second moment of gradient estimator:

$$\begin{aligned}
\mathbb{E} [\|q\|^2] &= \mathbb{E} \left[\left\| \frac{1}{n\tau} \sum_{i=1}^n (g_i)_{U_i} \right\|^2 \right] \\
&\stackrel{(D.1)}{=} \|\nabla f(x)\|^2 + \mathbb{E} \left[\left\| \frac{1}{n\tau} \sum_{i=1}^n ((g_i)_{U_i} - \nabla f_i(x)) \right\|^2 \right] \\
&\stackrel{(*)}{=} \|\nabla f(x)\|^2 + \frac{1}{n^2} \sum_{i=1}^n \mathbb{E} \left[\left\| \frac{1}{\tau} (g_i)_{U_i} - \nabla f_i(x) \right\|^2 \right] \\
&\stackrel{(D.15)}{\leq} \|\nabla f(x)\|^2 + \frac{1}{n^2} \sum_{i=1}^n \left(\left(\frac{1}{\tau} - 1 + \frac{\bar{\rho}}{\tau} \right) \|\nabla f_i(x)\|^2 + \frac{\bar{\sigma}^2}{\tau} \right) \\
&= \|\nabla f(x)\|^2 + \frac{\bar{\sigma}^2}{n\tau} + \left(\frac{1}{\tau} - 1 + \frac{\bar{\rho}}{\tau} \right) \frac{1}{n^2} \sum_{i=1}^n \|\nabla f_i(x)\|^2 \\
&\stackrel{(4.7)}{\leq} \|\nabla f(x)\|^2 + \frac{\bar{\sigma}^2}{n\tau} + \left(\frac{1}{\tau} - 1 + \frac{\bar{\rho}}{\tau} \right) \frac{1}{n} (\tilde{\rho} \|\nabla f(x)\|^2 + \tilde{\sigma}^2) \\
&= \left(1 + \frac{\tilde{\rho}}{n} \left(\frac{1}{\tau} - 1 + \frac{\bar{\rho}}{\tau} \right) \right) \|\nabla f(x)\|^2 + \frac{\bar{\sigma}^2}{n\tau} + \frac{\tilde{\sigma}^2}{n} \left(\frac{1}{\tau} - 1 + \frac{\bar{\rho}}{\tau} \right) \\
&\stackrel{(4.9)+(4.10)}{=} \hat{\rho} \|\nabla f(x)\|^2 + \frac{\bar{\sigma}^2}{n\tau} + \hat{\sigma}^2.
\end{aligned}$$

Above, $(*)$ holds since $\frac{1}{\tau}(g_i)_{U_i} - \nabla f_i(x)$ is zero mean for all i and U_i, U_j are independent for $i \neq j$. □

D.7 Proofs for Section 4.8

D.7.1 Useful lemmata

First, we mention a basic property of the proximal operator.

Proposition D.7.1. *Let R be a closed and convex function. Then for any $x, y \in \mathbb{R}^d$*

$$\|\text{prox}_{\alpha\psi}(x) - \text{prox}_{\alpha\psi}(y)\| \leq \|x - y\|. \quad (\text{D.16})$$

The next lemma, taken from [77, Lemma B.3], gives a basic recurrence for the sequence $\{h_i^k\}_{t=1}^\infty$ from ISEGA.

Lemma D.7.2. *If $h_i^{k+1} \stackrel{\text{def}}{=} h_i^k + \tau(g_i^k - h^k)$, where $g_i^k \stackrel{\text{def}}{=} h_i^k + \frac{1}{\tau}(\nabla f_i(x^k) - h_i^k)_{U_i^k}$, then*

$$\mathbb{E} [\|h_i^{k+1} - \nabla f_i(x^*)\|^2] = (1 - \tau) \|h_i^k - \nabla f_i(x^*)\|^2 + \tau \|\nabla f_i(x^k) - \nabla f_i(x^*)\|^2. \quad (\text{D.17})$$

We will also require a recurrent bound on sequence $\{g^k\}_{t=1}^\infty$ from ISEGA.

Lemma D.7.3. Consider any vectors v_i and set $v \stackrel{\text{def}}{=} \frac{1}{n} \sum_{i=1}^n v_i$. Then, we have

$$\mathbb{E} [\|g^k - v\|^2] \leq \frac{2}{n^2} \sum_{i=1}^n \left(\left(\frac{1}{\tau} + (n-1) \right) \|\nabla f_i(x^k) - v_i\|^2 + \left(\frac{1}{\tau} - 1 \right) \|h_i^k - v_i\|^2 \right). \quad (\text{D.18})$$

Proof. Writing $g^k - v = a + b$, where

$$a \stackrel{\text{def}}{=} \frac{1}{n} \sum_{i=1}^n \left(h_i^k - v_i - \tau^{-1} (h_i^k - v_i)_{U_i^k} \right)$$

and

$$b \stackrel{\text{def}}{=} \frac{1}{n} \sum_{i=1}^n \tau^{-1} (\nabla f_i(x^k) - v_i)_{U_i^k}$$

we get $\|g^k - v\|^2 = \|a + b\|^2 \leq 2(\|a\|^2 + \|b\|^2)$.

Let us bound $\mathbb{E} [\|b\|^2]$ using Young's inequality $2\langle x, y \rangle \leq \|x\|^2 + \|y\|^2$:

$$\begin{aligned} & \mathbb{E} [\|b\|^2] \\ &= \frac{1}{n^2} \mathbb{E} \left[\left\langle \sum_{i=1}^n \tau^{-1} (\nabla f_i(x^k) - v_i)_{U_i^k}, \sum_{i=1}^n \tau^{-1} (\nabla f_i(x^k) - v_i)_{U_i^k} \right\rangle \right] \\ &= \frac{1}{\tau^2 n^2} \mathbb{E} \left[\sum_{i=1}^n \left\| (\nabla f_i(x^k) - v_i)_{U_i^k} \right\|^2 \right] + \frac{2}{\tau^2 n^2} \mathbb{E} \left[\sum_{i \neq j} \left\langle (\nabla f_i(x^k) - v_i)_{U_i^k}, (\nabla f_j(x^k) - v_j)_{U_j^k} \right\rangle \right] \\ &= \frac{1}{\tau n^2} \sum_{i=1}^n \left\| \nabla f_i(x^k) - v_i \right\|^2 + \frac{2}{n^2} \sum_{i \neq j} \langle \nabla f_i(x^k) - v_i, \nabla f_j(x^k) - v_j \rangle \\ &\leq \frac{1}{\tau n^2} \sum_{i=1}^n \left\| \nabla f_i(x^k) - v_i \right\|^2 + \frac{1}{n^2} \sum_{i \neq j} (\left\| \nabla f_i(x^k) - v_i \right\|^2 + \left\| \nabla f_j(x^k) - v_j \right\|^2) \\ &= \frac{1}{n^2} \left(\frac{1}{\tau} + n - 1 \right) \sum_{i=1}^n \left\| \nabla f_i(x^k) - v_i \right\|^2. \end{aligned}$$

Similarly we bound $\mathbb{E} [\|a\|^2]$:

$$\begin{aligned} \mathbb{E} [\|a\|^2] &= \frac{1}{n^2} \mathbb{E} \left[\left\langle \sum_{i=1}^n \left(h_i^k - v_i - \tau^{-1} (h_i^k - v_i)_{U_i^k} \right), \sum_{i=1}^n \left(h_i^k - v_i - \tau^{-1} (h_i^k - v_i)_{U_i^k} \right) \right\rangle \right] \\ &= \frac{1}{n^2} \mathbb{E} \left[\sum_{i=1}^n \left\langle \left(h_i^k - v_i - \tau^{-1} (h_i^k - v_i)_{U_i^k} \right), \left(h_i^k - v_i - \tau^{-1} (h_i^k - v_i)_{U_i^k} \right) \right\rangle \right] \\ &\quad + \frac{2}{n^2} \mathbb{E} \left[\sum_{i \neq j} \left\langle \left(h_i^k - v_i - \tau^{-1} (h_i^k - v_i)_{U_i^k} \right), \left(h_j^k - v_j - \tau^{-1} (h_j^k - v_j)_{U_j^k} \right) \right\rangle \right] \\ &= \frac{\tau^{-1} - 1}{n^2} \sum_{i=1}^n \|h_i^k - v_i\|^2. \end{aligned}$$

It remains to combine the above results. □

D.7.2 Proof of Theorem 4.8.1

Proof. For convenience, denote $g^k \stackrel{\text{def}}{=} \frac{1}{n} \sum_{i=1}^n g_i^k$. It holds

$$\begin{aligned}
& \mathbb{E}[\|x^{k+1} - x^*\|^2] \\
&= \mathbb{E}[\|\text{prox}_{\alpha\psi}(x^k - \alpha g^k) - \text{prox}_{\alpha\psi}(x^* - \alpha \nabla f(x^*))\|^2] \\
&\stackrel{\text{(D.16)}}{\leq} \mathbb{E}[\|x^k - \alpha g^k - (x^* - \alpha \nabla f(x^*))\|^2] \\
&= \|x^k - x^*\|^2 - 2\alpha \langle \nabla f(x^k) - \nabla f(x^*), x^k - x^* \rangle + \alpha^2 \mathbb{E}[\|g^k - \nabla f(x^*)\|^2] \\
&\stackrel{\text{(D.18)}}{\leq} \|x^k - x^*\|^2 - 2\alpha \langle \nabla f(x^k) - \nabla f(x^*), x^k - x^* \rangle \\
&\quad + \alpha^2 \frac{2}{n^2} \sum_{i=1}^n \left(\left(\frac{1}{\tau} + (n-1) \right) \|\nabla f_i(x^k) - \nabla f_i(x^*)\|^2 + \left(\frac{1}{\tau} - 1 \right) \|h_i^k - \nabla f_i(x^*)\|^2 \right) \\
&\leq \|x^k - x^*\|^2 - \alpha\mu \|x^k - x^*\|^2 - 2\alpha D_f(x^k, x^*) \\
&\quad + \frac{2}{n^2} \sum_{i=1}^n \left(\left(\frac{1}{\tau} + (n-1) \right) \|\nabla f_i(x^k) - \nabla f_i(x^*)\|^2 + \left(\frac{1}{\tau} - 1 \right) \|h_i^k - \nabla f_i(x^*)\|^2 \right). \tag{D.19}
\end{aligned}$$

Moreover, we have from smoothness and convexity of f_i

$$-2D_{f_i}(x^k, x^*) \leq -\frac{1}{L} \|\nabla f_i(x^k) - \nabla f_i(x^*)\|^2. \tag{D.20}$$

Combining the above, for any $\omega \geq 0$ (which we choose later) we get

$$\begin{aligned}
& \mathbb{E}[\|x^{k+1} - x^*\|^2] + \alpha\omega \frac{1}{n} \sum_{i=1}^n \mathbb{E}[\|h_i^{k+1} - \nabla f_i(x^*)\|^2] \\
&\stackrel{\text{(D.19)}+\text{(D.17)}}{\leq} \|x^k - x^*\|^2 - \alpha\mu \|x^k - x^*\|^2 - 2\alpha D_f(x^k, x^*) \\
&\quad + \alpha^2 \frac{2}{n^2} \sum_{i=1}^n \left(\left(\frac{1}{\tau} + (n-1) \right) \|\nabla f_i(x^k) - \nabla f_i(x^*)\|^2 + \left(\frac{1}{\tau} - 1 \right) \|h_i^k - \nabla f_i(x^*)\|^2 \right) \\
&\quad + \alpha\omega \frac{1}{n} \sum_{i=1}^n ((1-\tau)\|h_i^k - \nabla f_i(x^*)\|^2 + \tau\|\nabla f_i(x^k) - \nabla f_i(x^*)\|^2) \\
&\stackrel{\text{(D.20)}}{\leq} \|x^k - x^*\|^2 - \alpha\mu \|x^k - x^*\|^2 - \frac{\alpha}{nL} \sum_{i=1}^n \|\nabla f_i(x^k) - \nabla f_i(x^*)\|^2 \\
&\quad + \alpha^2 \frac{2}{n^2} \sum_{i=1}^n \left(\left(\frac{1}{\tau} + (n-1) \right) \|\nabla f_i(x^k) - \nabla f_i(x^*)\|^2 + \left(\frac{1}{\tau} - 1 \right) \|h_i^k - \nabla f_i(x^*)\|^2 \right) \\
&\quad + \alpha\omega \frac{1}{n} \sum_{i=1}^n ((1-\tau)\|h_i^k - \nabla f_i(x^*)\|^2 + \tau\|\nabla f_i(x^k) - \nabla f_i(x^*)\|^2) \\
&= (1 - \alpha\mu) \|x^k - x^*\|^2 + \left(\omega\tau + \frac{2\alpha}{n} \left(\frac{1}{\tau} + n-1 \right) - \frac{1}{L} \right) \frac{\alpha}{n} \sum_{i=1}^n \|\nabla f_i(x^k) - \nabla f_i(x^*)\|^2 \\
&\quad + \left(\frac{2\alpha}{n} \left(\frac{1}{\tau} - 1 \right) + \omega(1-\tau) \right) \frac{\alpha}{n} \sum_{i=1}^n \|h_i^k - \nabla f_i(x^*)\|^2.
\end{aligned}$$

To get rid of gradient differences in this bound, we want to obtain $\frac{1}{L} \geq \frac{2\alpha}{n}(\frac{1}{\tau} + n - 1) + \omega\tau$, which, in turn, is satisfied if

$$\begin{aligned}\alpha &= \mathcal{O}\left(\frac{1 + \tau n}{L}\right), \\ \omega &= \mathcal{O}\left(\frac{1}{L\tau}\right).\end{aligned}$$

Next, we want to prove contraction with factor $(1 - \alpha\mu)$ in terms of $\|h_i^k - \nabla f_i(x^*)\|^2$, so we require

$$(1 - \alpha\mu)\nu \geq \omega(1 - \tau) + \frac{2\alpha}{n}\left(\frac{1}{\tau} - 1\right)$$

we shall choose α such that the following two properties hold:

$$\begin{aligned}\alpha &= \mathcal{O}\left(\frac{\tau}{\mu}\right), \\ \alpha &= \mathcal{O}\left(\frac{n\tau^2\omega}{1 - \tau}\right) = \mathcal{O}\left(\frac{n\tau}{(1 - \tau)L}\right) \geq \mathcal{O}\left(\frac{n\tau}{L}\right).\end{aligned}$$

In particular, the choice $\omega = \frac{1}{2L\tau}$ and $\alpha = \min\left(\frac{1}{4L(1 + \frac{1}{n\tau})}, \frac{1}{\frac{\mu}{\tau} + \frac{4L}{n\tau}}\right)$ works. \square

D.8 Proofs for Section D.2

One way to analyze a delayed algorithm is to define sequence of epoch start moments T_0, T_1, \dots such that $T_0 = 0$ and $T_{k+1} = \min\{t : t - \max_{i=1, \dots, n} d_i^k \geq T_k\}$. In case delays are bounded uniformly, i.e. for some number M it holds $d_i^k \leq M$ for all i and t , one can show by induction [138] that $T_k \leq Mk$.

In addition, we define for every i sequence

$$z_i^k = x^{t-d_i^k}.$$

For notational simplicity, we will assume that if worker i does not perform an update at iteration t , then all related vectors increase their counter without changing their value, i.e. $g_i^{k+1} = g_i^k$, $U_i^{k+1} = U_i^k$, $z_i^{k+1} = z_i^k$ and $x_i^{k+1} = x_i^k$. Then, we can write a simple identity for x_i^k that holds for any i and k ,

$$x_i^k = x^{t-d_i^k} - \alpha(g_i^k)_{U_i^k} = z_i^k - \alpha(g_i^k)_{U_i^k}. \quad (\text{D.21})$$

D.8.1 Useful lemmata

Lemma D.8.1. *Let Assumption 4.6.2 be satisfied and assume without loss of generality that $d_1^k < \dots < d_n^k$. Then, for any i*

$$\mathbb{E}\|x_i^k - \mathbb{E}[x_i^k \mid z_i^k, x_{i+1}^k, \dots, x_n^k]\|^2 \leq 4\alpha^2\tau\mathbb{E}[\sigma^2 + 2L(f_i(z_i^k) - f_i(x^*) - \langle \nabla f_i(x^*), z_i^k - x^* \rangle)] . \quad (\text{D.22})$$

Proof. Denote by \mathcal{F}_i^k the sigma-algebra generated by $z_i^k, x_{i+1}^k, \dots, x_n^k$. Then,

$$\mathbb{E}[\cdot \mid z_i^k, x_{i+1}^k, \dots, x_n^k] = \mathbb{E}[\cdot \mid \mathcal{F}_i^k] .$$

Since $d_1^k < \dots < d_n^k$, x_i^k is independent of the randomness in x_1^k, \dots, x_{i-1}^k as those vectors were obtained after x_i^k . Recall that

$$x_i^k \stackrel{(\text{D.21})}{=} z_i^k - \alpha(g_i^k)_{U_i^k}$$

and denote $\tilde{x}_i^k \stackrel{\text{def}}{=} z_i^k - \nabla f_i(z_i^k)$. Clearly, by uniform sampling of the blocks $\mathbb{E}[x_i^k \mid \mathcal{F}_i^k] = z_i^k - \tau\alpha\mathbb{E}[g_i^k \mid \mathcal{F}_i^k] = z_i^k - \tau\alpha\nabla f_i(z_i^k)$. Thus,

$$\begin{aligned} & \mathbb{E}\|x_i^k - \mathbb{E}[x_i^k \mid \mathcal{F}_i^k]\|^2 \\ &= \alpha^2\mathbb{E}\|(g_i^k)_{U_i^k} - \tau\nabla f_i(z_i^k)\|^2 \\ &= (1-\tau)\alpha^2\mathbb{E}\|\tau\nabla f_i(z_i^k)\|^2 + \tau\alpha^2\mathbb{E}\|g_i^k - \tau\nabla f_i(z_i^k)\|^2 \\ &= (1-\tau)\alpha^2\tau^2\mathbb{E}\|\nabla f_i(z_i^k)\|^2 + \tau\alpha^2\mathbb{E}[\|\nabla f_i(z_i^k) - \tau\nabla f_i(z_i^k)\|^2 + \|g_i^k - \nabla f_i(z_i^k)\|^2] \\ &\leq \tau\alpha^2\mathbb{E}[\|\nabla f_i(z_i^k)\|^2 + \|g_i^k - \nabla f_i(z_i^k)\|^2] \\ &\leq \tau\alpha^2\mathbb{E}[\|\nabla f_i(z_i^k)\|^2 + 2\sigma^2 + 4L(f_i(z_i^k) - f_i(x^*) - \langle \nabla f_i(x^*), z_i^k - x^* \rangle)] . \end{aligned}$$

In addition,

$$\begin{aligned} \|\nabla f_i(z_i^k)\|^2 &\leq 2\|\nabla f_i(z_i^k) - \nabla f_i(x^*)\|^2 + 2\|\nabla f_i(x^*)\|^2 \\ &\leq 4L(f_i(z_i^k) - f_i(x^*) - \langle \nabla f_i(x^*), z_i^k - x^* \rangle) + 2\sigma^2 . \end{aligned}$$

□

We will use in the proof of Theorem D.2.1 Jensen's inequality for a set of vectors $a_1, \dots, a_n \in \mathbb{R}^d$ in the form

$$\left\| \frac{1}{n} \sum_{i=1}^n a_i \right\|^2 \leq \frac{1}{n} \sum_{i=1}^n \|a_i\|^2 .$$

Lemma D.8.2. *Assume that f_i is L -smooth and μ -strongly convex. If $\tilde{x}_i^k \stackrel{\text{def}}{=} z_i^k - \tau\alpha\nabla f_i(z_i^k)$ and $x_i^* \stackrel{\text{def}}{=} x^* - \tau\alpha\nabla f_i(x^*)$, we have*

$$\|\tilde{x}_i^k - x_i^*\|^2 \leq (1 - \tau\alpha\mu)\|z_i^k - x^*\|^2 - 2\alpha\tau(1 - \tau\alpha L)(f_i(z_i^k) - f_i(x^*) - \langle \nabla f_i(z_i^k), z_i^k - x^* \rangle) .$$

Proof. It holds

$$\|\tilde{x}_i^k - x_i^*\|^2 = \|z_i^k - x^*\|^2 - 2\alpha\tau \langle \nabla f_i(z_i^k) - \nabla f_i(x^*), z_i^k - x^* \rangle + \alpha^2\tau^2 \|\nabla f_i(z_i^k) - \nabla f_i(x^*)\|^2.$$

Moreover, by strong convexity and smoothness of f_i (see e.g. [154])

$$2 \langle \nabla f_i(z_i^k) - \nabla f_i(x^*), z_i^k - x^* \rangle \geq \mu \|z_i^k - x^*\|^2 + 2(f_i(z_i^k) - f_i(x^*) - \langle \nabla f_i(z_i^k), z_i^k - x^* \rangle).$$

On the other hand, convexity and smoothness of f_i together imply

$$\|\nabla f_i(z_i^k) - \nabla f_i(x^*)\|^2 \leq 2L(f_i(z_i^k) - f_i(x^*) - \langle \nabla f_i(z_i^k), z_i^k - x^* \rangle).$$

Consequently,

$$\|\tilde{x}_i^k - x_i^*\|^2 \leq (1 - \tau\alpha\mu) \|z_i^k - x^*\|^2 - 2\tau\alpha(1 - \tau\alpha L)(f_i(z_i^k) - f_i(x^*) - \langle \nabla f_i(z_i^k), z_i^k - x^* \rangle).$$

□

D.8.2 Proof of Theorem D.2.1

We are going to prove a more general result that does not need uniform boundedness of delays over time. Theorem D.2.1 will follow as a special case of the more general theorem.

Theorem D.8.3. *Assume that every f_i is L -smooth and μ -strongly convex and also that the gradients noise has bounded variance at x^* as in Assumption 4.6.2. If also $\alpha \leq \frac{1}{2L(\tau + \frac{2}{n})}$, then for any $t \in [T_k, T_{k+1})$*

$$\mathbb{E} \|x^k - x^*\|^2 \leq (1 - \tau\alpha\mu)^t \max_{i=1, \dots, n} \|x^0 - x_i^*\|^2 + 4\alpha \frac{\sigma^2}{\mu n}.$$

Proof. Recall that we use in the Algorithm $w^k = \frac{1}{n} \sum_{i=1}^n x_i^k$ and that $x^k = \text{prox}_{\alpha\psi}(w^k)$. Next, by non-expansiveness of the proximal operator it holds for all t

$$\begin{aligned} \|x^k - x^*\|^2 &= \|\text{prox}_{\alpha\psi}(w^k) - \text{prox}_{\alpha\psi}(x^* - \alpha\nabla f(x^*))\|^2 \\ &\leq \|w^k - (x^* - \alpha\nabla f(x^*))\|^2. \end{aligned}$$

Denote for simplicity $w^* \stackrel{\text{def}}{=} x^* - \alpha\nabla f(x^*)$. Then, we have shown $\|x^k - x^*\|^2 \leq \|w^k - w^*\|^2$.

Fix any t and assume without loss of generality that $d_1^k < d_2^k < \dots < d_n^k$. Then, using the tower property of expectation

$$\mathbb{E} \|w^k - w^*\|^2 = \mathbb{E} [\mathbb{E} [\|w^k - w^*\|^2 \mid x_2^k, \dots, x_n^k]].$$

At the same time, conditioned on $x_1^k, x_2^k, \dots, x_n^k$ the only randomness in w^k is from x_1^k ,

so

$$\begin{aligned}\mathbb{E} [\|w^k - w^*\|^2 \mid z_1^k, x_2^k, \dots, x_n^k] &= \|\mathbb{E} [w^k - w^* \mid z_1^k, x_2^k, \dots, x_n^k]\|^2 \\ &\quad + \frac{1}{n^2} \mathbb{E} \|x_1^k - \mathbb{E}[x_1^k \mid z_1^k, x_2^k, \dots, x_n^k]\|^2.\end{aligned}$$

By continuing unrolling the first term in the right-hand side we arrive at

$$\begin{aligned}\mathbb{E} \|w^k - w^*\|^2 &\leq \|\mathbb{E} w^k - w^*\|^2 + \frac{1}{n^2} \sum_{i=1}^n \mathbb{E} \|x_i^k - \mathbb{E}[x_i^k \mid z_i^k, x_{i+1}^k, \dots, x_n^k]\|^2 \\ &= \|\mathbb{E} w^k - w^*\|^2 + \frac{1}{n^2} \sum_{i=1}^n \mathbb{E} \|x_i^k - \mathbb{E}[x_i^k \mid z_i^k, x_{i+1}^k, \dots, x_n^k]\|^2 \\ &\stackrel{\text{(D.22)}}{\leq} \|\mathbb{E} w^k - w^*\|^2 + 4\tau\alpha^2 \frac{\sigma^2}{n} \\ &\quad + \frac{8\tau\alpha^2 L}{n^2} \sum_{i=1}^n (f_i(z_i^k) - f_i(x^*) - \langle \nabla f_i(x^*), z_i^k - x^* \rangle).\end{aligned}$$

Moreover, by Jensen's inequality

$$\begin{aligned}\|\mathbb{E} w^k - w^*\|^2 &= \left\| \frac{1}{n} \sum_{i=1}^n \mathbb{E}[x_i^k - x_i^*] \right\|^2 \\ &\leq \frac{1}{n} \sum_{i=1}^n \|\mathbb{E}[x_i^k - x_i^*]\|^2 \\ &= \frac{1}{n} \sum_{i=1}^n \|\mathbb{E}[\mathbb{E}[x_i^k - x_i^* \mid \mathcal{F}_i^k]]\|^2 \\ &\leq \frac{1}{n} \sum_{i=1}^n \mathbb{E} \|\mathbb{E}[x_i^k - x_i^* \mid \mathcal{F}_i^k]\|^2.\end{aligned}$$

Combining it with our older results, we get

$$\mathbb{E} \|w^k - w^*\|^2 \leq \frac{1}{n} \sum_{i=1}^n \mathbb{E} \|\tilde{x}_i^k - x_i^*\|^2 + 4\tau\alpha^2 \frac{\sigma^2}{n} + \frac{8\tau\alpha^2 L}{n^2} \sum_{i=1}^n (f_i(z_i^k) - f_i(x^*) - \langle \nabla f_i(x^*), z_i^k - x^* \rangle).$$

Let us apply Lemma D.8.2 to verify that

$$\begin{aligned}
& \frac{1}{n} \sum_{i=1}^n \|\tilde{x}_i^k - x_i^*\|^2 + \frac{8\tau\alpha^2 L}{n^2} \sum_{i=1}^n (f_i(z_i^k) - f_i(x^*) - \langle \nabla f_i(x^*), z_i^k - x^* \rangle) \\
& \leq (1 - \tau\alpha\mu) \frac{1}{n} \sum_{i=1}^n \|z_i^k - x^*\|^2 \\
& \quad - 2\tau\alpha \underbrace{\left(1 - 2\tau\alpha L - \frac{4\alpha L}{n}\right)}_{\geq 0} \frac{1}{n} \sum_{i=1}^n (f_i(z_i^k) - f_i(x^*) - \langle \nabla f_i(x^*), z_i^k - x^* \rangle) \\
& \leq (1 - \tau\alpha\mu) \frac{1}{n} \sum_{i=1}^n \|z_i^k - x^*\|^2.
\end{aligned}$$

Since $z_i^k = x^{t-d_i^k}$, we have proved

$$\begin{aligned}
\mathbb{E}\|x^k - x^*\|^2 & \leq (1 - \tau\alpha\mu) \frac{1}{n} \sum_{i=1}^n \mathbb{E}\|z_i^k - x^*\|^2 + 4\tau\alpha^2 \frac{\sigma^2}{n} \\
& = (1 - \tau\alpha\mu) \frac{1}{n} \sum_{i=1}^n \mathbb{E}\|x^{t-d_i^k} - x^*\|^2 + 4\tau\alpha^2 \frac{\sigma^2}{n} \\
& \leq (1 - \tau\alpha\mu) \max_i \mathbb{E}\|x^{t-d_i^k} - x^*\|^2 + 4\tau\alpha^2 \frac{\sigma^2}{n}.
\end{aligned}$$

If we define sequences

$$v^k \stackrel{\text{def}}{=} \max_{i=1, \dots, n} \mathbb{E}\|x^{t-d_i^k} - x^*\|^2$$

and

$$\Psi^t \stackrel{\text{def}}{=} \max_{t \in [T_k, T_{k+1})} \left\{ \max\{0, v^k - 4\alpha \frac{\sigma^2}{\mu n}\} \right\},$$

it follows from the above that

$$\mathbb{E}\|x^k - x^*\|^2 - 4\alpha \frac{\sigma^2}{\mu n} \leq (1 - \tau\alpha\mu) (\max_i \mathbb{E}\|x^{t-d_i^k} - x^*\|^2 - 4\alpha \frac{\sigma^2}{\mu n}).$$

Therefore, if $\Psi^{k_0} = 0$ for some k_0 then $\Psi^t = 0$ for all $k \geq k_0$. Otherwise, $\Psi^{t+1} \leq (1 - \tau\alpha\mu)\Psi^t$ and for any $t \in [T_k, T_{k+1})$

$$\mathbb{E}\|x^k - x^*\|^2 \leq \psi^k \leq \Psi^t + 4\alpha \frac{\sigma^2}{\mu n} \leq (1 - \tau\alpha\mu)^t \|x^0 - x^*\|^2 + 4\alpha \frac{\sigma^2}{\mu n}.$$

□

Proposition D.8.4 ([138]). *If delays are uniformly bounded over time, i.e. $d_i^k \leq M$ for any i and t , then $T_k \leq Mk$.*

Combining Theorem D.8.3 and Proposition D.8.4 gives a lower bound on k and implies Theorem D.2.1.

Appendix E

Appendix for Chapter 5

E.1 Summary of complexity results

We provide a comprehensive table for faster navigation through special cases and their iteration complexities. In particular, for each special case of GJS, we provide the leading complexity term (i.e., a $\log \frac{1}{\varepsilon}$ factor is omitted in all results) and a reference to the corresponding corollary where this result is established. We also indicate how the operator \mathcal{B} appearing in the Lyapunov function is picked (this is not needed to run the method; it is only used in the analysis). All details can be found later in the Appendix.

E.2 Several lemmas

E.2.1 Existence lemma

Lemma E.2.1. *Suppose that $\mathcal{X} \in \text{Range}(\mathcal{M})$. Denote $\Gamma(\mathbf{X}) \stackrel{\text{def}}{=} \mathcal{U}\mathbf{X}\mathbf{e}$. Suppose that $\mathbb{E} \left[\left(\Gamma \mathcal{M}^{\frac{1}{2}} \right)^* \Gamma \mathcal{M}^{\frac{1}{2}} \right]$ exists and $\lambda_{\min}(\mathbb{E}[\mathcal{S}]) > 0$. Then, there are $\alpha > 0$ and \mathcal{B} such that (5.12) and (5.13) hold. Moreover, inequalities (5.12), (5.13) hold for $\alpha = 0, \mathcal{B} = 0$ without any extra assumptions.*

Proof. Consider only α, \mathcal{B} such that $\alpha < \lambda_{\min}(\mathbb{E}[\mathcal{S}]) \mu^{-1}$, $\lambda_{\min}(\mathcal{B}^* \mathcal{B}) > 0$, $\lambda_{\max}(\mathcal{B}^* \mathcal{B}) < \infty$. Let $\mathbf{Y} = \mathcal{M}^{\dagger \frac{1}{2}} \mathbf{X}$. Thus we have $\mathbb{E}[\|\mathcal{U}\mathbf{X}\mathbf{e}\|^2] \leq \|\mathbf{Y}\|^2 \lambda_{\max} \mathbb{E} \left[\left(\Gamma \mathcal{M}^{\frac{1}{2}} \right)^* \Gamma \mathcal{M}^{\frac{1}{2}} \right]$.

Thus

$$\begin{aligned} (1 - \alpha\mu) \|\mathcal{B}\mathbf{Y}\|^2 - \left\| (\mathcal{I} - \mathbb{E}[\mathcal{S}])^{\frac{1}{2}} \mathcal{B}\mathbf{Y} \right\|^2 &= \langle (\mathcal{B}\mathbf{Y})^\top, (\mathbb{E}[\mathcal{S}] - \alpha\mu\mathcal{I}) \mathcal{B}\mathbf{Y} \rangle \\ &\geq (\lambda_{\min}(\mathbb{E}[\mathcal{S}]) - \alpha\mu) \|\mathcal{B}\mathbf{Y}\|^2 \\ &\geq (\lambda_{\min}(\mathbb{E}[\mathcal{S}]) - \alpha\mu) \lambda_{\min}(\mathcal{B}^* \mathcal{B}) \|\mathbf{Y}\|^2. \end{aligned}$$

Therefore, to have (5.12), it suffices to set

$$\alpha \leq \frac{\lambda_{\min}(\mathbb{E}[\mathcal{S}]) \lambda_{\min}(\mathcal{B}^* \mathcal{B})}{\mu \lambda_{\min}(\mathcal{B}^* \mathcal{B}) + \frac{2}{n^2} \lambda_{\max} \left[\mathbb{E} \left[\left(\Gamma \mathcal{M}^{\frac{1}{2}} \right)^* \Gamma \mathcal{M}^{\frac{1}{2}} \right] \right]}.$$

Similarly, to satisfy (5.13), it suffices to have

$$\frac{2\alpha}{n} \lambda_{\max} \left(\mathbb{E} \left[\left(\Gamma \mathcal{M}^{\frac{1}{2}} \right)^* \Gamma \mathcal{M}^{\frac{1}{2}} \right] \right) + n \lambda_{\min}(\mathbb{E}[\mathcal{S}]) \lambda_{\max}(\mathcal{B}^* \mathcal{B}) \leq 1.$$

Algorithm		Theory		
#	Name	Cor. of Thm 5.4.2	\mathcal{BX}	Leading complexity term (i.e., $\log \frac{1}{\varepsilon}$ factor omitted)
27	SAGA	Corollary E.4.1	$\beta \mathbf{X}$	$n + \frac{4m}{\mu}$
28	SAGA	Corollary E.4.3	$\mathbf{X} \text{Diag}(b)$	$\max_j \left(\frac{1}{p_j} + \frac{1}{p_j} \frac{4v_j}{\mu n} \right)$
29	SEGA	Corollary E.5.1	$\beta \mathbf{X}$	$d + d \frac{4m}{\mu}$
30	SEGA	Corollary E.5.2	$\text{Diag}(b) \mathbf{X}$	$\max_i \left(\frac{1}{p_i} + \frac{1}{p_i} \frac{4m_i}{\mu} \right)$
31	SVRCD	Corollary E.5.3	$\beta \mathbf{X}$	$\frac{1}{\rho} + \max_i \frac{1}{p_i} \frac{4m_i}{\mu}$
32	SGD-star	Corollary E.6.1	0	$\max_j \frac{1}{p_j} \frac{v_j}{\mu n}$
33	LSVRG	Corollary E.7.1	$\beta \mathbf{X}$	$\frac{1}{\rho} + \max_j \frac{1}{p_j} \frac{4v_j}{\mu n}$
34	B2	Corollary E.8.1	$\beta \mathbf{X}$	$\frac{1}{\rho} + \frac{1}{\delta} \frac{4m}{\mu}$
35	LSVRG-inv	Corollary E.8.2	$\mathbf{X} \text{Diag}(b)$	$\max_j \frac{1}{p_j} + \frac{1}{\delta} \frac{4m}{\mu}$
36	SVRCD-inv	Corollary E.8.3	$\text{Diag}(b) \mathbf{X}$	$\max_i \frac{1}{p_i} + \frac{1}{\delta} \frac{4m}{\mu}$
37	RL	Corollary E.9.1	$\mathbf{X} \text{Diag}(b)$	$\max_{i,j} \left(\frac{1}{p_j} + \frac{1}{p_i} \frac{4m_i^j}{\mu} \right)$
38	LR	Corollary E.9.2	$\text{Diag}(b) \mathbf{X}$	$\max_{i,j} \left(\frac{1}{p_i} + \frac{1}{p_j} \frac{4v_j}{\mu} \right)$
39	SAEGA	Corollary E.10.1	$\mathbf{B} \circ \mathbf{X}$	$\max_{i,j} \left(\frac{1}{p_i q_j} + \frac{1}{p_i q_j} \frac{4m_i^j}{\mu n} \right)$
40	SVRCDG	Corollary E.10.2	$\beta \mathbf{X}$	$\frac{1}{\rho} + \max_{i,j} \frac{1}{p_i q_j} \frac{4m_i^j}{\mu n}$
41	ISAEGA	Corollary E.10.3	$\mathbf{B} \circ \mathbf{X}$	$\max_{j \in N_t, i, t} \left(\frac{1}{p_i^t q_j^t} + \left(1 + \frac{1}{n p_i^t q_j^t} \right) \frac{4m_i^j}{\mu} \right)$
42	ISEGA	Corollary E.10.4	$\mathbf{B} \circ \mathbf{X}$	$\max_{j \in N_t, i, t} \left(\frac{1}{p_i^t N_t } + \left(1 + \frac{1}{n p_i^t N_t } \right) \frac{4m_i^j}{\mu} \right)$
43	JS	Corollary E.11.1	$\beta \mathbf{XB}$	$\frac{4n^{-1} \eta \mu^{-1} \lambda_{\max}(\mathbf{B}^\top \mathbb{E}[\mathbf{R} \mathbf{B}] + \lambda_{\max}(\mathbf{B}^\top \mathbf{B})}{\lambda_{\min}(\mathbf{B}^\top \mathbb{E}[\mathbf{R} \mathbf{B}])}$

Table E.1: Iteration complexity of selected special cases of GJS (Algorithm 14). Whenever m appears in a result, we assume that $\mathbf{M}_j = m\mathbf{I}_d$ for all j (i.e., f_j is m -smooth). Whenever m_i appears in a result, we assume that f is \mathbf{M} -smooth with $\mathbf{M} = \text{Diag}(m_1, \dots, m_d)$. Whenever m_i^j appears in a result, we assume that $\mathbf{M}_j = \text{Diag}(m_1^j, \dots, m_d^j)$. Quantities p_i for $i \in [d]$, p_j for $j \in [n]$, ρ and δ are probabilities defining the algorithms.

A valid choice to satisfy the above is for example α, \mathcal{B} such that

$$\lambda_{\max}(\mathcal{B}^* \mathcal{B}) \leq \frac{1}{2n \lambda_{\min}(\mathbb{E}[\mathcal{S}])}, \quad \alpha \leq \frac{1}{\frac{1}{n} \lambda_{\max}\left(\mathbb{E}\left[\Gamma\left(\mathcal{M}^{\frac{1}{2}}\right)^* \Gamma \mathcal{M}^{\frac{1}{2}}\right]\right)}.$$

□

E.2.2 Smoothness lemmas

Let $h : \mathbb{R}^d \rightarrow \mathbb{R}$ be a differentiable and convex function. The Bregman distance of x and y with respect to h is defined by

$$D_h(x, y) \stackrel{\text{def}}{=} h(x) - h(y) - \langle \nabla h(y), x - y \rangle. \quad (\text{E.1})$$

Lemma E.2.2 (Lemma C.1.1 from the appendix of Chapter 3). *Suppose that function $h : \mathbb{R}^d \rightarrow \mathbb{R}$ is convex and \mathbf{M} -smooth, where $\mathbf{M} \succeq 0$. Then*

$$D_h(x, y) \geq \frac{1}{2} \|\nabla h(y) - \nabla h(x)\|_{\mathbf{M}^\dagger}^2, \quad \forall x, y \in \mathbb{R}^d. \quad (\text{E.2})$$

Further,

$$\langle \nabla h(x) - \nabla h(y), x - y \rangle \geq \|\nabla h(x) - \nabla h(y)\|_{\mathbf{M}^\dagger}^2. \quad (\text{E.3})$$

Proof. Fix y and consider the function $\phi(x) \stackrel{\text{def}}{=} h(x) - \langle \nabla h(y), x \rangle$. Clearly, ϕ is \mathbf{M} -smooth, and hence

$$\phi(x + d) \leq \phi(x) + \langle \nabla \phi(x), d \rangle + \frac{1}{2} \|d\|_{\mathbf{M}}^2, \quad \forall x, d \in \mathbb{R}^d. \quad (\text{E.4})$$

Moreover, since h is convex, ϕ is convex, non-negative and is minimized at y . Letting $t = \nabla h(x) - \nabla h(y)$, this implies that

$$\begin{aligned} \phi(y) &\leq \phi(x - \mathbf{M}^\dagger t) \\ &\stackrel{(\text{E.4})}{\leq} \phi(x) - \langle \nabla \phi(x), \mathbf{M}^\dagger t \rangle + \frac{1}{2} \|\mathbf{M}^\dagger t\|_{\mathbf{M}}^2 \\ &= \phi(x) - \langle t, \mathbf{M}^\dagger t \rangle + \frac{1}{2} \|\mathbf{M}^\dagger t\|_{\mathbf{M}}^2 \\ &= \phi(x) - \frac{1}{2} \|t\|_{\mathbf{M}^\dagger}^2, \end{aligned}$$

which is equivalent to (E.2). In the last step we have used the identities $(\mathbf{M}^\dagger)^\top = (\mathbf{M}^\top)^\dagger = \mathbf{M}^\dagger$ and $\mathbf{M}^\dagger \mathbf{M} \mathbf{M}^\dagger = \mathbf{M}^\dagger$.

To show (E.3), it suffices to sum inequality E.2 applied on vector pairs (x, y) and (y, x) . □

Lemma E.2.3. *Let (5.10) hold. That is, assume that function f_j are convex and \mathbf{M}_j -smooth. Then*

$$D_{f_j}(x, y) \geq \frac{1}{2} \|\nabla f_j(x) - \nabla f_j(y)\|_{\mathbf{M}_j^\dagger}^2, \quad \forall x, y \in \mathbb{R}^d. \quad (\text{E.5})$$

If $x - y \in \text{Null}(\mathbf{M}_j)$, then

$$(i) \quad f_j(x) = f_j(y) + \langle \nabla f_j(y), x - y \rangle, \quad (\text{E.6})$$

$$(ii) \quad \nabla f_j(x) - \nabla f_j(y) \in \text{Null}(\mathbf{M}_j), \quad (\text{E.7})$$

$$(iii) \quad \langle \nabla f_j(x) - \nabla f_j(y), x - y \rangle = 0. \quad (\text{E.8})$$

If, in addition, f_j is bounded below, then $\nabla f_j(x) \in \text{Range}(\mathbf{M}_j)$ for all x .

Proof. Inequality (E.5) follows by applying Lemma E.2.2 for $h = f_j$ and $\mathbf{M} = \mathbf{M}_j$. Identity (E.6) is a direct consequence of (5.10). Combining (E.5) and (E.6), we get $0 \geq \frac{1}{2} \|\nabla f_j(x) - \nabla f_j(y)\|_{\mathbf{M}_j^\dagger}^2$, which implies that

$$\nabla f_j(x) - \nabla f_j(y) \in \text{Null}(\mathbf{M}_j^\dagger) = \text{Null}(\mathbf{M}_j^\top) = \text{Null}(\mathbf{M}_j), \quad (\text{E.9})$$

recovering (E.7). By adding two copies of (E.6) (with the roles of x and y exchanged), we get (E.8). Finally, if f_j is bounded below, then in view of (E.6) there exists $c \in \mathbb{R}$ such that,

$$c \leq \inf_{x \in y + \text{Null}(\mathbf{M}_j)} f_j(x) \stackrel{(\text{E.6})}{=} \inf_{x \in y + \text{Null}(\mathbf{M}_j)} f_j(y) + \langle \nabla f_j(y), x - y \rangle.$$

This implies that $\nabla f_j(y) \in \text{Range}(\mathbf{M}_j^\top) = \text{Range}(\mathbf{M}_j)$. □

Lemma E.2.4. *Assume f is twice continuously differentiable. Then $\mathbf{G}(x) - \mathbf{G}(y) \in \text{Range}(\mathcal{M})$ for all $x, y \in \mathbb{R}^d$.*

Proof. For $\mathbf{G}(x) - \mathbf{G}(y) \in \text{Range}(\mathcal{M})$, it suffices to show that $\nabla f_j(x) - \nabla f_j(y) \in \text{Range}(\mathbf{M}_j)$. Without loss of generality, suppose that $f(z, w)$ (for $x = [z, w]$) is such that $f(z, \cdot)$ is linear (for fixed z ; from (E.6)) and $f(\cdot, w)$ is \mathbf{M}' smooth for full rank \mathbf{M}' . Note that

$$0 \preceq \nabla^2 f(x) = \begin{pmatrix} \nabla_{ww}^2 f(w, z) & \nabla_{wz}^2 f(w, z) \\ \nabla_{zw}^2 f(w, z) & \nabla_{zz}^2 f(w, z) \end{pmatrix} = \begin{pmatrix} \nabla_{ww}^2 f(w, z) & \nabla_{wz}^2 f(w, z) \\ \nabla_{zw}^2 f(w, z) & 0 \end{pmatrix}.$$

Since every submatrix of the above must be positive definite, it is easy to see that we must have both $\nabla_{wz}^2 f(w, z) = 0$, $\nabla_{zw}^2 f(w, z) = 0$. This, however, means that $f(w, z)$ is separable in z, w . Therefore indeed $\nabla f_j(x) - \nabla f_j(y) \in \text{Range}(\mathbf{M}_j)$ for all $x, y \in \mathbb{R}^d$ and all $j \in [n]$. □

E.2.3 Projection lemma

In the next lemma, we establish some basic properties of the interaction of the random projection matrices \mathcal{S} and $\mathcal{I} - \mathcal{S}$ with various matrices, operators, and norms.

Lemma E.2.5. *Let \mathcal{S} be a random projection operator and \mathcal{A} any deterministic linear operator commuting with \mathcal{S} , i.e., $\mathcal{A}\mathcal{S} = \mathcal{S}\mathcal{A}$. Further, let $\mathbf{X}, \mathbf{Y} \in \mathbb{R}^{d \times n}$ and define $\mathbf{Z} = (\mathcal{I} - \mathcal{S})\mathbf{X} + \mathcal{S}\mathbf{Y}$. Then*

$$(i) \quad \mathcal{A}\mathbf{Z} = (\mathcal{I} - \mathcal{S})\mathcal{A}\mathbf{X} + \mathcal{S}\mathcal{A}\mathbf{Y},$$

$$(ii) \quad \|\mathcal{A}\mathbf{Z}\|^2 = \|(\mathcal{I} - \mathcal{S})\mathcal{A}\mathbf{X}\|^2 + \|\mathcal{S}\mathcal{A}\mathbf{Y}\|^2,$$

(iii) $\mathbb{E} [\|\mathcal{A}\mathbf{Z}\|^2] = \|(\mathcal{I} - \mathbb{E}[\mathcal{S}])^{1/2}\mathcal{A}\mathbf{X}\|^2 + \|\mathbb{E}[\mathcal{S}]^{1/2}\mathcal{A}\mathbf{Y}\|^2$, where the expectation is with respect to \mathcal{S} .

Proof. Part (i) follows by noting that \mathcal{A} commutes with $\mathcal{I} - \mathcal{S}$. Part (ii) follows from (i) by expanding the square, and noticing that $(\mathcal{I} - \mathcal{S})\mathcal{S} = 0$. Part (iii) follows from (ii) after using the definition of the Frobenius norm, i.e., $\|\mathbf{M}\|^2 = \text{Tr}(\mathbf{M}^\top \mathbf{M})$, the identities $(\mathcal{I} - \mathcal{S})^2 = \mathcal{I} - \mathcal{S}$, $\mathcal{S}^2 = \mathcal{S}$, and taking expectation on both sides. \square

E.2.4 Decomposition lemma

In the next lemma, we give a bound on the expected squared distance of the gradient estimator g^k from $\nabla f(x^*)$.

Lemma E.2.6. *For all $k \geq 0$ we have*

$$\mathbb{E} [\|g^k - \nabla f(x^*)\|^2] \leq \frac{2}{n^2} \mathbb{E} [\|\mathcal{U}(\mathbf{G}(x^k) - \mathbf{G}(x^*)) \mathbf{e}\|^2] + \frac{2}{n^2} \mathbb{E} [\|\mathcal{U}(\mathbf{J}^k - \mathbf{G}(x^*)) \mathbf{e}\|^2]. \quad (\text{E.10})$$

Proof. In view of (5.6) and since $\nabla f(x^*) = \frac{1}{n} \mathbf{G}(x^*) \mathbf{e}$, we have

$$g^k - \nabla f(x^*) = \underbrace{\frac{1}{n} \mathcal{U}(\mathbf{G}(x^k) - \mathbf{G}(x^*)) \mathbf{e}}_a + \underbrace{\frac{1}{n} (\mathbf{J}^k - \mathbf{G}(x^*)) \mathbf{e} - \frac{1}{n} \mathcal{U}(\mathbf{J}^k - \mathbf{G}(x^*)) \mathbf{e}}_b. \quad (\text{E.11})$$

Applying the bound $\|a + b\|^2 \leq 2\|a\|^2 + 2\|b\|^2$ to (E.11) and taking expectations, we get

$$\begin{aligned} \mathbb{E} [\|g^k - \nabla f(x^*)\|^2] &\leq \mathbb{E} \left[\frac{2}{n^2} \|\mathcal{U}(\mathbf{G}(x^k) - \mathbf{G}(x^*)) \mathbf{e}\|^2 \right] \\ &\quad + \mathbb{E} \left[\frac{2}{n^2} \|(\mathbf{J}^k - \mathbf{G}(x^*)) \mathbf{e} - \mathcal{U}(\mathbf{J}^k - \mathbf{G}(x^*)) \mathbf{e}\|^2 \right] \\ &= \frac{2}{n^2} \mathbb{E} [\|\mathcal{U}(\mathbf{G}(x^k) - \mathbf{G}(x^*)) \mathbf{e}\|^2] \\ &\quad + \frac{2}{n^2} \mathbb{E} [\|(\mathcal{I} - \mathcal{U})(\mathbf{J}^k - \mathbf{G}(x^*)) \mathbf{e}\|^2]. \end{aligned}$$

It remains to note that

$$\begin{aligned} \mathbb{E} [\|(\mathcal{I} - \mathcal{U})(\mathbf{J}^k - \mathbf{G}(x^*)) \mathbf{e}\|^2] &= \mathbb{E} [\|\mathcal{U}(\mathbf{J}^k - \mathbf{G}(x^*)) \mathbf{e}\|^2] - \|(\mathbf{J}^k - \mathbf{G}(x^*)) \mathbf{e}\|^2 \\ &\leq \mathbb{E} [\|\mathcal{U}(\mathbf{J}^k - \mathbf{G}(x^*)) \mathbf{e}\|^2]. \end{aligned}$$

\square

E.3 Proof of Theorem 5.4.2

For simplicity of notation, in this proof, all expectations are conditional on x^k , i.e., the expectation is taken with respect to the randomness of g^k .

Since

$$x^* = \text{prox}_{\alpha\psi}(x^* - \alpha\nabla f(x^*)), \quad (\text{E.12})$$

and since the prox operator is non-expansive, we have

$$\begin{aligned} \mathbb{E} \left[\|x^{k+1} - x^*\|^2 \right] &\stackrel{(\text{E.12})}{=} \mathbb{E} \left[\|\text{prox}_{\alpha\psi}(x^k - \alpha g^k) - \text{prox}_{\alpha\psi}(x^* - \alpha\nabla f(x^*))\|^2 \right] \\ &\leq \mathbb{E} \left[\|x^k - x^* - \alpha(g^k - \nabla f(x^*))\|^2 \right] \\ &\stackrel{(5.7)}{=} \|x^k - x^*\|^2 - 2\alpha \langle \nabla f(x^k) - \nabla f(x^*), x^k - x^* \rangle \\ &\quad + \alpha^2 \mathbb{E} \left[\|g^k - \nabla f(x^*)\|^2 \right] \\ &\stackrel{(5.9)+(E.1)}{\leq} (1 - \alpha\mu) \|x^k - x^*\|^2 + \alpha^2 \mathbb{E} \left[\|g^k - \nabla f(x^*)\|^2 \right] \\ &\quad - 2\alpha D_f(x^k, x^*). \end{aligned} \quad (\text{E.13})$$

Since $f(x) = \frac{1}{n} \sum_{j=1}^n f_j(x)$, in view of (E.1) and (E.5) we have

$$\begin{aligned} D_f(x^k, x^*) &\stackrel{(\text{E.1})}{=} \frac{1}{n} \sum_{j=1}^n D_{f_j}(x^k, x^*) \stackrel{(\text{E.5})}{\geq} \frac{1}{2n} \sum_{j=1}^n \|\nabla f_j(x^k) - \nabla f_j(x^*)\|_{\mathbf{M}_j^\dagger}^2 \\ &= \frac{1}{2n} \left\| \mathcal{M}^{\dagger \frac{1}{2}} (\mathbf{G}(x^k) - \mathbf{G}(x^*)) \right\|^2 \end{aligned} \quad (\text{E.14})$$

By combining (E.13) and (E.14), we get

$$\begin{aligned} \mathbb{E} \left[\|x^{k+1} - x^*\|^2 \right] &\leq (1 - \alpha\mu) \|x^k - x^*\|^2 + \alpha^2 \mathbb{E} \left[\|g^k - \nabla f(x^*)\|^2 \right] \\ &\quad - \frac{\alpha}{n} \left\| \mathcal{M}^{\dagger \frac{1}{2}} (\mathbf{G}(x^k) - \mathbf{G}(x^*)) \right\|^2. \end{aligned}$$

Next, applying Lemma E.2.6 leads to the estimate

$$\begin{aligned} \mathbb{E} \left[\|x^{k+1} - x^*\|^2 \right] &\leq (1 - \alpha\mu) \|x^k - x^*\|^2 - \frac{\alpha}{n} \left\| \mathcal{M}^{\dagger \frac{1}{2}} (\mathbf{G}(x^k) - \mathbf{G}(x^*)) \right\|^2 \\ &\quad + \frac{2\alpha^2}{n^2} \mathbb{E} \left[\|\mathcal{U} (\mathbf{G}(x^k) - \mathbf{G}(x^*)) \mathbf{e}\|^2 \right] \\ &\quad + \frac{2\alpha^2}{n^2} \mathbb{E} \left[\|\mathcal{U} (\mathbf{J}^k - \mathbf{G}(x^*)) \mathbf{e}\|^2 \right]. \end{aligned} \quad (\text{E.15})$$

In view of (5.8), we have $\mathbf{J}^{k+1} = (\mathcal{I} - \mathcal{S})\mathbf{J}^k + \mathcal{S}\mathbf{G}(x^k)$, whence

$$\underbrace{\mathbf{J}^{k+1} - \mathbf{G}(x^*)}_{\mathbf{Z}} = (\mathcal{I} - \mathcal{S}) \underbrace{(\mathbf{J}^k - \mathbf{G}(x^*))}_{\mathbf{X}} + \mathcal{S} \underbrace{(\mathbf{G}(x^k) - \mathbf{G}(x^*))}_{\mathbf{Y}}. \quad (\text{E.16})$$

Since, by assumption, both \mathcal{B} and $\mathcal{M}^{\dagger \frac{1}{2}}$ commute with \mathcal{S} , so does their composition $\mathcal{A} \stackrel{\text{def}}{=} \mathcal{B}\mathcal{M}^{\dagger \frac{1}{2}}$. Applying Lemma E.2.5, we get

$$\begin{aligned} \mathbb{E} \left[\left\| \mathcal{B}\mathcal{M}^{\dagger \frac{1}{2}} (\mathbf{J}^{k+1} - \mathbf{G}(x^*)) \right\|^2 \right] &= \left\| (\mathcal{I} - \mathbb{E}[\mathcal{S}])^{\frac{1}{2}} \mathcal{B}\mathcal{M}^{\dagger \frac{1}{2}} (\mathbf{J}^k - \mathbf{G}(x^*)) \right\|^2 \\ &\quad + \left\| \mathbb{E}[\mathcal{S}]^{\frac{1}{2}} \mathcal{B}\mathcal{M}^{\dagger \frac{1}{2}} (\mathbf{G}(x^k) - \mathbf{G}(x^*)) \right\|^2 \quad (\text{E.17}) \end{aligned}$$

Adding α -multiple of (E.17) to (E.15) yields

$$\begin{aligned} &\mathbb{E} \left[\|x^{k+1} - x^*\|^2 \right] + \alpha \mathbb{E} \left[\left\| \mathcal{B}\mathcal{M}^{\dagger \frac{1}{2}} (\mathbf{J}^{k+1} - \mathbf{G}(x^*)) \right\|^2 \right] \\ &\leq (1 - \alpha\mu) \|x^k - x^*\|^2 + \frac{2\alpha^2}{n^2} \mathbb{E} \left[\left\| \mathcal{U} (\mathbf{G}(x^k) - \mathbf{G}(x^*)) \right\|^2 \right] \\ &\quad + \frac{2\alpha^2}{n^2} \mathbb{E} \left[\left\| \mathcal{U} (\mathbf{J}^k - \mathbf{G}(x^*)) \right\|^2 \right] + \alpha \left\| (\mathcal{I} - \mathbb{E}[\mathcal{S}])^{\frac{1}{2}} \mathcal{B}\mathcal{M}^{\dagger \frac{1}{2}} (\mathbf{J}^k - \mathbf{G}(x^*)) \right\|^2 \\ &\quad + \alpha \left\| \mathbb{E}[\mathcal{S}]^{\frac{1}{2}} \mathcal{B}\mathcal{M}^{\dagger \frac{1}{2}} (\mathbf{G}(x^k) - \mathbf{G}(x^*)) \right\|^2 - \frac{\alpha}{n} \left\| \mathcal{M}^{\dagger \frac{1}{2}} (\mathbf{G}(x^k) - \mathbf{G}(x^*)) \right\|^2 \\ &\stackrel{(5.12)}{\leq} (1 - \alpha\mu) \|x^k - x^*\|^2 + (1 - \alpha\mu)\alpha \left\| \mathcal{B}\mathcal{M}^{\dagger \frac{1}{2}} (\mathbf{J}^k - \mathbf{G}(x^*)) \right\|^2 \\ &\quad + \frac{2\alpha^2}{n^2} \mathbb{E} \left[\left\| \mathcal{U} (\mathbf{G}(x^k) - \mathbf{G}(x^*)) \right\|^2 \right] + \alpha \left\| \mathbb{E}[\mathcal{S}]^{\frac{1}{2}} \mathcal{B}\mathcal{M}^{\dagger \frac{1}{2}} (\mathbf{G}(x^k) - \mathbf{G}(x^*)) \right\|^2 \\ &\quad - \frac{\alpha}{n} \left\| \mathcal{M}^{\dagger \frac{1}{2}} (\mathbf{G}(x^k) - \mathbf{G}(x^*)) \right\|^2 \\ &\stackrel{(5.13)}{\leq} (1 - \alpha\mu) \left(\|x^k - x^*\|^2 + \alpha \left\| \mathcal{B}\mathcal{M}^{\dagger \frac{1}{2}} (\mathbf{J}^k - \mathbf{G}(x^*)) \right\|^2 \right). \end{aligned}$$

Above, we have used (5.12) with $\mathbf{X} = \mathbf{J}^k - \mathbf{G}(x^*)$ and (5.13) with $\mathbf{X} = \mathbf{G}(x^k) - \mathbf{G}(x^*)$.

E.4 Special cases: SAGA-like methods

E.4.1 Basic variant of SAGA [37]

Suppose that for all j , f_j is m -smooth (i.e., $\mathbf{M}_j = m\mathbf{I}_d$). To recover basic SAGA [37], consider the following choice of random operators \mathcal{S}, \mathcal{U} :

$$(\forall j) \text{ with probability } \frac{1}{n} : \quad \mathcal{S}\mathbf{X} = \mathbf{X}\mathbf{e}_j\mathbf{e}_j^\top \quad \text{and} \quad \mathcal{U}\mathbf{X} = \mathbf{X}n\mathbf{e}_j\mathbf{e}_j^\top.$$

The resulting algorithm is stated as Algorithm 27. Further, as a direct consequence of Theorem 5.4.2, convergence rate of SAGA (Algorithm 27) is presented in Corollary E.4.1.

Corollary E.4.1 (Convergence rate of SAGA). *Let $\alpha = \frac{1}{4m+\mu n}$. Then, iteration complexity of Algorithm 27 (proximal SAGA) is $\left(4\frac{m}{\mu} + n\right) \log \frac{1}{\epsilon}$.*

Algorithm 27 SAGA [37]**Require:** learning rate $\alpha > 0$, starting point $x^0 \in \mathbb{R}^d$ Set $\psi_j^0 = x^0$ for each $j \in \{1, 2, \dots, n\}$ **for** $k = 0, 1, 2, \dots$ **do**Sample $j \in [n]$ uniformly at randomSet $\phi_j^{k+1} = x^k$ and $\phi_i^{k+1} = \phi_i^k$ for $i \neq j$

$$g^k = \nabla f_j(\phi_j^{k+1}) - \nabla f_j(\phi_j^k) + \frac{1}{n} \sum_{i=1}^n \nabla f_i(\phi_i^k)$$

$$x^{k+1} = \text{prox}_{\alpha\psi}(x^k - \alpha g^k)$$

end for**E.4.2 SAGA with arbitrary sampling**

In contrast to Section E.4.1, here we use the general matrix smoothness assumption, i.e., that f_j is \mathbf{M}_j smooth. We recover results from [165]. Denote \mathbf{p} to be probability vector, i.e., $\mathbf{p}_i = \mathbb{P}(i \in R)$ where R is a random subset of $[n]$.

We shall consider the following choice of random operators \mathcal{S}, \mathcal{U} :

$$(\forall R) \text{ with probability } \mathbf{p}_R : \quad \mathcal{S}\mathbf{X} = \mathbf{X} \sum_{j \in R} \mathbf{e}_j \mathbf{e}_j^\top \quad \text{and} \quad \mathcal{U}\mathbf{X} = \mathbf{X} \sum_{j \in R} \frac{1}{\mathbf{p}_j} \mathbf{e}_j \mathbf{e}_j^\top.$$

The resulting algorithm is stated as Algorithm 28.

Algorithm 28 SAGA with arbitrary sampling (a variant of [165])**Require:** learning rate $\alpha > 0$, starting point $x^0 \in \mathbb{R}^d$, random sampling $R \subseteq \{1, 2, \dots, n\}$ Set $\phi_j^0 = x^0$ for each $j \in [n]$ **for** $k = 0, 1, 2, \dots$ **do**Sample random $R^k \subseteq \{1, 2, \dots, n\}$

$$\text{Set } \phi_j^{k+1} = \begin{cases} x^k & j \in R^k \\ \phi_j^k & j \notin R^k \end{cases}$$

$$g^k = \frac{1}{n} \sum_{j=1}^n \nabla f_j(\phi_j^k) + \sum_{j \in R^k} \frac{1}{n\mathbf{p}_j} (\nabla f_j(\phi_j^{k+1}) - \nabla f_j(\phi_j^k))$$

$$x^{k+1} = \text{prox}_{\alpha\psi}(x^k - \alpha g^k)$$

end for

In order to give tight rates under \mathbf{M} -smoothness, we need to do a bit more work. First, let $v \in \mathbb{R}^n$ be a vector for which the following inequality *expected separable over-approximation* inequality holds

$$\mathbb{E} \left[\left\| \sum_{j \in R} \mathbf{M}_j^{\frac{1}{2}} h_j \right\|^2 \right] \leq \sum_{j=1}^n \mathbf{p}_j v_j \|h_j\|^2, \quad \forall h_1, \dots, h_n \in \mathbb{R}^d. \quad (\text{E.18})$$

Since the function on the left is a quadratic in $h = (h_1, \dots, h_n) \in \mathbb{R}^{nd}$, this inequality is satisfied for large enough values of v_j . A variant of (E.18) was used to obtain the best known rates for coordinate descent with arbitrary sampling [166, 167].

Further, we shall consider the following assumption:

Assumption E.4.2. Suppose that for all k

$$\mathbf{G}(x^k) - \mathbf{G}(x^*) = \mathcal{M}^\dagger \mathcal{M} (\mathbf{G}(x^k) - \mathbf{G}(x^*)) \quad (\text{E.19})$$

and

$$\mathbf{J}^k - \mathbf{G}(x^*) = \mathcal{M}^\dagger \mathcal{M} (\mathbf{J}^k - \mathbf{G}(x^*)). \quad (\text{E.20})$$

The assumption, although in a slightly less general form, was demonstrated to obtain tightest complexity results for SAGA [165]. Note that if for each j , f_j corresponds to loss function of a linear model, then (E.19) and (E.20) follow for free. Further, Lemmas E.2.3 and E.2.4 give some easy-to-interpret sufficient conditions, such as lower boundness of all functions f_j (which happens for any loss function), or twice differentiability of all functions f_j .

Corollary E.4.3 (Convergence rate of SAGA). Let $\alpha = \min_j \frac{np_j}{4v_j + n\mu}$. Then the iteration complexity of Algorithm 28 is $\max_j \left(\frac{4v_j + n\mu}{n\mu p_j} \right) \log \frac{1}{\epsilon}$.

Remark 20. Corollary E.4.3 is slightly more general than Theorem 4.6 from [165] does not explicitly require linear models and \mathbf{M} smoothness implied by the linearity.

E.5 Special cases: SEGA-like methods

Let $n = 1$. Note that now operators \mathcal{S} and \mathcal{U} act on $d \times n$ matrices, i.e., on vectors in \mathbb{R}^d . To simplify notation, instead of $\mathbf{X} \in \mathbb{R}^{d \times n}$ we will write $x = (x_1, \dots, x_d) \in \mathbb{R}^d$.

E.5.1 Basic variant of SEGA [77]

Suppose that f is m -smooth (i.e., $\mathbf{M}_1 = m\mathbf{I}_d$) with $m > 0$. To recover basic SEGA from [77], consider the following choice of random operators \mathcal{S} and \mathcal{U} :

$$(\forall i) \text{ with probability } \frac{1}{d}: \quad \mathcal{S}x = e_i e_i^\top x = x_i e_i \quad \text{and} \quad \mathcal{U}x = d e_i e_i^\top x = d x_i e_i.$$

The resulting algorithm is stated as Algorithm 29.

Corollary E.5.1 (Convergence rate of SEGA). Let $\alpha = \frac{1}{4md + \mu d}$. Then the iteration complexity of Algorithm 29 is $\left(4 \frac{md}{\mu} + d \right) \log \frac{1}{\epsilon}$.

E.5.2 SEGA with arbitrary sampling

Consider a more general setup to that in Section E.5.1 and let us allow the smoothness matrix to be an arbitrary diagonal (positive semidefinite) matrix: $\mathbf{M} = \text{Diag}(m_1, \dots, m_d)$

Algorithm 29 SEGA [77]**Require:** Step size $\alpha > 0$, starting point $x^0 \in \mathbb{R}^d$ Set $h^0 = 0$ **for** $k = 0, 1, 2, \dots$ **do**Sample $i \in \{1, 2, \dots, d\}$ uniformly at randomSet $h^{k+1} = h^k + (\nabla_i f(x^k) - h_i^k) e_i$ $g^k = h^k + d(\nabla_i f(x^k) - h_i^k) e_i$ $x^{k+1} = \text{prox}_{\alpha\psi}(x^k - \alpha g^k)$ **end for**

with $m_1, \dots, m_d > 0$. In this regime, we will establish a convergence rate for an arbitrary sampling strategy, and then use this to develop importance sampling.

Let $p \in \mathbb{R}^d$ be a probability vector with entries $p_i = \mathbb{P}(i \in L)$. Consider the following choice of random operators \mathcal{S} and \mathcal{U} :

$$(\forall L) \text{ with prob. } p_L : \mathcal{S}x = \sum_{i \in L} e_i e_i^\top x = \sum_{i \in L} x_i e_i \quad \text{and} \quad \mathcal{U}x = \sum_{i \in L} \frac{1}{p_i} e_i e_i^\top x = \sum_{i \in L} \frac{x_i}{p_i} e_i. \quad (\text{E.21})$$

The resulting algorithm is stated as Algorithm 30.

Algorithm 30 SEGA with arbitrary sampling**Require:** Step size $\alpha > 0$, starting point $x^0 \in \mathbb{R}^d$, random sampling $L \subseteq \{1, 2, \dots, d\}$ Set $h^0 = 0$ **for** $k = 0, 1, 2, \dots$ **do**Sample random $L^k \subseteq \{1, 2, \dots, d\}$ Set $h^{k+1} = h^k + \sum_{i \in L^k} (\nabla_i f(x^k) - h_i^k) e_i$ $g^k = h^k + \sum_{i \in L^k} \frac{1}{p_i} (\nabla_i f(x^k) - h_i^k) e_i$ $x^{k+1} = \text{prox}_{\alpha\psi}(x^k - \alpha g^k)$ **end for**

Corollary E.5.2 (Convergence rate of SEGA). *Iteration complexity of Algorithm 30 with $\alpha = \min_i \frac{p_i}{4m_i + \mu}$ is $\max_i \left(\frac{4m_i + \mu}{p_i \mu} \right) \log \frac{1}{\epsilon}$.*

Corollary E.5.2 indicates an up to constant factor optimal choice $p_i \propto m_i$, which yields, up to a constant factor, $\frac{\sum_{i=1}^d m_i}{\mu} \log \frac{1}{\epsilon}$ complexity. In the applications where m is not unique¹, it is the best to choose one which minimizes $m^\top e$.

Remark 21. Note that if $p_i = 1$ for all i (i.e., if $\mathcal{U} = \mathcal{I}$), we recover proximal gradient descent as a special case.

¹For example when a general matrix smoothness holds; one has to upper bound it by a diagonal matrix in order to comply with the assumptions of the section. In such case, there is an infinite array of possible choices of m .

E.5.3 SVRCD with arbitrary sampling

As as a particular special case of Algorithm 14 we get a new method, which we call *Stochastic Variance Reduced Coordinate Descent* (SVRCD). The algorithm is similar to SEGA. The main difference is that SVRCD does not *update* a subset L of coordinates of vector h^k each iteration. Instead, with probability ρ , it sets h^k to $\nabla f(x^k)$.

We choose \mathcal{S} and \mathcal{U} via

$$\mathcal{S}\mathbf{X} = \begin{cases} 0 & \text{w.p. } 1 - \rho \\ \mathbf{X} & \text{w.p. } \rho \end{cases} \quad \text{and} \quad (\forall L) \quad \text{w.p. } p_L : \mathcal{U}\mathbf{X} = \sum_{i \in L} \frac{1}{p_i} \mathbf{e}_i \mathbf{e}_i^\top \mathbf{X},$$

where again $p_i = \mathbb{P}(i \in L)$. The randomness of \mathcal{S} is independent from the randomness of \mathcal{U} (which comes from the randomness of L). The resulting algorithm is stated as Algorithm 31.

Algorithm 31 SVRCD [NEW METHOD]

Require: starting point $x^0 \in \mathbb{R}^d$, random sampling $L \subseteq \{1, 2, \dots, d\}$, probability ρ , stepsize $\alpha > 0$

Set $h^0 = 0$

for $k = 0, 1, 2, \dots$ **do**

 Sample random $L^k \subseteq \{1, 2, \dots, d\}$

$$g^k = h^k + \sum_{i \in L^k} \frac{1}{p_i} (\nabla_i f(x^k) - h_i^k) \mathbf{e}_i$$

$$x^{k+1} = \text{prox}_{\alpha\psi}(x^k - \alpha g^k)$$

$$\text{Set } h^{k+1} = \begin{cases} h^k & \text{with probability } 1 - \rho \\ \nabla f(x^k) & \text{with probability } \rho \end{cases}$$

end for

As in Section E.5.2, we shall assume that f is $\mathbf{M} = \text{Diag}(m_1, \dots, m_d)$ -smooth.

Corollary E.5.3. *The iteration complexity of Algorithm 31 with $\alpha = \min_i \frac{1}{4m_i/p_i + \mu/\rho}$ is*

$$\left(\frac{1}{\rho} + \max_i \frac{4m_i}{p_i \mu} \right) \log \frac{1}{\epsilon}.$$

Corollary E.5.3 indicates optimal choice $p \propto m$.

Remark 22. If $p_i = 1$ for all i and $\rho = 1$, we recover proximal gradient descent as a special case.

E.6 Special cases: SGD-star

Suppose that $\mathbf{G}(x^*)$ is known. We will show that shifted a version of SGD-AS converges with linear rate in such case. Let $\mathbf{J}^0 = \mathbf{G}(x^*)$. Consider the following choice of random

operators \mathcal{S}, \mathcal{U} :

$$\mathcal{S}\mathbf{X} = 0 \quad \text{and} \quad (\forall R) \text{ with probability } p_R : \quad \mathcal{U}\mathbf{X} = \mathbf{X} \sum_{j \in R} \frac{1}{p_j} e_j e_j^\top.$$

The resulting algorithm is stated as Algorithm 32, which is in fact arbitrary sampling version of SGD-star from [55].

Algorithm 32 SGD-star [55]

Require: learning rate $\alpha > 0$, starting point $x^0 \in \mathbb{R}^d$, random sampling $R \subseteq \{1, 2, \dots, n\}$
for $k = 0, 1, 2, \dots$ **do**
 Sample random $R^k \subseteq \{1, 2, \dots, n\}$
 $g^k = \frac{1}{n} \mathbf{G}(x^*) e + \sum_{j \in R^k} \frac{1}{np_j} (\nabla f_j(x^k) - \nabla f_j(x^*))$
 $x^{k+1} = \text{prox}_{\alpha\psi}(x^k - \alpha g^k)$
end for

Corollary E.6.1 (Convergence rate of SGD-AS-star). *Suppose that f_j is M_j -smooth for all j and suppose that v satisfies (E.18). Let $\alpha = n \min_j \frac{p_j}{v_j}$. Then, the iteration complexity of Algorithm 32 is*

$$\max_j \left(\frac{v_j}{np_j \mu} \right) \log \frac{1}{\epsilon}.$$

Remark 23. In overparameterized models, one has $\mathbf{G}(x^*) = 0$. In such a case, Algorithm 32 becomes SGD-AS [60], and we recover its tight convergence rate.

E.7 Special cases: loopless SVRG with arbitrary sampling (LSVRG)

In this section we extend Loopless SVRG (i.e., LSVRG) from [83, 106] to arbitrary sampling. The main difference to SAGA is that LSVRG does not update \mathbf{J}^k at all with probability $1 - \rho$. However, with probability $1 - \rho$, it sets \mathbf{J}^k to $\mathbf{G}(x^k)$. Define \mathcal{S} and \mathcal{U} as follows:

$$\mathcal{S}\mathbf{X} = \begin{cases} 0 & \text{w.p. } 1 - \rho \\ \mathbf{X} & \text{w.p. } \rho \end{cases} \quad \text{and} \quad (\forall R) \text{ with probability } p_R : \mathcal{U}\mathbf{X} = \mathbf{X} \sum_{i \in R} \frac{1}{p_j} e_j e_j^\top,$$

where $p_j = \mathbb{P}(j \in R)$.

The resulting algorithm is stated as Algorithm 33.

In order to give tight rates under M -smoothness, we shall consider ESO assumption (E.18) and Assumption E.4.2 (same as for SAGA-AS).

The next corollary shows the convergence result.

Algorithm 33 LSVRG (LSVRG [83, 106] with arbitrary sampling) **[NEW METHOD]**

Require: learning rate $\alpha > 0$, starting point $x^0 \in \mathbb{R}^d$, random sampling $R \subseteq \{1, 2, \dots, n\}$
Set $\phi = x^0$
for $k = 0, 1, 2, \dots$ **do**
 Sample a random subset $R^k \subseteq \{1, 2, \dots, n\}$
 $g^k = \frac{1}{n} \sum_{j=1}^n \nabla f_j(\phi^k) + \sum_{j \in R^k} \frac{1}{np_j} (\nabla f_j(x^k) - \nabla f_j(\phi^k))$
 $x^{k+1} = \text{prox}_{\alpha\psi}(x^k - \alpha g^k)$
 Set $\phi^{k+1} = \begin{cases} x^k & \text{with probability } \rho \\ \phi^k & \text{with probability } 1 - \rho \end{cases}$
end for

Corollary E.7.1 (Convergence rate of LSVRG). *Let $\alpha = \min_j \frac{n}{4\frac{v_j}{p_j} + \frac{\mu n}{\rho}}$. Then, the iteration complexity of Algorithm 33 is*

$$\max_j \left(4 \frac{v_j}{n\mu p_j} + \frac{1}{\rho} \right) \log \frac{1}{\epsilon}.$$

Remark 24. One can consider a slightly more general setting with

$$\mathcal{S}\mathbf{X} = \begin{cases} 0 & \text{w.p. } 1 - \rho \\ \mathbf{X} \sum_{i \in R'} e_j e_j^\top & \text{w.p. } \rho \end{cases},$$

where distribution of $R' \subseteq [n]$ is arbitrary. Clearly, such methods is a special case of Algorithm 14, and setting $R' = [n]$ with probability 1, LSVRG is obtained. However, in a general form, such algorithm resembles SCSG [117]. However, unlike SCSG, the described method converges linearly, thus is superior to SCSG.

E.8 Special cases: methods with Bernoulli \mathcal{U}

Throughout this section, we will suppose that $\mathbf{M}_j = m\mathbf{I}_d$ for all j . This is sufficient to establish strong results. Indeed, Bernoulli \mathcal{U} does not allow for an efficient importance sampling and hence one can't develop arbitrary sampling results similar to those in Section E.4.2 or Section E.5.2.

E.8.1 B2 (Bernoulli \mathcal{S})

Let $n = 1$. Note that now operators \mathcal{S} and \mathcal{U} act on $d \times n$ matrices, i.e., on vectors in \mathbb{R}^d . To simplify notation, instead of $\mathbf{X} \in \mathbb{R}^{d \times n}$ we will write $x = (x_1, \dots, x_d) \in \mathbb{R}^d$.

Given probabilities $0 < \rho, \delta \leq 1$, let both \mathcal{S} and \mathcal{U} be Bernoulli (i.e., scaling) sketches:

$$\mathcal{S}x = \begin{cases} 0 & \text{w.p. } 1 - \rho \\ x & \text{w.p. } \rho \end{cases} \quad \text{and} \quad \mathcal{U}x = \begin{cases} 0 & \text{w.p. } 1 - \delta \\ \frac{1}{\delta}x & \text{w.p. } \delta \end{cases}.$$

The resulting algorithm is stated as Algorithm 34.

Algorithm 34 B2 [NEW METHOD]

Require: learning rate $\alpha > 0$, starting point $x^0 \in \mathbb{R}^d$, probabilities $\delta \in (0, 1]$ and $\rho \in (0, 1]$

Set $\phi = x^0$

for $k = 0, 1, 2, \dots$ **do**

$$g^k = \begin{cases} \nabla f(\phi^k) & \text{with probability } 1 - \delta \\ \frac{1}{\delta} \nabla f(x^k) - (\frac{1}{\delta} - 1) \nabla f(\phi^k) & \text{with probability } \delta \end{cases}$$

$$x^{k+1} = \text{prox}_{\alpha\psi}(x^k - \alpha g^k)$$

$$\text{Set } \phi^{k+1} = \begin{cases} x^k & \text{with probability } \rho \\ \phi^k & \text{with probability } 1 - \rho \end{cases}$$

end for

Corollary E.8.1 (Convergence rate B2). *Suppose that f is m -smooth. Let $\alpha = \frac{1}{4\frac{m}{\delta} + \frac{\mu}{\rho}}$. Then, the iteration complexity of Algorithm 34 is*

$$\left(4\frac{m}{\mu\delta} + \frac{1}{\rho}\right) \log \frac{1}{\epsilon}.$$

Remark 25. It is possible to choose correlated \mathcal{S} and \mathcal{U} without any sacrifice in the rate.

E.8.2 LSVRG-inv (right \mathcal{S})

Given a probability scalar $0 < \delta \leq 1$, consider choosing operators \mathcal{S} and \mathcal{U} as follows:

$$\mathcal{S}\mathbf{X} = \mathbf{X} \sum_{j \in R} e_j e_j^\top \quad \text{w.p. } p_R \quad \text{and} \quad \mathcal{U}\mathbf{X} = \begin{cases} 0 & \text{w.p. } 1 - \delta \\ \frac{1}{\delta} \mathbf{X} & \text{w.p. } \delta. \end{cases}$$

The resulting algorithm is stated as Algorithm 35.

Corollary E.8.2 (Convergence rate of LSVRG-inv). *Suppose that each f_i is m -smooth. Let $\alpha = \min_j \frac{1}{4\frac{m}{\delta} + \frac{\mu}{p_j}}$. Then, the iteration complexity of Algorithm 35 is*

$$\max_j \left(4\frac{m}{\mu\delta} + \frac{1}{p_j}\right) \log \frac{1}{\epsilon}.$$

Algorithm 35 LSVRG-inv [NEW METHOD]

Require: starting point $x^0 \in \mathbb{R}^d$, random sampling $R \subseteq \{1, 2, \dots, n\}$, probability $\delta \in (0, 1]$, learning rate $\alpha > 0$
 Set $\phi_j^0 = x^0$ for $j = 1, 2, \dots, n$
for $k = 0, 1, 2, \dots$ **do**

$$g^k = \begin{cases} \frac{1}{n} \sum_{j=1}^n \nabla f_j(\phi_j^k) & \text{with probability } 1 - \delta \\ \frac{1}{\delta} \nabla f(x^k) - \left(\frac{1}{\delta} - 1\right) \frac{1}{n} \sum_{j=1}^n \nabla f_j(\phi_j^k) & \text{with probability } \delta \end{cases}$$

 $x^{k+1} = \text{prox}_{\alpha\psi}(x^k - \alpha g^k)$
 Sample a random subset $R^k \subseteq \{1, 2, \dots, n\}$
 Set $\phi_j^{k+1} = \begin{cases} x^k & j \in R^k \\ \phi_j^k & j \notin R^k \end{cases}$
end for

E.8.3 SVRCD-inv (left \mathcal{S})

Let $n = 1$. Note that now operators \mathcal{S} and \mathcal{U} act on $d \times n$ matrices, i.e., on vectors in \mathbb{R}^d . To simplify notation, instead of $\mathbf{X} \in \mathbb{R}^{d \times n}$ we will write $x = (x_1, \dots, x_d) \in \mathbb{R}^d$.

Consider again setup where $n = 1$. Choose operators \mathcal{S} and \mathcal{U} as follows:

$$\mathcal{S}x = \sum_{i \in L} e_i e_i^\top x \quad \text{w.p. } p_L \quad \text{and} \quad \mathcal{U}x = \begin{cases} 0 & \text{w.p. } 1 - \delta \\ \frac{1}{\delta} x & \text{w.p. } \delta. \end{cases}$$

For convenience, let p be the probability vector defined as: $p_i = \mathbb{P}(i \in L)$.

The resulting algorithm is stated as Algorithm 36.

Algorithm 36 SVRCD-inv [NEW METHOD]

Require: starting point $x^0 \in \mathbb{R}^d$, random sampling $L \subseteq \{1, 2, \dots, d\}$, probability $\delta \in (0, 1]$, learning rate $\alpha > 0$
 Choose $h^0 \in \mathbb{R}^d$
for $k = 0, 1, 2, \dots$ **do**

$$g^k = \begin{cases} h^k & \text{with probability } 1 - \delta \\ \frac{1}{\delta} \nabla f(x^k) - \left(\frac{1}{\delta} - 1\right) h^k & \text{with probability } \delta \end{cases}$$

 $x^{k+1} = \text{prox}_{\alpha\psi}(x^k - \alpha g^k)$
 Sample a random subset $L^k \subseteq \{1, 2, \dots, d\}$
 Set $h^{k+1} = h^k + \sum_{i \in L^k} (\nabla_i f(x^k) - h_i^k) e_i$
end for

Corollary E.8.3 (Convergence rate of SVRCD-inv). *Suppose that each f_j is m -smooth.*

Let $\alpha = \min_i \frac{1}{4\frac{m}{\delta} + p_i}$. Then, the iteration complexity of Algorithm 36 is

$$\max_i \left(4\frac{m}{\mu\delta} + \frac{1}{p_i} \right) \log \frac{1}{\epsilon}.$$

E.9 Special cases: combination of left and right sketches

E.9.1 RL (right sampling \mathcal{S} , left unbiased sampling \mathcal{U})

Consider choosing \mathcal{S} and \mathcal{U} as follows:

$$\mathcal{S}\mathbf{X} = \mathbf{X} \sum_{j \in R} e_j e_j^\top \quad \text{w.p. } p_R \quad \text{and} \quad \mathcal{U}\mathbf{X} = \sum_{i \in L} \frac{1}{p_i} e_i e_i^\top \mathbf{X} \quad \text{w.p. } p_L.$$

The resulting algorithm is stated as Algorithm 37.

Algorithm 37 RL [NEW METHOD]

Require: starting point $x^0 \in \mathbb{R}^d$, random sampling $L \subseteq \{1, 2, \dots, d\}$, random sampling $R \subseteq \{1, 2, \dots, n\}$, learning rate $\alpha > 0$

Set $\phi_j^0 = x^0$ for each j

for $k = 0, 1, 2, \dots$ **do**

 Sample random $R^k \subseteq \{1, 2, \dots, n\}$

 Set $\phi_j^{k+1} = \begin{cases} x^k & j \in R^k \\ \phi_j^k & j \notin R^k \end{cases}$

 Sample random $L^k \subseteq \{1, 2, \dots, d\}$

$$g^k = \frac{1}{n} \sum_{j=1}^n \nabla f_j(\phi_j^k) + \sum_{i \in L^k} \frac{1}{p_i} \left(\nabla_i f(x^k) - \frac{1}{n} \sum_{j=1}^n \nabla_i f_j(\phi_j^k) \right) e_i$$

$$x^{k+1} = \text{prox}_{\alpha\psi}(x^k - \alpha g^k)$$

end for

Corollary E.9.1 (Convergence rate of RL). Suppose that each f_j is $\text{Diag}(m^j)$ -smooth, where $m^j \in \mathbb{R}^d$ and $\text{Diag}(m^j) \succ 0$. Let $\alpha = \min_{i,j} \left(4\frac{m_i^j}{p_i} + \frac{\mu}{p_j} \right)^{-1}$. Then, the iteration complexity of Algorithm 37 is

$$\max_{i,j} \left(4\frac{m_i^j}{\mu p_i} + \frac{1}{p_j} \right) \log \frac{1}{\epsilon}.$$

E.9.2 LR (left sampling \mathcal{S} , right unbiased sampling \mathcal{U})

Consider choosing \mathcal{S} and \mathcal{U} as follows:

$$\mathcal{S}\mathbf{X} = \sum_{i \in L} e_i e_i^\top \mathbf{X} \quad \text{w.p. } p_L \quad \text{and} \quad \mathcal{U}\mathbf{X} = \mathbf{X} \sum_{j \in R} \frac{1}{p_j} e_j e_j^\top \quad \text{w.p. } p_R.$$

The resulting algorithm is stated as Algorithm 38.

Algorithm 38 LR [NEW METHOD]

Require: starting point $x^0 \in \mathbb{R}^d$, random sampling $L \subseteq \{1, 2, \dots, d\}$, random sampling $R \subseteq \{1, 2, \dots, n\}$, learning rate $\alpha > 0$

Set $h^0 = x^0$ for each j

for $k = 0, 1, 2, \dots$ **do**

 Sample random $L^k \subseteq \{1, 2, \dots, d\}$

 Set $h^{k+1} = h^k + \sum_{i \in L^k} (\nabla_i f(x^k) - h_i^k) e_i$

 Sample random $R^k \subseteq \{1, 2, \dots, n\}$

$g^k = \nabla f(h^k) + \sum_{j \in R^k} \frac{1}{np_j} (\nabla f_j(x^k) - \nabla f_j(h^k))$

$x^{k+1} = \text{prox}_{\alpha\psi}(x^k - \alpha g^k)$

end for

Corollary E.9.2 (Convergence rate of LR). *Suppose that each f_j is \mathbf{M}_j -smooth, and suppose that $v \in \mathbb{R}^n$ is such that (E.18) holds. Let $\alpha = \min_{i,j} \frac{1}{4v_j p_j^{-1} + \mu p_i^{-1}}$. Then, the iteration complexity of Algorithm 38 is*

$$\max_{i,j} \left(4 \frac{v_i}{\mu p_j} + \frac{1}{p_i} \right) \log \frac{1}{\epsilon}.$$

E.10 Special cases: joint left and right sketches

E.10.1 SAEGA

Another new special case of Algorithm 14 we propose is SAEGA (the name comes from the combination of names SAGA and SEGA). In SAEGA, both \mathcal{S} and \mathcal{U} are fully correlated and consist of right and left sketch. However, the mentioned right and left sketches are independent. In particular, we have

$$\mathcal{S}\mathbf{X} = \mathbf{X}_{LR} = \left(\sum_{i \in L} e_i e_i^\top \right) \mathbf{X} \left(\sum_{j \in R} e_j e_j^\top \right),$$

where $L \subset [d]$, and $R \subset [n]$ are independent random sets. Next, \mathcal{U} is chosen as

$$\mathcal{U}\mathbf{X} = \mathcal{S} \left(\left(p^{-1} (p^{-1})^\top \right) \circ \mathbf{X} \right),$$

where $p_i = \mathbb{P}(i \in L)$ and $p_j = \mathbb{P}(j \in R)$. The resulting algorithm is stated as Algorithm 39.

Suppose that for all $j \in [n]$, $\mathbf{M}_j = \text{Diag}(m^j) \succ 0$ is diagonal matrix². Let $\mathbf{P} \in \mathbb{R}^{n \times n}$ be the probability matrix with respect to R -sampling, i.e., $\mathbf{P}_{jj'} = \mathbb{P}(j \in R, j' \in R)$.

²A block diagonal matrix \mathbf{M}_j with blocks such that $\mathcal{M}\mathcal{P}_\mathcal{S} = \mathcal{P}_\mathcal{S}\mathcal{M}$ would work as well

Algorithm 39 SAEGA [NEW METHOD]

Input: $x^0 \in \mathbb{R}^d$, random sampling $L \subseteq \{1, 2, \dots, d\}$, random sampling $R \subseteq \{1, 2, \dots, n\}$, stepsize α
 $\mathbf{J}^0 = 0$
for $k = 0, 1, 2, \dots$ **do**
 Sample random $L^k \subseteq \{1, 2, \dots, d\}$ and $R^k \subseteq \{1, 2, \dots, n\}$
 Compute $\nabla_{ij} f_j(x^k)$ for all $i \in L^k$ and $j \in R^k$
 $\mathbf{J}_{ij}^{k+1} = \begin{cases} \nabla_{ij} f_j(x^k) & i \in L^k \text{ and } j \in R^k \\ \mathbf{J}_{ij}^k & \text{otherwise} \end{cases}$
 $g^k = \left(\mathbf{J}^k + \left(\textcolor{red}{p}^{-1} (\textcolor{blue}{p}^{-1})^\top \right) \circ (\mathbf{J}^{k+1} - \mathbf{J}^k) \right) \textcolor{blue}{e}$
 $x^{k+1} = \text{prox}_{\alpha\psi}(x^k - \alpha g^k)$
end for

Corollary E.10.1. Consider any (elementwise) positive vector q such that

$$\text{Diag}(\textcolor{blue}{p})^{-1} \textcolor{blue}{P} \text{Diag}(\textcolor{blue}{p})^{-1} \preceq \text{Diag}(q)^{-1}.$$

Let $\alpha = \min_{i,j} \frac{n \textcolor{red}{p}_i \textcolor{blue}{q}_j}{4m_i^j + n\mu}$. Then, the iteration complexity of Algorithm 39 is

$$\max_{i,j} \left(4 \frac{m_i^j}{\mu n \textcolor{red}{p}_i \textcolor{blue}{q}_j} + \frac{1}{\textcolor{red}{p}_i} \frac{1}{\textcolor{blue}{q}_j} \right) \log \frac{1}{\epsilon}.$$

E.10.2 SVRCDDG

Next new special case of Algorithm 14 we propose is SVRCDDG. SVRCDDG uses the same random operator \mathcal{U} as SAEGA. The difference to SAEGA lies in operator \mathcal{S} which is Bernoulli random variable:

$$\mathcal{S}\mathbf{X} = \begin{cases} 0 & \text{w.p. } 1 - \rho \\ \mathbf{X} & \text{w.p. } \rho \end{cases}, \quad \mathcal{U}\mathbf{X} = \mathbf{I}_L \left(\left(\textcolor{red}{p}^{-1} (\textcolor{blue}{p}^{-1})^\top \right) \circ \mathbf{X} \right) \mathbf{I}_{R},$$

where $L \subseteq [d]$, and $R \subseteq [n]$ are independent random sets and $\textcolor{red}{p}_i = \mathbb{P}(i \in L)$ and $\textcolor{blue}{p}_j = \mathbb{P}(j \in R)$.

The resulting algorithm is stated as Algorithm 40.

Suppose that for all j , $\mathbf{M}_j = \text{Diag}(m^j)$ is diagonal matrix³. For notational simplicity, denote $\mathbf{M}' \in \mathbb{R}^{d \times n}$ to be the matrix with j th column equal to m_j . Let $\textcolor{blue}{P} \in \mathbb{R}^{n \times n}$ be the probability matrix with respect to R - sampling, i.e., $\textcolor{blue}{P}_{jj'} = \mathbb{P}(j \in R, j' \in R)$.

Corollary E.10.2. Consider any (elementwise) positive vector q such that

$$\text{Diag}(\textcolor{blue}{p})^{-1} \textcolor{blue}{P} \text{Diag}(\textcolor{blue}{p})^{-1} \preceq \text{Diag}(q)^{-1}.$$

³Block diagonal \mathbf{M}_j with blocks such that $\mathcal{M}\mathcal{S} = \mathcal{S}\mathcal{M}$ would work as well

Algorithm 40 SVRCDG [NEW METHOD]

Input: $x^0 \in \mathbb{R}^d$, random sampling $L \subseteq \{1, 2, \dots, d\}$, random sampling $R \subseteq \{1, 2, \dots, n\}$, stepsize α , probability ρ
 $\mathbf{J}^0 = 0$
for $k = 0, 1, 2, \dots$ **do**
 Sample random $L^k \subseteq \{1, 2, \dots, d\}$ and $R^k \subseteq \{1, 2, \dots, n\}$
 Observe $\nabla_i f_j(x^k)$ for all $i \in L^k$ and $j \in R^k$
 $g^k = \left(\mathbf{J}^k + \left(\textcolor{red}{p}^{-1} (\textcolor{blue}{p}^{-1})^\top \right) \circ \left(\mathbf{I}_{L^k} (\mathbf{G}(x^k) - \mathbf{J}^k) \mathbf{I}_{R^k} \right) \right) e$
 $x^{k+1} = \text{prox}_{\alpha\psi}(x^k - \alpha g^k)$
 $\mathbf{J}^{k+1} = \begin{cases} \mathbf{G}(x^k) & \text{with probability } \rho \\ \mathbf{J}^k & \text{with probability } 1 - \rho \end{cases}$
end for

Let $\alpha = \min_{i,j} \frac{1}{4 \frac{m_i^j}{\textcolor{red}{p}_i \textcolor{blue}{q}_j n} + \frac{1}{\rho} \mu}$. Then, the iteration complexity of Algorithm 40 is

$$\max_{i,j} \left(4 \frac{m_i^j}{\mu n \textcolor{red}{p}_i \textcolor{blue}{q}_j} + \frac{1}{\rho} \right) \log \frac{1}{\epsilon}.$$

E.10.3 ISAEGA (with distributed data)

In this section, we consider a distributed setting from [137]. In particular, [137] proposed a strategy of running coordinate descent on top of various optimization algorithms such as GD, SGD or SAGA, while keeping the convergence rate of the original method. This allows for sparse communication from workers to master.

However, ISAGA (distributed SAGA with RCD on top of it), as proposed, assumes zero gradients at the optimum which only holds for overparameterized models. It was stated as an open question whether it is possible to derive SEGA on top of it such that the mentioned assumption can be dropped. We answer this question positively, proposing ISAEGA (Algorithm 41). Next, algorithms proposed in [137] only allow for uniform sampling under simple smoothness. In contrast, we develop an arbitrary sampling strategy for general matrix smoothness⁴.

Assume that we have T parallel units, each owning set of indices N_t (for $1 \leq t \leq T$). Next, consider distributions \mathcal{D}_t over subsets of N_t and distributions \mathcal{R}_t over subsets coordinates $[d]$ for each machine. Each iteration we sample $R_t \sim \mathcal{D}_t$, $L_t \sim \mathcal{R}_t$ (for $1 \leq t \leq T$) and observe the corresponding part of Jacobian $\mathbf{J}_{\cap_t(L_t, R_t)}^k$. Thus the corresponding random Jacobian sketch becomes

$$\mathbf{S}\mathbf{X} = \mathbf{X}_{\cap_t(L_t, R_t)} = \sum_{t=1}^T \left(\sum_{i \in L_t} \textcolor{red}{e}_i \textcolor{red}{e}_i^\top \right) \mathbf{X}_{:N_t} \left(\sum_{j \in R_t} \textcolor{blue}{e}_j \textcolor{blue}{e}_j^\top \right).$$

⁴We do so only for ISAEGA. However, our framework allows obtaining arbitrary sampling results for ISAGA, ISEGA and ISGD (with no variance at optimum) as well. We omit it for space limitations

Next, for each $1 \leq t \leq T$ consider vector $\mathbf{p}^t \in \mathbb{R}^d$, $\mathbf{p}^t \in \mathbb{R}^{|N_t|}$ such that $\mathbb{P}(i \in L_t) = \mathbf{p}_i^t$ and $\mathbb{P}(j \in R_t) = \mathbf{p}_j^t$. Given the notation, random operator \mathcal{U} is chosen as

$$\mathcal{U}\mathbf{X} = \sum_{t=1}^T \left((\mathbf{p}^t)^{-1} \left((\mathbf{p}^t)^{-1} \right)^\top \right) \circ \left(\left(\sum_{i \in L_t} \mathbf{e}_i \mathbf{e}_i^\top \right) \mathbf{X}_{:N_t} \left(\sum_{j \in R_t} \mathbf{e}_j \mathbf{e}_j^\top \right) \right).$$

The resulting algorithm is stated as Algorithm 41.

Algorithm 41 ISAEGA [NEW METHOD]

Input: $x^0 \in \mathbb{R}^d$, # parallel units T , each owning set of indices N_t (for $1 \leq t \leq T$), distributions \mathcal{D}_t over subsets of N_t , distributions \mathcal{R}_t over subsets coordinates $[d]$, stepsize α
 $\mathbf{J}^0 = 0$
for $k = 0, 1, 2, \dots$ **do**
 for $t = 1, \dots, T$ in parallel **do**
 Sample $R_t \sim \mathcal{D}_t$; $R_t \subseteq N_t$ (independently on each machine)
 Sample $L_t \sim \mathcal{R}_t$; $L_t \subseteq [d]$ (independently on each machine)
 Observe $\nabla_{L_t} f_j(x^k)$ for $j \in R_t$
 For $i \in [d], j \in N_t$ set $\mathbf{J}_{i,j}^{k+1} = \begin{cases} \nabla_i f_j(x^k) & \text{if } i \in [d], j \in R_t, i \in L_t \\ \mathbf{J}_{i,j}^k & \text{otherwise} \end{cases}$
 Send $\mathbf{J}_{:N_t}^{k+1} - \mathbf{J}_{:N_t}^k$ to master ▷ Sparse; low communication
 end for
 $g^k = \left(\mathbf{J}^k + \sum_{t=1}^T \left(\mathbf{p}^{t-1} \mathbf{p}^{t-1\top} \right) \circ \left(\left(\sum_{i \in L_t} \mathbf{e}_i \mathbf{e}_i^\top \right) (\mathbf{J}^{k+1} - \mathbf{J}^k)_{:N_t} \left(\sum_{j \in R_t} \mathbf{e}_j \mathbf{e}_j^\top \right) \right) \right) \mathbf{e}$
 $x^{k+1} = \text{prox}_{\alpha\psi}(x^k - \alpha g^k)$
end for

Suppose that for all $1 \leq j \leq n$, $\mathbf{M}_j = \text{Diag}(m^j)$ is diagonal matrix⁵. Let $\mathbf{P}^t \in \mathbb{R}^{|N_t| \times |N_t|}$ be the probability matrix with respect to R_t -sampling, i.e., $\mathbf{P}_{jj'}^t = \mathbb{P}(j \in R_t, j' \in R_t)$.

Corollary E.10.3. *For all t consider any (elementwise) positive vector \mathbf{q}^t such that $\text{Diag}(\mathbf{p}^t)^{-1} \mathbf{P}^t \text{Diag}(\mathbf{p}^t)^{-1} \preceq \text{Diag}(\mathbf{q}^t)^{-1}$. Let $\alpha = \min_{j \in N_t, i, t} \frac{1}{4m_i^j \left(1 + \frac{1}{n\mathbf{p}_i^t \mathbf{q}_j^t} \right) + \frac{\mu}{\mathbf{p}_i^t \mathbf{q}_j^t}}$. Then, iteration complexity of Algorithm 41 is $\max_{j \in N_t, i, t} \left(4 \frac{m_i^j}{\mu} \left(1 + \frac{1}{n\mathbf{p}_i^t \mathbf{q}_j^t} \right) + \frac{1}{\mathbf{p}_i^t \mathbf{q}_j^t} \right) \log \frac{1}{\epsilon}$.*

Thus, for all j , it does not make sense to increase sampling size beyond point where $\mathbf{p}_i^t \mathbf{q}_j^t \geq \frac{1}{n}$ as the convergence speed would not increase significantly⁶.

Remark 26. In special case when $R_t = N_t$ always, ISAEGA becomes ISEGA from [137]. However [137] assumes that $|N_t|$ is constant in t and $L_t = \mathbf{e}_i$ with probability $\frac{1}{d}$. Thus, even special case of Corollary E.10.3 generalizes results on ISEGA from [137]. For completeness, we state ISEGA as Algorithm 42 and Corollary E.10.4 provides its iteration complexity.

Algorithm 42 ISEGA (ISEGA [137] with arbitrary sampling) **[NEW METHOD]**

Input: $x^0 \in \mathbb{R}^d$, # parallel units T , each owning set of indices N_t (for $1 \leq t \leq T$), distributions \mathcal{D}_t over subsets coordinates $[d]$, stepsize α
 $\mathbf{J}^0 = 0$
for $k = 0, 1, 2, \dots$ **do**
 for $t = 1, \dots, T$ in parallel **do**
 Sample $L_t \sim \mathcal{D}_t$; $L_t \subseteq [d]$ (independently on each machine)
 Observe $\nabla_{L_t} f_j(x^k)$ for $j \in N_t$
 For $i \in [d], j \in N_t$ set $\mathbf{J}_{i,j}^{k+1} = \begin{cases} \nabla_i f_j(x^k) & \text{if } i \in [d], j \in N_t, i \in L_t \\ \mathbf{J}_{i,j}^k & \text{otherwise} \end{cases}$
 Send $\mathbf{J}_{:N_t}^{k+1} - \mathbf{J}_{:N_t}^k$ to master ▷ Sparse; low communication
 end for
 $g^k = \left(\mathbf{J}^k + \sum_{t=1}^T \left(p^{t-1} e^\top \right) \circ \left(\left(\sum_{i \in L_t} e_i e_i^\top \right) (\mathbf{J}^{k+1} - \mathbf{J}^k)_{:N_t} \right) \right) e$
 $x^{k+1} = \text{prox}_{\alpha\psi}(x^k - \alpha g^k)$
end for

Corollary E.10.4. Let $\alpha = \min_{j \in N_t, i, t} \frac{1}{4m_i^j \left(1 + \frac{1}{np_i^t |N_t|} \right) + \frac{\mu}{p_i^t |N_t|}}$. Then, iteration complexity of Algorithm 41 is $\max_{j \in N_t, i, t} \left(4 \frac{m_i^j}{\mu} \left(1 + \frac{1}{np_i^t |N_t|} \right) + \frac{1}{p_i^t |N_t|} \right) \log \frac{1}{\epsilon}$.

E.11 Special cases: JacSketch

As next special case of GJS (Algorithm 14) we present JacSketch (JS) motivated by [65]. The algorithm observes every iteration a single right sketch of the Jacobian and constructs operators \mathcal{S}, \mathcal{U} in the following fashion:

$$\mathcal{S}\mathbf{X} = \mathbf{X}\mathbf{R} \quad \text{and} \quad \mathcal{U}\mathbf{X} = \mathbf{X}\mathbf{R}\mathbb{E}[\mathbf{R}]^{-1}$$

where $\mathbf{R} \in \mathbb{R}^{n \times n}$ is random projection matrix.

Algorithm 43 JS (JacSketch)

1: **Parameters:** Stepsize $\alpha > 0$, Distribution \mathcal{D} over random projector matrices $\mathbf{R} \in \mathbb{R}^{n \times n}$
2: **Initialization:** Choose solution estimate $x^0 \in \mathbb{R}^d$ and Jacobian estimate $\mathbf{J}^0 \in \mathbb{R}^{d \times n}$
3: **for** $k = 0, 1, 2, \dots$ **do**
4: Sample realization of $\mathbf{R} \sim \mathcal{D}$ perform sketches $\mathbf{G}(x^k)\mathbf{R}$
5: $\mathbf{J}^{k+1} = \mathbf{J}^k - (\mathbf{J}^k - \mathbf{G}(x^k)\mathbf{R})$
6: $g^k = \frac{1}{n} \mathbf{J}^k e + \frac{1}{n} (\mathbf{G}(x^k) - \mathbf{J}^k) \mathbf{R} \mathbb{E}[\mathbf{R}]^{-1} e$
7: $x^{k+1} = \text{prox}_{\alpha\psi}(x^k - \alpha g^k)$
8: **end for**

⁵block diagonal \mathbf{M}_j with blocks such that $\mathcal{M}\mathcal{S} = \mathcal{S}\mathcal{M}$ would work as well

⁶For indices i, j, t which maximize the rate from Corollary E.10.3.

Note that Algorithm 43 differs to what was proposed in [65] in the following points.

- Approach from [65] uses a scalar random variable $\theta_{\mathbf{R}}$ to set $\mathcal{U}\mathbf{X} = \theta_{\mathbf{R}}\mathbf{X}\mathbf{R}$. Instead, we set $\mathbb{E}[\mathcal{U}] = \mathbf{X}\mathbf{R}\mathbb{E}[\mathbf{R}]^{-1}$. This tweak allows Algorithm 14 to recover the tightest known analysis of SAGA as a special case. Note that the approach from [65] only recovers tight rates for SAGA under uniform sampling.
- Unlike [65], our setup allows for proximable regularizer, thus is more general.
- Approach from [65] allows projections under a general weighted norm. Algorithm 14 only allows for non-weighted norm; which is only done for the sake of simplicity as the chapter is already very notation-heavy. However, GJS (Algorithm 14) is general enough to allow for an arbitrary weighted norm.

The next corollary shows the convergence result.

Corollary E.11.1 (Convergence rate of JacSketch). *Suppose that operator \mathcal{M} is commutative with right multiplication by \mathbf{R} always. Consider any $\mathbf{B} \in \mathbb{R}^{n \times n}$ which commutes with \mathbf{R} always. Denote*

$$\mathbf{M}^{\frac{1}{2}} \stackrel{\text{def}}{=} \begin{pmatrix} \mathbf{M}_1^{\frac{1}{2}} & & \\ & \ddots & \\ & & \mathbf{M}_n^{\frac{1}{2}} \end{pmatrix}$$

and

$$\eta \stackrel{\text{def}}{=} \lambda_{\max} \left(\mathbf{M}^{\frac{1}{2}^\top} \left(\mathbb{E} [\mathbf{R}\mathbb{E}[\mathbf{R}]^{-1} \textcolor{blue}{e}\textcolor{blue}{e}^\top \mathbb{E}[\mathbf{R}]^{-1} \mathbf{R}] \otimes \mathbf{I}_d \right) \mathbf{M}^{\frac{1}{2}} \right),$$

and let

$$\alpha = \frac{\lambda_{\min} (\mathbf{B}^\top \mathbb{E}[\mathbf{R}] \mathbf{B})}{4n^{-1}\eta\lambda_{\max} (\mathbf{B}^\top \mathbb{E}[\mathbf{R}] \mathbf{B}) + \mu\lambda_{\max} (\mathbf{B}^\top \mathbf{B})}.$$

Then, the iteration complexity of Algorithm 43 is

$$\frac{4n^{-1}\eta\mu^{-1}\lambda_{\max} (\mathbf{B}^\top \mathbb{E}[\mathbf{R}] \mathbf{B}) + \lambda_{\max} (\mathbf{B}^\top \mathbf{B})}{\lambda_{\min} (\mathbf{B}^\top \mathbb{E}[\mathbf{R}] \mathbf{B})} \log \frac{1}{\epsilon}.$$

E.12 Special cases: proofs

In this section, we provide the proofs of all corollaries listed in previous sections. For simplicity, we will use the following notation throughout this section: $\Gamma(\mathbf{X}) = \mathcal{U}(\mathbf{X})\textcolor{blue}{e}$.

E.12.1 SAGA methods: proofs

Setup for Corollary E.4.1

Note first that the choice of \mathcal{S}, \mathcal{U} yields

$$\begin{aligned} \mathbb{E}[\mathcal{S}(\mathbf{X})] &= \frac{1}{n}\mathbf{X}, \\ \mathbb{E}[\|\Gamma(\mathbf{X})\|^2] &= n^2\mathbb{E}[\langle \mathbf{X}^\top, \textcolor{blue}{e}_j\textcolor{blue}{e}_j^\top \textcolor{blue}{e}\textcolor{blue}{e}^\top \textcolor{blue}{e}_j\textcolor{blue}{e}_j^\top \mathbf{X}^\top \rangle] = n\|\mathbf{X}\|^2. \end{aligned}$$

Next, as we have no prior knowledge about $\mathbf{G}(x^*)$, let $\mathcal{R} \equiv \mathcal{I}$; i.e. $\text{Range}(\mathcal{R}) = \mathbb{R}^{d \times n}$. Lastly, consider \mathcal{B} operator to be a multiplication with constant β : $\mathcal{B}(\mathbf{X}) = \beta \mathbf{X}$. Thus, for (5.12) we should have

$$\frac{2\alpha}{n}m + \beta^2 \left(1 - \frac{1}{n}\right) \leq (1 - \alpha\mu)\beta^2$$

and for (5.13) we should have

$$\frac{2\alpha}{n}m + \frac{\beta^2}{n} \leq \frac{1}{n}.$$

It remains to notice that choices $\alpha = \frac{1}{4m + \mu n}$ and $\beta^2 = \frac{1}{2}$ are valid to satisfy the above bounds.

Setup for Corollary E.4.3

First note that $\mathbb{E}[\mathcal{S}(\mathbf{X})] = \mathbf{X} \text{Diag}(\mathbf{p})$. Next, due to (E.20), (E.19), inequalities (5.12) and (5.13) with choice $\mathbf{Y} = \mathcal{M}^{\frac{1}{2}} \mathbf{X}$ become respectively:

$$\frac{2\alpha}{n^2} \mathbb{E} \left[\left\| \sum_{j \in R} p_j^{-1} \mathbf{M}_j^{\frac{1}{2}} \mathbf{Y}_{:j} \right\|^2 \right] + \left\| (\mathcal{I} - \mathbb{E}[\mathcal{S}])^{\frac{1}{2}} \mathcal{B}(\mathbf{Y}) \right\|^2 \leq (1 - \alpha\mu) \|\mathcal{B}(\mathbf{Y})\|^2 \quad (\text{E.22})$$

$$\frac{2\alpha}{n^2} \mathbb{E} \left[\left\| \sum_{j \in R} p_j^{-1} \mathbf{M}_i^{\frac{1}{2}} \mathbf{Y}_{:i} \right\|^2 \right] + \left\| (\mathbb{E}[\mathcal{S}])^{\frac{1}{2}} \mathcal{B}(\mathbf{Y}) \right\|^2 \leq \frac{1}{n} \|\mathbf{Y}\|^2 \quad (\text{E.23})$$

Note that

$$\mathbb{E} \left[\left\| \sum_{j \in R} p_j^{-1} \mathbf{M}_j^{\frac{1}{2}} \mathbf{Y}_{:j} \right\|^2 \right] = \mathbb{E} \left[\left\| \sum_{j \in R} \mathbf{M}_j^{\frac{1}{2}} (p_j^{-1} \mathbf{Y}_{:j}) \right\|^2 \right] \leq \sum_{j=1}^n p_j^{-1} v_j \|\mathbf{Y}_{:j}\|^2,$$

where we used ESO assumption (E.18) in the last bound above. Next, choose \mathcal{B} to be right multiplication with $\text{Diag}(b)$. Thus, for (E.22) it suffices to have for all $j \in [n]$

$$\frac{2\alpha}{n^2} v_j p_j^{-1} + b_j^2 (1 - p_j) \leq b_j^2 (1 - \alpha\mu) \quad \Rightarrow \quad \frac{2\alpha}{n^2} v_j p_j^{-1} + b_j^2 \alpha\mu \leq b_j^2 p_j$$

For (E.23) it suffices to have for all $j \in [n]$

$$\frac{2\alpha}{n^2} v_j p_j^{-1} + b_j^2 p_j \leq \frac{1}{n}$$

It remains to notice that the choice $b_j^2 = \frac{1}{2n p_j}$ and $\alpha = \min_j \frac{n p_j}{4v_j + n\mu}$ is valid.

E.12.2 SEGA methods: proofs

Setup for Corollary E.5.1

Note that

$$\begin{aligned}\mathbb{E}[\mathcal{S}x] &= \frac{1}{d}x, \\ \mathbb{E}[\|\Gamma(x)\|^2] &= d^2\mathbb{E}[\langle x, e_i e_i^\top e_i e_i^\top x \rangle] = d\|x\|^2.\end{aligned}$$

Next, choose operator \mathcal{B} to be constant; in particular $\mathcal{B}x = \beta x$. Thus to satisfy (5.12) it suffices to have

$$2\alpha dm + \beta^2 \left(1 - \frac{1}{d}\right) \leq \beta^2(1 - \alpha\mu) \quad \Rightarrow \quad 2\alpha dm + \alpha\mu\beta^2 \leq \frac{\beta^2}{d}.$$

To satisfy (5.13), it suffices to have

$$2\alpha dm + \frac{\beta^2}{d} \leq 1.$$

It remains to notice that $\beta^2 = \frac{d}{2}$ and $\alpha = \frac{1}{4md+\mu d}$ satisfies the above conditions.

Setup for Corollary E.5.2

Note that $\mathbb{E}[\mathcal{S}(x)] = \text{Diag}(\mathbf{p})x$ and

$$\mathbb{E}[\|\Gamma(x)\|^2] = \|x\|_{\mathbb{E}[\sum_{i \in L} \frac{1}{p_i} e_i e_i^\top \sum_{i \in L} \frac{1}{p_i} e_i e_i^\top]}^2 = \|x\|_{\mathbf{p}^{-1}}^2.$$

Let us consider \mathcal{B} to be the operator corresponding to left multiplication with matrix $\text{Diag}(b)$: $\mathcal{B}(x) = \text{Diag}(b)x$. Thus, for (5.12) it suffices to have for all i

$$2\alpha m_i p_i^{-1} + b_i^2(1 - p_i) \leq b_i^2(1 - \alpha\mu) \quad \Rightarrow \quad 2\alpha m_i p_i^{-1} + b_i^2\alpha\mu \leq b_i^2 p_i.$$

For (5.13) it suffices to have for all i

$$2\alpha m_i p_i^{-1} + b_i^2 p_i \leq 1$$

It remains to notice that choice $b_i^2 = \frac{1}{2p_i}$ and $\alpha = \min_i \frac{p_i}{4m_i + \mu}$ is valid.

Setup for Corollary E.5.3

Note that $\mathbb{E}[\mathcal{S}(x)] = \rho x$ and

$$\mathbb{E}[\|\Gamma(x)\|^2] = \|x\|_{\mathbb{E}[\sum_{i \in L} \frac{1}{p_i} e_i e_i^\top \sum_{i \in L} \frac{1}{p_i} e_i e_i^\top]}^2 = \|x\|_{\mathbf{p}^{-1}}^2.$$

Let us consider \mathcal{B} to be the operator corresponding to scalar multiplication with β .

Thus, for (5.12) it suffices to have for all i

$$2\alpha w_i p_i^{-1} + \beta^2(1 - \rho) \leq \beta^2(1 - \alpha\mu) \quad \Rightarrow \quad 2\alpha w_i p_i^{-1} + \beta^2\alpha\mu \leq \beta^2\rho.$$

For (5.13) it suffices to have for all i ,

$$2\alpha w_i p_i^{-1} + \beta^2\rho \leq 1.$$

It remains to notice that choice $\beta^2 = \frac{1}{2\rho}$ and $\alpha = \min_i \frac{1}{4w_i p_i^{-1} + \mu\rho^{-1}}$ is valid.

E.12.3 Setup for Corollary E.6.1

Choose \mathcal{B} to be operator which maps everything into 0. On top of that, by construction we have $\mathcal{R} = 0$ and thus (5.12) is satisfied for free. Moreover, from (E.18) we have (following the steps from Section E.12.1):

$$\mathbb{E} \left[\|\Gamma(\mathcal{M}^{\frac{1}{2}}(\mathbf{X}))\|^2 \right] \leq \sum_{j=1}^n p_j^{-1} v_j \|\mathbf{X}_{:j}\|^2.$$

Further, due to (E.19) and (E.20), to satisfy (5.13) we shall have

$$\frac{2\alpha}{n^2} \sum_{j=1}^n p_j^{-1} v_j \|\mathbf{Y}_{:j}\|^2 \leq \frac{1}{n} \|\mathbf{Y}\|^2,$$

which simplifies to

$$\frac{2\alpha}{n} \frac{v_j}{p_j} \leq 1$$

and thus it suffices to choose $\alpha = \frac{n}{2} \min_j \frac{p_j}{v_j}$.

Remark 27. Factor 2 can be omitted since for Lemma E.2.6, the second factor is 0 and thus we no longer need the Jensen's inequality.

E.12.4 Setup for Corollary E.7.1

First note that $\mathbb{E}[\mathcal{S}(\mathbf{X})] = \rho\mathbf{X}$. Next, due to (E.20), (E.19), inequalities (5.12) and (5.13) with choice $\mathbf{Y} = \mathcal{M}^{\dagger\frac{1}{2}}\mathbf{X}$ become respectively:

$$\frac{2\alpha}{n^2} \mathbb{E} \left[\left\| \sum_{j \in R} p_j^{-1} \mathbf{M}_j^{\frac{1}{2}} \mathbf{Y}_{:j} \right\|^2 \right] + \left\| (\mathcal{I} - \mathbb{E}[\mathcal{S}])^{\frac{1}{2}} \mathcal{B}(\mathbf{Y}) \right\|^2 \leq (1 - \alpha\mu) \|\mathcal{B}(\mathbf{Y})\|^2, \quad (\text{E.24})$$

and

$$\frac{2\alpha}{n^2} \mathbb{E} \left[\left\| \sum_{j \in R} p_j^{-1} \mathbf{M}_i^{\frac{1}{2}} \mathbf{Y}_{:i} \right\|^2 \right] + \left\| (\mathbb{E}[\mathcal{S}])^{\frac{1}{2}} \mathcal{B}(\mathbf{Y}) \right\|^2 \leq \frac{1}{n} \|\mathbf{Y}\|^2. \quad (\text{E.25})$$

Note next that

$$\mathbb{E} \left[\left\| \sum_{j \in R} p_j^{-1} \mathbf{M}_j^{\frac{1}{2}} \mathbf{Y}_{:j} \right\|^2 \right] = \mathbb{E} \left[\left\| \sum_{j \in R} \mathbf{M}_j^{\frac{1}{2}} (p_j^{-1} \mathbf{Y}_{:j}) \right\|^2 \right] \leq \sum_{j=1}^n p_j^{-1} v_j \|\mathbf{Y}_{:j}\|^2,$$

where we used ESO assumption (E.18) in the last bound above. Next, choose \mathcal{B} to be multiplication with scalar β . Thus, for (E.24) it suffices to have for all $j \in [n]$

$$\frac{2\alpha}{n^2} v_j p_j^{-1} + \beta^2(1 - \rho) \leq \beta^2(1 - \alpha\mu) \quad \Rightarrow \quad \frac{2\alpha}{n^2} v_j p_j^{-1} + \beta^2 \alpha\mu \leq \beta^2 \rho$$

For (E.25) it suffices to have for all $j \in [n]$

$$\frac{2\alpha}{n^2} v_j p_j^{-1} + \beta^2 \rho \leq \frac{1}{n}$$

It remains to notice that choice $\beta^2 = \frac{1}{2n\rho}$ and $\alpha = \min_j \frac{n}{4v_j p_j^{-1} + n\mu\rho^{-1}}$ is valid.

E.12.5 Methods with Bernoulli \mathcal{U} : proofs

Setup for Corollary E.8.1

Note first that the choice of \mathcal{S}, \mathcal{U} yield

$$\begin{aligned} \mathbb{E}[\mathcal{S}(x)] &= \rho x, \\ \mathbb{E}[\|\Gamma(x)\|^2] &= \mathbb{E}[\|\mathcal{U}x\|^2] = \delta^{-1} \|x\|^2. \end{aligned}$$

Next, consider \mathcal{B} operator to be a multiplication with a constant b .

Thus for (5.12) we should have

$$2\alpha\delta^{-1}L + b^2(1 - \rho) \leq (1 - \alpha\mu)b^2$$

and for (5.13) we should have

$$2\alpha\delta^{-1}L + \rho b^2 \leq 1.$$

It remains to notice that choices $\alpha = \frac{1}{4\delta^{-1}L + \mu\rho^{-1}}$ and $b^2 = \frac{1}{2\rho}$ are valid to satisfy the above bounds.

Setup for Corollary E.8.2

Note first that the choice of \mathcal{S}, \mathcal{U} yields

$$\begin{aligned} \mathbb{E}[\mathcal{S}(\mathbf{X})] &= \mathbf{X} \text{Diag}(\mathbf{p}) \\ \mathbb{E}[\|\Gamma(\mathbf{X})\|^2] &= \mathbb{E}[\|\mathcal{U}(\mathbf{X})\mathbf{e}\|^2] = \delta^{-1} \|\mathbf{X}\mathbf{e}\|^2 \leq \delta^{-1} n \|\mathbf{X}\|^2 \end{aligned}$$

Next, as we have no prior knowledge about $\mathbf{G}(x^*)$, consider \mathcal{R} to be identity operator;

i.e. $\text{Range}(\mathcal{R}) = \mathbb{R}^{d \times n}$. Lastly, consider \mathcal{B} operator to be a right multiplication with $\text{Diag}(b)$.

Thus for (5.12) we should have

$$\forall j : \quad \frac{2\alpha}{n} \delta^{-1} m + \alpha \mu b_j^2 \leq b_j^2 p_j$$

and for (5.13) we should have

$$\forall j : \quad \frac{2\alpha}{n} \delta^{-1} m + p_j b_j^2 \leq \frac{1}{n}$$

It remains to notice that choices $\alpha = \min_j \frac{1}{4\delta^{-1}m + \mu p_j^{-1}}$ and $b_j^2 = \frac{1}{2np_j}$ are valid to satisfy the above bounds.

Setup for Corollary E.8.3

Note first that the choice of \mathcal{S}, \mathcal{U} yields

$$\begin{aligned} \mathbb{E}[\mathcal{S}(x)] &= p \circ x, \\ \mathbb{E}[\|\Gamma(x)\|^2] &= \mathbb{E}[\|\mathcal{U}(x)\|^2] = \delta^{-1} \|x\|^2. \end{aligned}$$

Next, as we have no prior knowledge about $\mathbf{G}(x^*)$, consider \mathcal{R} to be identity operator; i.e. $\text{Range}(\mathcal{R}) = \mathbb{R}^{d \times n}$. Lastly, consider \mathcal{B} operator to be left multiplication with matrix $\text{Diag}(b)$.

Thus, for (5.12) we should have $2\alpha \delta^{-1} m + b_i^2 \alpha \mu \leq b_i^2 p_i$ for all i , and for (5.13), we should have $2\alpha \delta^{-1} m + p_i b_i^2 \leq 1$.

It remains to notice that the choices $\alpha = \min_i \frac{1}{4\delta^{-1}m + \mu p_i^{-1}}$ and $b_i^2 = \frac{1}{2p_i^{-1}}$ are valid to satisfy the above bounds.

E.12.6 Combination of left and right sketches: proofs

Setup for Corollary E.9.1

Note first that the choice of \mathcal{S}, \mathcal{U} yields $\mathbb{E}[\mathcal{S}(\mathbf{X})] = \mathbf{X} \text{Diag}(p)$ and

$$\mathbb{E}[\|\Gamma(\mathbf{X})\|^2] = \|\mathcal{M}^{\frac{1}{2}}(\mathbf{X}) e\|_{\text{Diag}(p^{-1})}^2 \leq n \sum_{j=1}^n \|\mathbf{M}_j \mathbf{X}_{:,j}\|_{\text{Diag}(p^{-1})}^2 = n \sum_{j=1}^n \|\mathbf{X}_{:,j}\|_{\text{Diag}(m^j \circ p^{-1})}^2.$$

Let \mathcal{B} be right multiplication by $\text{Diag}(b)$. Thus for (5.12) we should have

$$\forall i, j : \quad 2 \frac{\alpha}{n} m_i^j p_i^{-1} + b_j^2 \alpha \mu \leq b_j^2 p_j$$

and for (5.13) we should have

$$\forall i, j : \quad 2 \frac{\alpha}{n} m_i^j p_i^{-1} + p_j b_j^2 \leq \frac{1}{n}.$$

It remains to notice that choices $\alpha = \min_{i,j} \frac{1}{4m_i^j p_i^{-1} + \mu p_j^{-1}}$ and $b_j^2 = \frac{1}{2p_j n}$ are valid to satisfy the above bounds.

Setup for Corollary E.9.2

Note first that the choice of \mathcal{S}, \mathcal{U} yields

$$\begin{aligned} \mathbb{E}[\mathcal{S}(\mathbf{X})] &= \mathbf{Diag}(\mathbf{p})\mathbf{X} \\ \mathbb{E}[\|\Gamma(\mathbf{X})\|^2] &\leq \sum_{j=1}^n p_j^{-1} v_j \|\mathbf{X}_{:,j}\|^2 \end{aligned}$$

The second inequality is a direct consequence of ESO (which is shown in Section E.12.1).

Let \mathcal{B} be left multiplication by $\mathbf{Diag}(b)$. Thus for (5.12) we should have

$$\forall i, j : \quad 2\frac{\alpha}{n} v_j p_j^{-1} + b_i^2 \alpha \mu \leq b_i^2 p_i$$

and for (5.13) we should have

$$\forall i, j : \quad 2\frac{\alpha}{n} v_j p_j^{-1} + p_i b_i^2 \leq \frac{1}{n}.$$

It remains to notice that choices $\alpha = \min_{i,j} \frac{1}{4v_j p_j^{-1} + \mu p_i^{-1}}$ and $b_i^2 = \frac{1}{2p_i n}$ are valid to satisfy the above bounds.

E.12.7 Joint sketches: proofs

Setup for Corollary E.10.1

For notational simplicity, denote $\mathbf{M}'^{\frac{1}{2}} \in \mathbb{R}^{d \times n}$ to be the matrix with j th column equal to (elementwise) square root of m_j . We have

$$\mathbb{E}[\mathcal{S}(\mathbf{X})] = (\mathbf{p}\mathbf{p}^\top) \circ \mathbf{X}$$

and

$$\begin{aligned}
& \mathbb{E} \left[\left\| \Gamma(\mathcal{M}^{\frac{1}{2}} \mathbf{X}) \right\|^2 \right] \\
&= \mathbb{E} \left[\left\| \left(\left(\mathbf{p}^{-1} \mathbf{p}^{-1\top} \right) \circ \left(\left(\sum_{i \in L} \mathbf{e}_i \mathbf{e}_i^\top \right) (\mathbf{M}'^{\frac{1}{2}} \circ \mathbf{X}) \left(\sum_{j \in R} \mathbf{e}_j \mathbf{e}_j^\top \right) \right) \right) \mathbf{e} \right\|^2 \right] \\
&= \mathbb{E} \left[\left\| \left(\left(\sum_{i \in L, j \in R} \mathbf{e}_i \mathbf{e}_j^\top \right) \circ \left(\mathbf{p}^{-1} \mathbf{p}^{-1\top} \right) \circ \mathbf{M}'^{\frac{1}{2}} \circ \mathbf{X} \right) \mathbf{e} \right\|^2 \right] \\
&= \mathbb{E} \left[\left\| \left(\sum_{i \in L} \mathbf{e}_i \mathbf{e}_i^\top \right) \left(\left(\mathbf{p}^{-1} \mathbf{p}^{-1\top} \right) \circ \mathbf{M}'^{\frac{1}{2}} \circ \mathbf{X} \right) \mathbf{e}_R \right\|^2 \right] \\
&= \mathbb{E} \left[\left\| \left(\left(\mathbf{p}^{-\frac{1}{2}} \mathbf{p}^{-1\top} \right) \circ \mathbf{M}'^{\frac{1}{2}} \circ \mathbf{X} \right) \mathbf{e}_R \right\|^2 \right] \\
&= \mathbb{E} \left[\text{Tr} \left(\left(\left(\mathbf{p}^{-\frac{1}{2}} \mathbf{p}^{-1\top} \right) \circ \mathbf{M}'^{\frac{1}{2}} \circ \mathbf{X} \right) \mathbf{I}_{R,R} \left(\left(\mathbf{p}^{-\frac{1}{2}} \mathbf{p}^{-1\top} \right)^\top \circ \mathbf{M}'^{\frac{1}{2}\top} \circ \mathbf{X}^\top \right) \right) \right] \\
&= \text{Tr} \left(\left(\left(\mathbf{p}^{-\frac{1}{2}} \mathbf{p}^{-1\top} \right) \circ \mathbf{M}'^{\frac{1}{2}} \circ \mathbf{X} \right) \mathbf{P} \left(\left(\mathbf{p}^{-\frac{1}{2}} \mathbf{p}^{-1\top} \right)^\top \circ \mathbf{M}'^{\frac{1}{2}\top} \circ \mathbf{X}^\top \right) \right) \\
&= \text{Tr} \left(\left(\left(\mathbf{p}^{-\frac{1}{2}} \mathbf{e}^\top \right) \circ \mathbf{M}'^{\frac{1}{2}} \circ \mathbf{X} \right) \text{Diag}(\mathbf{p})^{-1} \mathbf{P} \text{Diag}(\mathbf{p})^{-1} \left(\left(\mathbf{p}^{-\frac{1}{2}} \mathbf{e}^\top \right)^\top \circ \mathbf{M}'^{\frac{1}{2}\top} \circ \mathbf{X}^\top \right) \right) \\
&\leq \text{Tr} \left(\left(\left(\mathbf{p}^{-\frac{1}{2}} \mathbf{e}^\top \right) \circ \mathbf{M}'^{\frac{1}{2}} \circ \mathbf{X} \right) \text{Diag}(\mathbf{q})^{-1} \left(\left(\mathbf{p}^{-\frac{1}{2}} \mathbf{e}^\top \right)^\top \circ \mathbf{M}'^{\frac{1}{2}\top} \circ \mathbf{X}^\top \right) \right) \\
&= \left\| \mathbf{X} \circ \mathbf{M}'^{\frac{1}{2}} \circ \left(\mathbf{p}^{-\frac{1}{2}} \mathbf{q}^{-\frac{1}{2}\top} \right) \right\|^2. \tag{E.26}
\end{aligned}$$

Next, choose operator \mathbf{B} to be such that $\mathcal{B}(\mathbf{X}) \stackrel{\text{def}}{=} \mathbf{B} \circ \mathbf{X}$ for $\mathbf{B} \in \mathbb{R}^{d \times n}$. Thus, for (5.12) and (5.13) we shall have respectively

$$\forall i, j : \quad \frac{2\alpha}{n^2} \left(\frac{m_i^j}{\mathbf{p}_i \mathbf{q}_j} \right) + \mathbf{B}_{i,j}^2 \alpha \mu \leq \mathbf{B}_{i,j}^2 \mathbf{p}_i \mathbf{q}_j$$

and

$$\forall i, j : \quad \frac{2\alpha}{n^2} \left(\frac{m_i^j}{\mathbf{p}_i \mathbf{q}_j} \right) + \mathbf{B}_{i,j}^2 \mathbf{p}_i \mathbf{q}_j \leq \frac{1}{n}.$$

It remains to choose $\mathbf{B}_{i,j}^2 = \frac{1}{2n \mathbf{p}_i \mathbf{q}_j}$ and $\alpha = \min_{i,j} \frac{n \mathbf{p}_i \mathbf{q}_j}{4m_i^j + n\mu}$.

Setup for Corollary E.10.2

We have

$$\mathbb{E}[\mathcal{S}(\mathbf{X})] = \rho \mathbf{X}.$$

Next, choose operator \mathbf{B} to be such that $\mathcal{B}(\mathbf{X}) \stackrel{\text{def}}{=} \beta \circ \mathbf{X}$ for scalar β which would

be specified soon. Proceeding with bound (E.26), for (5.12) and (5.13) we shall have respectively

$$\forall i, j : \quad \frac{2\alpha}{n^2} \left(\frac{m_i^j}{\mathbf{p}_i \mathbf{q}_j} \right) + \beta^2 \alpha \mu \leq \beta^2 \rho$$

and

$$\forall i, j : \quad \frac{2\alpha}{n^2} \left(\frac{m_i^j}{\mathbf{p}_i \mathbf{q}_j} \right) + \beta^2 \rho \leq \frac{1}{n}.$$

It remains to choose $\beta^2 = \frac{1}{2n\rho}$ and $\alpha = \min_{i,j} \frac{1}{4 \frac{m_i^j}{n \mathbf{p}_i \mathbf{q}_j} + \rho^{-1} \mu}$.

Setup for Corollary E.10.3

For notational simplicity, denote $\mathbf{M}' \in \mathbb{R}^{d \times n}$ to be a matrix with j th column equal to m_j .

Let $\Gamma_t(\mathbf{X}_{:N_t}) = \left(\mathbf{p}^{t-1} \mathbf{p}^{t-1\top} \right) \circ \left(\left(\sum_{i \in L_t} \mathbf{e}_i \mathbf{e}_i^\top \right) \mathbf{X}_{:N_t} \left(\sum_{j \in R_t} \mathbf{e}_j \mathbf{e}_j^\top \right) \right) \mathbf{e}_{N_t}$. Thus

$$\mathbb{E} [\mathcal{S}(\mathbf{X})] = \sum_{t=1}^T \left(\mathbf{p}^t \mathbf{p}^{t\top} \right) \circ \mathbf{X}_{:N_t}$$

and

$$\mathbb{E} [\|\Gamma(\mathbf{X})\|^2] = \mathbb{E} \left[\left\| \sum_{t=1}^T \Gamma_t(\mathbf{X}_{:N_t}) \right\|^2 \right] \quad (\text{E.27})$$

$$\begin{aligned} &= \mathbb{E} \left[\left\| \sum_{t=1}^T \Gamma_t(\mathbf{X}_{:N_t}) - \mathbb{E} \left[\sum_{t=1}^T \Gamma_t(\mathbf{X}_{:N_t}) \right] \right\|^2 \right] + \left\| \mathbb{E} \left[\sum_{t=1}^T \Gamma_t(\mathbf{X}_{:N_t}) \right] \right\|^2 \\ &= \mathbb{E} \left[\left\| \sum_{t=1}^T (\Gamma_t(\mathbf{X}_{:N_t}) - \mathbf{X}_{:N_t} \mathbf{e}_{N_t}) \right\|^2 \right] + \|\mathbf{X} \mathbf{e}\|^2 \\ &= \sum_{t=1}^T \mathbb{E} [\|\Gamma_t(\mathbf{X}_{:N_t}) - \mathbf{X}_{:N_t} \mathbf{e}_{N_t}\|^2] + \|\mathbf{X} \mathbf{e}\|^2 \\ &\leq \sum_{t=1}^T \mathbb{E} [\|\Gamma_t(\mathbf{X}_{:N_t})\|^2] + \|\mathbf{X} \mathbf{e}\|^2 \\ &\leq \sum_{t=1}^T \mathbb{E} [\|\Gamma_t(\mathbf{X}_{:N_t})\|^2] + n \|\mathbf{X}\|^2. \end{aligned} \quad (\text{E.28})$$

Using the bounds from Section E.12.7 we further get

$$\mathbb{E} [\|\Gamma(\mathcal{M}^{\frac{1}{2}} \mathbf{X})\|^2] \stackrel{(\text{E.28})+(\text{E.26})}{\leq} \sum_{t=1}^T \left\| \mathbf{X}_{:N_t} \circ \left(\mathbf{p}^{t-\frac{1}{2}} \mathbf{q}^{t-\frac{1}{2}\top} \right) \circ \mathbf{M}'_{:N_t} \right\|^2 + n \|\mathbf{M}' \circ \mathbf{X}\|^2.$$

Next, choose operator \mathbf{B} to be such that for any \mathbf{X} : $\mathcal{B}(\mathbf{X}) \stackrel{\text{def}}{=} \mathbf{B} \circ \mathbf{X}$ where $\mathcal{B} \in \mathbb{R}^{d \times n}$. Thus, for (5.12) and (5.13) we shall have respectively

$$\forall i, t, j \in N_t : \quad \frac{2\alpha}{n^2} m_i^j \left(\frac{1}{p_i^t q_j^t} + n \right) + \mathbf{B}_{i,j}^2 \alpha \mu \leq \mathbf{B}_{i,j}^2 p_i^t q_j^t$$

and

$$\forall i, t, j \in N_t : \quad \frac{2\alpha}{n^2} m_i^j \left(\frac{1}{p_i^t q_j^t} + n \right) + \mathbf{B}_{i,j}^2 p_i^t q_j^t \leq \frac{1}{n}.$$

It remains to choose $\mathbf{B}_{i,j}^2 = \frac{1}{2n p_i^t q_j^t}$ and $\alpha = \min_{j \in N_t, i, t} \frac{1}{4m_i^j \left(1 + \frac{1}{n p_i^t q_j^t} \right) + \frac{\mu}{p_i^t q_j^t}}$.

E.12.8 Setup for Corollary E.11.1

Let x be column-wise vectorization of \mathbf{X} . Note that

$$\Gamma(\mathcal{M}^{\frac{1}{2}}(\mathbf{X})) = \mathcal{M}^{\frac{1}{2}}(\mathbf{X}) \mathbf{R} \mathbb{E}[\mathbf{R}]^{-1} \mathbf{e} = (\mathbf{e}^\top \mathbb{E}[\mathbf{R}]^{-1} \mathbf{R} \otimes \mathbf{I}_d) \begin{pmatrix} \mathbf{M}_1^{\frac{1}{2}} & & \\ & \ddots & \\ & & \mathbf{M}_n^{\frac{1}{2}} \end{pmatrix} x.$$

Thus, $\mathbb{E} \left[\left\| \Gamma(\mathcal{M}^{\frac{1}{2}}(\mathbf{X})) \right\|^2 \right] \leq \|\mathbf{X}\|^2 \eta$. Let $\mathcal{B}(\mathbf{X}) = \beta \mathbf{X} \mathbf{B}$. Thus, we have

$$\begin{aligned} & (1 - \alpha \mu) \|\mathcal{B} \mathbf{X}\|^2 - \left\| (\mathcal{I} - \mathbb{E}[\mathcal{S}])^{\frac{1}{2}} \mathcal{B} \mathbf{Y} \right\|^2 \\ &= \beta^2 \text{Tr}(\mathbf{X} \mathbf{B}^\top (\mathbb{E}[\mathbf{R}] - \alpha \mu \mathbf{I}) \mathbf{B} \mathbf{X}^\top) \\ &\leq \beta^2 \lambda_{\min}(\mathbf{B}^\top (\mathbb{E}[\mathbf{R}] - \alpha \mu \mathbf{I}) \mathbf{B}) \|\mathbf{X}\|^2 \\ &\leq \beta^2 (\lambda_{\min}(\mathbf{B}^\top \mathbb{E}[\mathbf{R}] \mathbf{B}) - \alpha \mu \lambda_{\max}(\mathbf{B}^\top \mathbf{B})) \|\mathbf{X}\|^2. \end{aligned}$$

Further,

$$\left\| (\mathbb{E}[\mathcal{S}])^{\frac{1}{2}} \mathcal{B} \mathcal{M}^{\frac{1}{2}} \mathbf{X} \right\|^2 = \beta^2 \text{Tr}(\mathbf{X} \mathbf{B}^\top \mathbb{E}[\mathbf{R}] \mathbf{B} \mathbf{X}^\top) \leq \beta^2 \|\mathbf{X}\|^2 \lambda_{\max}(\mathbf{B}^\top \mathbb{E}[\mathbf{R}] \mathbf{B}).$$

Using the derived bounds together with (E.20), (E.19), for conditions (5.12) and (5.13) it suffices to have:

$$\frac{2\alpha}{n^2} \eta + \beta^2 \alpha \mu \lambda_{\max}(\mathbf{B}^\top \mathbf{B}) \leq \beta^2 \lambda_{\min}(\mathbf{B}^\top \mathbb{E}[\mathbf{R}] \mathbf{B}), \quad (\text{E.29})$$

and

$$\frac{2\alpha}{n^2} \eta + \beta^2 \lambda_{\max}(\mathbf{B}^\top \mathbb{E}[\mathbf{R}] \mathbf{B}) \leq \frac{1}{n}. \quad (\text{E.30})$$

It remains to notice that choices $\beta^2 = \frac{1}{2n\lambda_{\max}(\mathbf{B}^\top \mathbb{E}[\mathbf{R}]\mathbf{B})}$ and

$$\alpha = \frac{\lambda_{\min}(\mathbf{B}^\top \mathbb{E}[\mathbf{R}]\mathbf{B})}{4n^{-1}\eta\lambda_{\max}(\mathbf{B}^\top \mathbb{E}[\mathbf{R}]\mathbf{B}) + \mu\lambda_{\max}(\mathbf{B}^\top \mathbf{B})}$$

are valid.

E.13 Convergence under strong growth condition

In this section, we extend the result of Algorithm 14 to the case when $F \stackrel{\text{def}}{=} f + \psi$ satisfies a strong growth condition instead of quasi strong convexity. Note that strong growth is weaker (more general) than quasi strong convexity [91].

Suppose that \mathcal{X}^* is a set of minimizers of convex function F . Clearly, \mathcal{X}^* must be convex. Define $[x]^*$ to be a projection of x onto \mathcal{X}^* .

Assumption E.13.1. Suppose that F satisfies strong growth, i.e. for every x :

$$F(x) - F([x]^*) \geq \frac{\mu}{2} \|x - [x]^*\|^2. \quad (\text{E.31})$$

E.13.1 Technical proposition and lemma

In order to establish the convergence results, it will be useful to establish Proposition E.13.2 and Lemma E.13.3.

Proposition E.13.2. ([217, 165]) Let f be \mathbf{M} -smooth and suppose that (E.31) holds. Suppose that $x^{k+1} = x^k - \alpha g^k$ where $\mathbb{E}[g^k] = \nabla f(x^k)$ and $\alpha \leq \frac{1}{3\lambda_{\max}(\mathbf{M})}$. Then

$$\mathbb{E}_k \left[\|x^{k+1} - [x^{k+1}]^*\|^2 \right] \leq \frac{1}{1 + \mu\alpha} \mathbb{E}_k \left[\|x^k - [x^k]^*\|^2 \right] + \frac{2\alpha^2}{1 + \mu\alpha} \mathbb{E}_k \left[\|g^k - \nabla f(x^k)\|^2 \right].$$

Lemma E.13.3. For any $x^* \in \mathcal{X}^*$ we have

$$\mathbb{E} \left[\|g^k - \nabla f(x^k)\|^2 \right] \leq \frac{2}{n^2} \mathbb{E} \left[\|\mathcal{U}(\mathbf{G}(x^*) - \mathbf{J}^k)e\|^2 \right] + \frac{2}{n^2} \mathbb{E} \left[\|\mathcal{U}(\mathbf{G}(x^k) - \mathbf{G}(x^*))e\|^2 \right]. \quad (\text{E.32})$$

Proof.

$$\begin{aligned}
& \mathbb{E} [\|g^k - \nabla f(x^k)\|^2] \\
&= \mathbb{E} \left[\left\| \frac{1}{n} \mathbf{J}^k \mathbf{e} - \frac{1}{n} \mathcal{U}(\mathbf{G}(x^k) - \mathbf{J}^k) \mathbf{e} - \frac{1}{n} \mathbf{G}(x^k) \mathbf{e} \right\|^2 \right] \\
&= \frac{1}{n^2} \mathbb{E} \left[\left\| (\mathbf{J}^k - \mathbf{G}(x^*)) \mathbf{e} - \mathcal{U}(\mathbf{G}(x^*) - \mathbf{J}^k) \mathbf{e} + \mathcal{U}(\mathbf{G}(x^k) - \mathbf{G}(x^*)) \mathbf{e} + (\mathbf{G}(x^*) - \mathbf{G}(x^k)) \mathbf{e} \right\|^2 \right] \\
&\leq \frac{2}{n^2} \mathbb{E} \left[\left\| (\mathbf{J}^k - \mathbf{G}(x^*)) \mathbf{e} - \mathcal{U}(\mathbf{G}(x^*) - \mathbf{J}^k) \mathbf{e} \right\|^2 \right] \\
&\quad + \frac{2}{n^2} \mathbb{E} \left[\left\| \mathcal{U}(\mathbf{G}(x^k) - \mathbf{G}(x^*)) \mathbf{e} + (\mathbf{G}(x^*) - \mathbf{G}(x^k)) \mathbf{e} \right\|^2 \right] \\
&\leq \frac{2}{n^2} \mathbb{E} \left[\left\| \mathcal{U}(\mathbf{G}(x^*) - \mathbf{J}^k) \mathbf{e} \right\|^2 \right] + \frac{2}{n^2} \mathbb{E} \left[\left\| \mathcal{U}(\mathbf{G}(x^k) - \mathbf{G}(x^*)) \mathbf{e} \right\|^2 \right].
\end{aligned}$$

□

Lastly, it is necessary to assume the null space consistency of solution set \mathcal{X}^* under \mathbf{M} smoothness. A similar assumption was considered in [165].

Assumption E.13.4. For any $x^*, y^* \in \mathcal{X}$ we have

$$\mathcal{M}^{\dagger \frac{1}{2}} \mathbf{G}(x^*) = \mathcal{M}^{\dagger \frac{1}{2}} \mathbf{G}(y^*). \quad (\text{E.33})$$

E.13.2 Convergence proof

We next state the convergence result of Algorithm 14 under strong growth condition.

Theorem E.13.5. Suppose that (E.31) holds. Let \mathcal{B} be any linear operator commuting with \mathcal{S} , and assume $\mathcal{M}^{\dagger \frac{1}{2}}$ commutes with \mathcal{S} . Let \mathcal{R} be any linear operator for which $\mathcal{R}(\mathbf{J}^k) = \mathcal{R}(\mathbf{G}(x^*))$ for every $k \geq 0$. Define the Lyapunov function Ψ^k as per (5.11) for any $x^* \in \mathcal{X}^*$. Suppose that $\alpha \leq \frac{1}{\lambda_{\max}(\mathbf{M})}$ and \mathcal{B} are chosen so that

$$\begin{aligned}
& \frac{2\alpha}{n^2} \left(\frac{3 + \mu\alpha}{1 + \mu\alpha} \right) \mathbb{E} [\|\mathcal{U} \mathbf{X} \mathbf{e}\|^2] + \left\| (\mathcal{I} - \mathbb{E}[\mathcal{S}])^{\frac{1}{2}} \mathcal{B} \mathcal{M}^{\dagger \frac{1}{2}} \mathbf{X} \right\|^2 \\
& \leq \left(1 - \frac{\alpha\mu}{2 + 2\alpha\mu} \right) \left\| \mathcal{B} \mathcal{M}^{\dagger \frac{1}{2}} \mathbf{X} \right\|^2
\end{aligned} \quad (\text{E.34})$$

whenever $\mathbf{X} \in \text{Range}(\mathcal{R})^\perp$ and

$$\frac{2\alpha}{n^2} \left(\frac{3 + \mu\alpha}{1 + \mu\alpha} \right) \mathbb{E} [\|\mathcal{U} \mathbf{X} \mathbf{e}\|^2] + \left\| (\mathbb{E}[\mathcal{S}])^{\frac{1}{2}} \mathcal{B} \mathcal{M}^{\dagger \frac{1}{2}} \mathbf{X} \right\|^2 \leq \frac{1}{n} \left\| \mathcal{M}^{\dagger \frac{1}{2}} \mathbf{X} \right\|^2 \quad (\text{E.35})$$

for all $\mathbf{X} \in \mathbb{R}^{d \times n}$. Then for all $k \geq 0$, we have

$$\mathbb{E} [\Psi^k] \leq \left(1 - \frac{\alpha\mu}{2 + 2\alpha\mu} \right)^k \Psi^0.$$

Proof. Consider any $x^* \in \mathcal{X}^*$. Due to non-expansiveness of the prox operator we have

$$\begin{aligned}
\mathbb{E} \left[\|x^{k+1} - [x^{k+1}]^*\|_2^2 \right] &\leq \mathbb{E} \left[\|x^{k+1} - [x^k]^*\|_2^2 \right] \\
&\stackrel{(\text{E.12})}{=} \mathbb{E} \left[\|\text{prox}_{\alpha\psi}(x^k - \alpha g^k) - \text{prox}_{\alpha\psi}([x^k]^* - \alpha \nabla f([x^k]^*))\|_2^2 \right] \\
&\leq \mathbb{E} \left[\|x^k - \alpha g^k - ([x^k]^* - \alpha \nabla f([x^k]^*))\|_2^2 \right] \\
&= \|x^k - [x^k]^*\|_2^2 - 2\alpha \langle \nabla f(x^k) - \nabla f([x^k]^*), x^k - [x^k]^* \rangle \\
&\quad + \alpha^2 \mathbb{E} \left[\|g^k - \nabla f([x^k]^*)\|_2^2 \right] \\
&\stackrel{(\text{E.3})}{\leq} \|x^k - [x^k]^*\|_2^2 - \frac{2\alpha}{n} \left\| \mathcal{M}^{\dagger \frac{1}{2}}(\mathbf{G}(x^k) - \mathbf{G}([x^k]^*)) \right\|^2 \\
&\quad + \alpha^2 \mathbb{E} \left[\|g^k - \nabla f([x^k]^*)\|_2^2 \right].
\end{aligned}$$

Combining the above bound with Proposition E.13.2 yields

$$\begin{aligned}
&\mathbb{E} \left[\|x^{k+1} - [x^{k+1}]^*\|_2^2 \right] \\
&\leq \left(\frac{1}{2+2\alpha\mu} + \frac{1}{2} \right) \|x^k - [x^k]^*\|^2 - \frac{\alpha}{n} \left\| \mathcal{M}^{\dagger \frac{1}{2}}(\mathbf{G}(x^k) - \mathbf{G}([x^k]^*)) \right\|^2 \\
&\quad + \frac{1}{2} \alpha^2 \mathbb{E} \left[\|g^k - \nabla f([x^k]^*)\|^2 \right] + \frac{\alpha^2}{1+\mu\alpha} \mathbb{E} \left[\|g^k - \nabla f(x^k)\|^2 \right] \\
&\leq \left(\frac{\alpha\mu+2}{2+2\alpha\mu} \right) \|x^k - [x^k]^*\|^2 - \frac{\alpha}{n} \left\| \mathcal{M}^{\dagger \frac{1}{2}}(\mathbf{G}(x^k) - \mathbf{G}([x^k]^*)) \right\|^2 \\
&\quad + \frac{1}{2} \alpha^2 \mathbb{E} \left[\|g^k - \nabla f([x^k]^*)\|^2 \right] + \frac{\alpha^2}{1+\mu\alpha} \mathbb{E} \left[\|g^k - \nabla f(x^k)\|^2 \right] \\
&\stackrel{(\text{E.32})}{\leq} \left(\frac{\alpha\mu+2}{2+2\alpha\mu} \right) \|x^k - [x^k]^*\|^2 - \frac{\alpha}{n} \left\| \mathcal{M}^{\dagger \frac{1}{2}}(\mathbf{G}(x^k) - \mathbf{G}([x^k]^*)) \right\|^2 \\
&\quad + \frac{2\alpha^2}{n^2(1+\mu\alpha)} \left(\mathbb{E} \left[\|\mathcal{U}(\mathbf{G}(x^*) - \mathbf{J}^k)e\|^2 \right] + \mathbb{E} \left[\|\mathcal{U}(\mathbf{G}(x^k) - \mathbf{G}(x^*))e\|^2 \right] \right) \\
&\quad + \frac{1}{2} \alpha^2 \mathbb{E} \left[\|g^k - \nabla f([x^k]^*)\|^2 \right] \\
&\stackrel{(\text{E.10})}{\leq} \left(\frac{\alpha\mu+2}{2+2\alpha\mu} \right) \|x^k - [x^k]^*\|^2 - \frac{\alpha}{n} \left\| \mathcal{M}^{\dagger \frac{1}{2}}(\mathbf{G}(x^k) - \mathbf{G}([x^k]^*)) \right\|^2 \\
&\quad + \frac{\alpha^2}{n^2} \left(\frac{2}{1+\mu\alpha} + 1 \right) \left(\mathbb{E} \left[\|\mathcal{U}(\mathbf{G}(x^*) - \mathbf{J}^k)e\|^2 \right] + \mathbb{E} \left[\|\mathcal{U}(\mathbf{G}(x^k) - \mathbf{G}(x^*))e\|^2 \right] \right) \\
&\stackrel{(\text{E.33})}{\leq} \left(\frac{\alpha\mu+2}{2+2\alpha\mu} \right) \|x^k - [x^k]^*\|^2 - \frac{\alpha}{n} \left\| \mathcal{M}^{\dagger \frac{1}{2}}(\mathbf{G}(x^k) - \mathbf{G}(x^*)) \right\|^2 \\
&\quad + \frac{\alpha^2}{n^2} \left(\frac{2}{1+\mu\alpha} + 1 \right) \left(\mathbb{E} \left[\|\mathcal{U}(\mathbf{G}(x^*) - \mathbf{J}^k)e\|^2 \right] + \mathbb{E} \left[\|\mathcal{U}(\mathbf{G}(x^k) - \mathbf{G}(x^*))e\|^2 \right] \right).
\end{aligned}$$

Since, by assumption, both \mathcal{B} and $\mathcal{M}^{\dagger \frac{1}{2}}$ commute with \mathcal{S} , so does their composition $\mathcal{A} \stackrel{\text{def}}{=} \mathcal{B}\mathcal{M}^{\dagger \frac{1}{2}}$. Applying Lemma E.2.5, we get

$$\begin{aligned} \mathbb{E} \left[\left\| \mathcal{B}\mathcal{M}^{\dagger \frac{1}{2}} (\mathbf{J}^{k+1} - \mathbf{G}(x^*)) \right\|^2 \right] &= \left\| (\mathcal{I} - \mathbb{E}[\mathcal{S}])^{\frac{1}{2}} \mathcal{B}\mathcal{M}^{\dagger \frac{1}{2}} (\mathbf{J}^k - \mathbf{G}(x^*)) \right\|^2 \\ &\quad + \left\| \mathbb{E}[\mathcal{S}]^{\frac{1}{2}} \mathcal{B}\mathcal{M}^{\dagger \frac{1}{2}} (\mathbf{G}(x^k) - \mathbf{G}(x^*)) \right\|^2 \quad (\text{E.36}) \end{aligned}$$

Adding α multiple of (E.36) to the previous bounds yields

$$\begin{aligned} &\mathbb{E} \left[\left\| x^{k+1} - [x^{k+1}]^* \right\|_2^2 \right] + \alpha \mathbb{E} \left[\left\| \mathcal{B}\mathcal{M}^{\dagger \frac{1}{2}} (\mathbf{J}^{k+1} - \mathbf{G}(x^*)) \right\|^2 \right] \\ \leq &\left(1 - \frac{\alpha\mu}{2 + 2\alpha\mu} \right) \left\| x^k - [x^k]^* \right\|^2 - \frac{\alpha}{n} \left\| \mathcal{M}^{\dagger \frac{1}{2}} (\mathbf{G}(x^k) - \mathbf{G}(x^*)) \right\|^2 \\ &+ \frac{\alpha^2}{n^2} \left(\frac{3 + \mu\alpha}{1 + \mu\alpha} \right) \left(\mathbb{E} \left[\left\| \mathcal{U}(\mathbf{G}(x^*) - \mathbf{J}^k) \right\|^2 \right] + \mathbb{E} \left[\left\| \mathcal{U}(\mathbf{G}(x^k) - \mathbf{G}(x^*)) \right\|^2 \right] \right) \\ &+ \alpha \left\| (\mathcal{I} - \mathbb{E}[\mathcal{S}])^{\frac{1}{2}} \mathcal{B}\mathcal{M}^{\dagger \frac{1}{2}} (\mathbf{J}^k - \mathbf{G}(x^*)) \right\|^2 + \alpha \left\| \mathbb{E}[\mathcal{S}]^{\frac{1}{2}} \mathcal{B}\mathcal{M}^{\dagger \frac{1}{2}} (\mathbf{G}(x^k) - \mathbf{G}(x^*)) \right\|^2 \\ \stackrel{(\text{E.35})}{\leq} &\left(1 - \frac{\alpha\mu}{2 + 2\alpha\mu} \right) \left\| x^k - [x^k]^* \right\|^2 + \frac{\alpha^2}{n^2} \left(\frac{3 + \mu\alpha}{1 + \mu\alpha} \right) \mathbb{E} \left[\left\| \mathcal{U}(\mathbf{G}(x^*) - \mathbf{J}^k) \right\|^2 \right] \\ &+ \alpha \left\| (\mathcal{I} - \mathbb{E}[\mathcal{S}])^{\frac{1}{2}} \mathcal{B}\mathcal{M}^{\dagger \frac{1}{2}} (\mathbf{J}^k - \mathbf{G}(x^*)) \right\|^2 \\ \stackrel{(\text{E.34})}{\leq} &\left(1 - \frac{\alpha\mu}{2 + 2\alpha\mu} \right) \left(\left\| x^k - [x^k]^* \right\|^2 + \alpha \left\| \mathcal{B}\mathcal{M}^{\dagger \frac{1}{2}} (\mathbf{J}^k - \mathbf{G}(x^*)) \right\|^2 \right). \end{aligned}$$

□

Remark 28. Since $2 + 2\alpha\mu = \mathcal{O}(1)$ and $\frac{3\mu\alpha}{1+\mu\alpha} = \mathcal{O}(1)$ the convergence rate under strong growth provided by Theorem E.13.5 is of the same order as the convergence rate under quasi strong convexity (Theorem 5.4.2).

Appendix F

Appendix for Chapter 6

F.1 Special cases

F.1.1 Proximal SGD for stochastic optimization

Algorithm 44 SGD

Require: learning rate $\alpha > 0$, starting point $x^0 \in \mathbb{R}^d$, distribution \mathcal{D} over ξ

for $k = 0, 1, 2, \dots$ **do**
 Sample $\xi \sim \mathcal{D}$
 $g^k = \nabla f_\xi(x^k)$
 $x^{k+1} = \text{prox}_{\alpha\psi}(x^k - \alpha g^k)$
end for

We start with stating the problem, the assumptions on the objective and on the stochastic gradients for SGD [159]. Consider the expectation minimization problem

$$\min_{x \in \mathbb{R}^d} f(x) + \psi(x), \quad f(x) \stackrel{\text{def}}{=} \mathbb{E}[f_\xi(x)] \quad (\text{F.1})$$

where $\xi \sim \mathcal{D}$, $f_\xi(x)$ is differentiable and L -smooth almost surely in ξ .

Lemma F.1.1 shows that the stochastic gradient $g^k = \nabla f_\xi(x^k)$ satisfies Assumption 6.3.1. The corresponding choice of parameters can be found in Table 6.2.

Lemma F.1.1 (Generalization of Lemmas 1,2 from [159]). *Assume that $f_\xi(x)$ is convex in x for every ξ . Then for every $x \in \mathbb{R}^d$*

$$\mathbb{E} [\|\nabla f_\xi(x) - \nabla f(x^*)\|^2] \leq 4L(D_f(x, x^*)) + 2\sigma^2, \quad (\text{F.2})$$

where $\sigma^2 \stackrel{\text{def}}{=} \mathbb{E}_\xi [\|\nabla f_\xi(x^*)\|^2]$. *If further $f(x)$ is μ -strongly convex with possibly non-convex f_ξ , then for every $x \in \mathbb{R}^d$*

$$\mathbb{E} [\|\nabla f_\xi(x) - \nabla f(x^*)\|^2] \leq 4L\kappa(D_f(x, x^*)) + 2\sigma^2, \quad (\text{F.3})$$

where $\kappa = \frac{L}{\mu}$.

Corollary F.1.2. *Assume that $f_\xi(x)$ is convex in x for every ξ and f is μ -strongly quasi-convex. Then SGD with $\alpha \leq \frac{1}{2L}$ satisfies*

$$\mathbb{E} [\|x^k - x^*\|^2] \leq (1 - \alpha\mu)^k \|x^0 - x^*\|^2 + \frac{2\alpha\sigma^2}{\mu}. \quad (\text{F.4})$$

If we further assume that $f(x)$ is μ -strongly convex with possibly non-convex $f_\xi(x)$, SGD with $\alpha \leq \frac{1}{2L\kappa}$ satisfies (F.4) as well.

Proof. It suffices to plug parameters from Table 6.2 into Theorem 6.3.4. \square

Proof of Lemma F.1.1

The proof is a direct generalization to the one from [159]. Note that

$$\begin{aligned} & \frac{1}{2} \mathbb{E} [\|\nabla f_\xi(x) - \nabla f(x^*)\|^2] - \mathbb{E} [\|\nabla f_\xi(x^*) - \nabla f(x^*)\|^2] \\ &= \frac{1}{2} \mathbb{E} [\|\nabla f_\xi(x) - \nabla f(x^*)\|^2 - \|\nabla f_\xi(x^*) - \nabla f(x^*)\|^2] \\ &\stackrel{(F.62)}{\leq} \mathbb{E} [\|\nabla f_\xi(x) - \nabla f_\xi(x^*)\|^2] \\ &\leq 2LD_f(x, x^*). \end{aligned}$$

It remains to rearrange the above to get (F.2). To obtain (F.3), we shall proceed similarly:

$$\begin{aligned} & \frac{1}{2} \mathbb{E} [\|\nabla f_\xi(x) - \nabla f(x^*)\|^2] - \mathbb{E} [\|\nabla f_\xi(x^*) - \nabla f(x^*)\|^2] \\ &= \frac{1}{2} \mathbb{E} [\|\nabla f_\xi(x) - \nabla f(x^*)\|^2 - \|\nabla f_\xi(x^*) - \nabla f(x^*)\|^2] \\ &\stackrel{(F.62)}{\leq} \mathbb{E} [\|\nabla f_\xi(x) - \nabla f_\xi(x^*)\|^2] \\ &\leq L^2 \|x - x^*\|^2 \\ &\leq 2 \frac{L^2}{\mu} D_f(x, x^*). \end{aligned}$$

Again, it remains to rearrange the terms.

F.1.2 SGD-SR

In this section, we recover convergence result of SGD under expected smoothness property from [60]. This setup allows obtaining tight convergence rates of SGD under arbitrary stochastic reformulation of finite sum minimization¹.

The stochastic reformulation is a special instance of (F.1):

$$\min_{x \in \mathbb{R}^d} f(x) + \psi(x), \quad f(x) = \mathbb{E} [f_\xi(x)], \quad f_\xi(x) \stackrel{\text{def}}{=} \frac{1}{n} \sum_{i=1}^n \xi_i f_i(x) \quad (\text{F.5})$$

where ξ is a random vector from distribution \mathcal{D} such that for all i : $\mathbb{E} [\xi_i] = 1$ and f_i (for all i) is smooth, possibly non-convex function. We next state the expected smoothness assumption. A specific instances of this assumption allows to get tight convergence rates of SGD, which we recover in this section.

¹For technical details on how to exploit expected smoothness for specific reformulations, see [60]

Algorithm 45 SGD-SR

Require: learning rate $\alpha > 0$, starting point $x^0 \in \mathbb{R}^d$, distribution \mathcal{D} over $\xi \in \mathbb{R}^n$ such that $\mathbb{E}[\xi]$ is vector of ones

for $k = 0, 1, 2, \dots$ **do**

Sample $\xi \sim \mathcal{D}$

$g^k = \nabla f_\xi(x^k)$

$x^{k+1} = \text{prox}_{\alpha\psi}(x^k - \alpha g^k)$

end for

Assumption F.1.3 (Expected smoothness). *We say that f is \mathcal{L} -smooth in expectation with respect to distribution \mathcal{D} if there exists $\mathcal{L} = \mathcal{L}(f, \mathcal{D}) > 0$ such that*

$$\mathbb{E} [\|\nabla f_\xi(x) - \nabla f_\xi(x^*)\|^2] \leq 2\mathcal{L}D_f(x, x^*), \quad (\text{F.6})$$

for all $x \in \mathbb{R}^d$. For simplicity, we will write $(f, \mathcal{D}) \sim ES(\mathcal{L})$ to say that (F.6) holds.

Next, we present Lemma F.1.4 which shows that choice of constants for Assumption 6.3.1 from Table 6.2 is valid.

Lemma F.1.4 (Generalization of Lemma 2.4, [60]). *If $(f, \mathcal{D}) \sim ES(\mathcal{L})$, then*

$$\mathbb{E} [\|\nabla f_\xi(x) - \nabla f(x^*)\|^2] \leq 4\mathcal{L}D_f(x, x^*) + 2\sigma^2. \quad (\text{F.7})$$

where $\sigma^2 \stackrel{\text{def}}{=} \mathbb{E} [\|\nabla f_\xi(x^*) - \nabla f(x^*)\|^2]$.

A direct consequence of Theorem 6.3.4 in this setup is Corollary F.1.5.

Corollary F.1.5. *Assume that $f(x)$ is μ -strongly quasi-convex and $(f, \mathcal{D}) \sim ES(\mathcal{L})$. Then SGD-SR with $\alpha^k \equiv \alpha \leq \frac{1}{2\mathcal{L}}$ satisfies*

$$\mathbb{E} [\|x^k - x^*\|^2] \leq (1 - \alpha\mu)^k \|x^0 - x^*\|^2 + \frac{2\alpha\sigma^2}{\mu}. \quad (\text{F.8})$$

Proof of Lemma F.1.4

Here we present the generalization of the proof of Lemma 2.4 from [60] for the case when $\nabla f(x^*) \neq 0$. In this proof all expectations are conditioned on x^k .

$$\begin{aligned} \mathbb{E} [\|\nabla f_\xi(x) - \nabla f(x^*)\|^2] &= \mathbb{E} [\|\nabla f_\xi(x) - \nabla f_\xi(x^*) + \nabla f_\xi(x^*) - \nabla f(x^*)\|^2] \\ &\stackrel{(\text{F.61})}{\leq} 2\mathbb{E} [\|\nabla f_\xi(x) - \nabla f_\xi(x^*)\|^2] + 2\mathbb{E} [\|\nabla f_\xi(x^*) - \nabla f(x^*)\|^2] \\ &\stackrel{(\text{F.6})}{\leq} 4\mathcal{L}D_f(x, x^*) + 2\sigma^2. \end{aligned}$$

F.1.3 SGD-MB

In this section, we present a specific practical formulation of (F.5) which was not considered in [60]. The resulting algorithm (Algorithm 46) is novel; it was not considered in [60]

as a specific instance of SGD-SR. The key idea behind SGD-MB is constructing unbiased gradient estimate via with-replacement sampling.

Consider random variable $\nu \sim \mathcal{D}$ such that

$$\mathbb{P}(\nu = i) = p_i; \quad \sum_{i=1}^n p_i = 1. \quad (\text{F.9})$$

Notice that if we define

$$f'_i(x) \stackrel{\text{def}}{=} \frac{1}{np_i} f_i(x), \quad i = 1, 2, \dots, n, \quad (\text{F.10})$$

then

$$f(x) = \frac{1}{n} \sum_{i=1}^n f_i(x) \stackrel{(\text{F.10})}{=} \sum_{i=1}^n p_i f'_i(x) \stackrel{(\text{F.9})}{=} \mathbb{E}[f'_\nu(x)]. \quad (\text{F.11})$$

So, we have rewritten the finite sum problem (6.3) into the *equivalent stochastic optimization problem*

$$\min_{x \in \mathbb{R}^d} \mathbb{E}[f'_\nu(x)]. \quad (\text{F.12})$$

We are now ready to describe our method. At each iteration k we sample $\nu_i^k, \dots, \nu_\tau^k \sim \mathcal{D}$ independently ($1 \leq \tau \leq n$), and define $g^k \stackrel{\text{def}}{=} \frac{1}{\tau} \sum_{i=1}^{\tau} \nabla f'_{\nu_i^k}(x^k)$. Further, we use g^k as a stochastic gradient, resulting in Algorithm 46.

Algorithm 46 SGD-MB

Require: learning rate $\alpha > 0$, starting point $x^0 \in \mathbb{R}^d$, distribution \mathcal{D} over ν such that (F.9) holds.

for $k = 0, 1, 2, \dots$ **do**

Sample $\nu_i^k, \dots, \nu_\tau^k \sim \mathcal{D}$ independently

$g^k = \frac{1}{\tau} \sum_{i=1}^{\tau} \nabla f'_{\nu_i^k}(x^k)$

$x^{k+1} = x^k - \alpha g^k$

end for

To remain in full generality, consider the following Assumption.

Assumption F.1.6. *There exists constants $A' > 0$ and $D' \geq 0$ such that*

$$\mathbb{E}[\|\nabla f'_\nu(x)\|^2] \leq 2A'(f(x) - f(x^*)) + D' \quad (\text{F.13})$$

for all $x \in \mathbb{R}^d$.

Note that it is sufficient to have convex and smooth f_i in order to satisfy Assumption F.1.6, as Lemma F.1.7 states.

Lemma F.1.7. Let $\sigma^2 \stackrel{\text{def}}{=} \mathbb{E} [\|\nabla f'_\nu(x^*)\|^2]$. If f_i are convex and L_i -smooth, then Assumption F.1.6 holds for $A' = 2\mathcal{L}$ and $D' = 2\sigma^2$, where

$$\mathcal{L} \leq \max_i \frac{L_i}{np_i}. \quad (\text{F.14})$$

If moreover $\nabla f_i(x^*) = 0$ for all i , then Assumption F.1.6 holds for $A' = \mathcal{L}$ and $D' = 0$.

Next, Lemma F.1.8 states that Algorithm 46 indeed satisfies Assumption 6.3.1.

Lemma F.1.8. Suppose that Assumption F.1.6 holds. Then g^k is unbiased; i.e. $\mathbb{E} [g^k] = \nabla f(x^k)$. Further,

$$\mathbb{E} [\|g^k\|^2] \leq \frac{2A' + 2L(\tau - 1)}{\tau} (f(x^k) - f(x^*)) + \frac{D'}{\tau}.$$

Thus, parameters from Table 6.2 are validated. As a direct consequence of Theorem 6.3.4 we get Corollary F.1.9.

Corollary F.1.9. As long as $0 < \alpha \leq \frac{\tau}{A' + L(\tau - 1)}$, we have

$$\mathbb{E} \|x^k - x^*\|^2 \leq (1 - \alpha\mu)^k \|x^0 - x^*\|^2 + \frac{\alpha D'}{\mu\tau}. \quad (\text{F.15})$$

Remark 29. For $\tau = 1$, SGD-MB is a special of the method from [60], Section 3.2. However, for $\tau > 1$, this is a different method; the difference lies in the with-replacement sampling. Note that with-replacement trick allows for efficient and implementation of independent importance sampling² with complexity $\mathcal{O}(\tau \log(n))$. In contrast, implementation of without-replacement importance sampling has complexity $\mathcal{O}(n)$, which can be significantly more expensive to the cost of evaluating $\sum_{i \in S} \nabla f_i(x)$.

Proof of Lemma F.1.8

Notice first that

$$\begin{aligned} \mathbb{E} [g^k] &\stackrel{(\text{F.10})}{=} \frac{1}{\tau} \sum_{i=1}^{\tau} \mathbb{E} \left[\frac{1}{np_{\nu_i^k}} \nabla f_{\nu_i^k}(x^k) \right] \\ &= \mathbb{E} \left[\frac{1}{np_{\nu}} \nabla f_{\nu}(x^k) \right] \\ &\stackrel{(\text{F.9})}{=} \sum_{i=1}^n p_i \frac{1}{np_i} \nabla f_i(x^k) \\ &= \nabla f(x^k). \end{aligned}$$

²Distribution of random sets S for which random variables $i \in S$ and $j \in S$ are independent for $j \neq i$.

So, g^k is an unbiased estimator of the gradient $\nabla f(x^k)$. Next,

$$\begin{aligned}
\mathbb{E} [\|g^k\|^2] &= \mathbb{E} \left[\left\| \frac{1}{\tau} \sum_{i=1}^{\tau} \nabla f'_{\nu_i^k}(x^k) \right\|^2 \right] \\
&= \frac{1}{\tau^2} \mathbb{E} \left[\sum_{i=1}^{\tau} \|\nabla f'_{\nu_i^k}(x^k)\|^2 + 2 \sum_{i < j} \langle \nabla f'_{\nu_i^k}(x^k), \nabla f'_{\nu_j^k}(x^k) \rangle \right] \\
&= \frac{1}{\tau} \mathbb{E} [\|\nabla f'_{\nu}(x^k)\|^2] + \frac{2}{\tau^2} \sum_{i < j} \langle \mathbb{E} [\nabla f'_{\nu_i^k}(x^k)], \mathbb{E} [\nabla f'_{\nu_j^k}(x^k)] \rangle \\
&= \frac{1}{\tau} \mathbb{E} [\|\nabla f'_{\nu}(x^k)\|^2] + \frac{\tau-1}{\tau} \|\nabla f(x^k)\|^2 \\
&\stackrel{(F.13)}{\leq} \frac{2A'(f(x^k) - f(x^*)) + D' + 2L(\tau-1)(f(x^k) - f(x^*))}{\tau}.
\end{aligned}$$

Proof of Lemma F.1.7

Let $\mathcal{L} = \mathcal{L}(f, \mathcal{D}) > 0$ be any constant for which

$$\mathbb{E}_{\xi \sim \mathcal{D}} \|\nabla \phi_{\xi}(x) - \nabla \phi_{\xi}(x^*)\|^2 \leq 2\mathcal{L}(f(x) - f(x^*)) \quad (\text{F.16})$$

holds for all $x \in \mathbb{R}^d$. This is the expected smoothness property (for a single item sampling) from [60]. It was shown in [60, Proposition 3.7] that (F.16) holds, and that \mathcal{L} satisfies (F.14). The claim now follows by applying [60, Lemma 2.4].

F.1.4 SGD-star

Consider problem (F.5). Suppose that $\nabla f_i(x^*)$ is known for all i . In this section we present a novel algorithm — SGD-star — which is SGD-SR shifted by the stochastic gradient in the optimum. The method is presented under Expected Smoothness Assumption (F.6), obtaining general rates under arbitrary sampling. The algorithm is presented as Algorithm 47.

Algorithm 47 SGD-star

Require: learning rate $\alpha > 0$, starting point $x^0 \in \mathbb{R}^d$, distribution \mathcal{D} over $\xi \in \mathbb{R}^n$ such that $\mathbb{E}[\xi]$ is vector of ones
for $k = 0, 1, 2, \dots$ **do**
 Sample $\xi \sim \mathcal{D}$
 $g^k = \nabla f_{\xi}(x^k) - \nabla f_{\xi}(x^*) + \nabla f(x^*)$
 $x^{k+1} = \text{prox}_{\alpha\psi}(x^k - \alpha g^k)$
end for

Suppose that $(f, \mathcal{D}) \sim ES(\mathcal{L})$. Note next that SGD-star is just SGD-SR applied on objective $D_f(x, x^*)$ instead of $f(x)$ when $\nabla f(x^*) = 0$. This careful design of the

objective yields $(D_f(\cdot, x^*), \mathcal{D}) \sim ES(\mathcal{L})$ and $\mathbb{E} \left[\|\nabla_x D_{f_\xi}(x, x^*)\|^2 \mid x = x^* \right] = 0$, and thus Lemma (F.1.4) becomes

Lemma F.1.10 (Lemma 2.4, [60]). *If $(f, \mathcal{D}) \sim ES(\mathcal{L})$, then*

$$\mathbb{E} \left[\|g^k - \nabla f(x^*)\|^2 \right] \leq 4\mathcal{L}D_f(x^k, x^*). \quad (\text{F.17})$$

A direct consequence of Corollary (thus also a direct consequence of Theorem 6.3.4) in this setup is Corollary F.1.11.

Corollary F.1.11. *Suppose that $(f, \mathcal{D}) \sim ES(\mathcal{L})$. Then SGD-star with $\alpha = \frac{1}{2\mathcal{L}}$ satisfies*

$$\mathbb{E} \left[\|x^k - x^*\|^2 \right] \leq \left(1 - \frac{\mu}{2\mathcal{L}}\right)^k \|x^0 - x^*\|^2. \quad (\text{F.18})$$

Remark 30. Note that results from this section are obtained by applying results from F.1.2. Since Section F.1.3 presets a specific sampling algorithm for SGD-SR, the results can be thus extended to SGD-star as well.

Proof of Lemma F.1.10

In this proof all expectations are conditioned on x^k .

$$\begin{aligned} \mathbb{E} \left[\|g^k - \nabla f(x^*)\|^2 \right] &= \mathbb{E} \left[\|\nabla f_\xi(x^k) - \nabla f_\xi(x^*)\|^2 \right] \\ &\stackrel{(\text{F.6})}{\leq} 4\mathcal{L}D_f(x^k, x^*). \end{aligned}$$

F.1.5 SAGA

In this section we show that our approach is suitable for SAGA [37] (see Algorithm 48). Consider the finite-sum minimization problem

$$f(x) = \frac{1}{n} \sum_{i=1}^n f_i(x) + \psi(x), \quad (\text{F.19})$$

where f_i is convex, L -smooth for each i and f is μ -strongly convex.

Lemma F.1.12. *We have*

$$\mathbb{E} \left[\|g^k - \nabla f(x^*)\|^2 \mid x^k \right] \leq 4LD_f(x^k, x^*) + 2\sigma_k^2 \quad (\text{F.20})$$

and

$$\mathbb{E} \left[\sigma_{k+1}^2 \mid x^k \right] \leq \left(1 - \frac{1}{n}\right) \sigma_k^2 + \frac{2L}{n} D_f(x^k, x^*), \quad (\text{F.21})$$

where $\sigma_k^2 = \frac{1}{n} \sum_{i=1}^n \|\nabla f_i(\phi_i^k) - \nabla f_i(x^*)\|^2$.

Algorithm 48 SAGA [37]**Require:** learning rate $\alpha > 0$, starting point $x^0 \in \mathbb{R}^d$ Set $\phi_j^0 = x^0$ for each $j \in [n]$ **for** $k = 0, 1, 2, \dots$ **do**Sample $j \in [n]$ uniformly at randomSet $\phi_j^{k+1} = x^k$ and $\phi_i^{k+1} = \phi_i^k$ for $i \neq j$

$$g^k = \nabla f_j(\phi_j^{k+1}) - \nabla f_j(\phi_j^k) + \frac{1}{n} \sum_{i=1}^n \nabla f_i(\phi_i^k)$$

$$x^{k+1} = \text{prox}_{\alpha\psi}(x^k - \alpha g^k)$$

end for

Clearly, Lemma F.1.12 shows that Algorithm 48 satisfies Assumption 6.3.1; the corresponding parameter choice can be found in Table 6.2. Thus, as a direct consequence of Theorem 6.3.4 with $M = 4n$ we obtain the next corollary.

Corollary F.1.13. SAGA with $\alpha = \frac{1}{6L}$ satisfies

$$\mathbb{E}V^k \leq \left(1 - \min\left\{\frac{\mu}{6L}, \frac{1}{2n}\right\}\right)^k V^0. \quad (\text{F.22})$$

Proof of Lemma F.1.12

Note that Lemma F.1.12 is a special case of Lemmas 3,4 from [137] without prox term. We reprove it with prox for completeness.

Let all expectations be conditioned on x^k in this proof. Note that L -smoothness and convexity of f_i implies

$$\frac{1}{2L} \|\nabla f_i(x) - \nabla f_i(y)\|^2 \leq f_i(x) - f_i(y) - \langle \nabla f_i(y), x - y \rangle, \quad \forall x, y \in \mathbb{R}^d, i \in [n]. \quad (\text{F.23})$$

By definition of g^k we have

$$\begin{aligned}
& \mathbb{E} \left[\|g^k - \nabla f(x^*)\|^2 \right] \\
&= \mathbb{E} \left[\left\| \nabla f_j(\phi_j^{k+1}) - \nabla f_j(\phi_j^k) + \frac{1}{n} \sum_{i=1}^n \nabla f_i(\phi_i^k) - \nabla f(x^*) \right\|^2 \right] \\
&= \mathbb{E} \left[\left\| \nabla f_j(x^k) - \nabla f_j(x^*) + \nabla f_j(x^*) - \nabla f_j(\phi_j^k) + \frac{1}{n} \sum_{i=1}^n \nabla f_i(\phi_i^k) - \nabla f(x^*) \right\|^2 \right] \\
&\stackrel{(F.61)}{\leq} 2\mathbb{E} \left[\|\nabla f_j(x^k) - \nabla f_j(x^*)\|^2 \mid x^k \right] \\
&\quad + 2\mathbb{E} \left[\|\nabla f_j(x^*) - \nabla f_j(\phi_j^k) - \mathbb{E} [\nabla f_j(x^*) - \nabla f_j(\phi_j^k)]\|^2 \right] \\
&\stackrel{(F.63)+(F.23)}{\leq} \frac{4L}{n} \sum_{i=1}^n D_{f_i}(x^k, x^*) + 2\mathbb{E} \left[\|\nabla f_j(x^*) - \nabla f_j(\phi_j^k)\|^2 \mid x^k \right] \\
&= 4LD_f(x^k, x^*) + 2 \underbrace{\frac{1}{n} \sum_{i=1}^n \|\nabla f_i(\phi_i^k) - \nabla f_i(x^*)\|^2}_{\sigma_k^2}.
\end{aligned}$$

To proceed with (F.21), we have

$$\begin{aligned}
\mathbb{E} [\sigma_{k+1}^2] &= \frac{1}{n} \sum_{i=1}^n \mathbb{E} \left[\|\nabla f_i(\phi_i^{k+1}) - \nabla f_i(x^*)\|^2 \right] \\
&= \frac{1}{n} \sum_{i=1}^n \left(\frac{n-1}{n} \|\nabla f_i(\phi_i^k) - \nabla f_i(x^*)\|^2 + \frac{1}{n} \|\nabla f_i(x^k) - \nabla f_i(x^*)\|^2 \right) \\
&\stackrel{(F.23)}{\leq} \left(1 - \frac{1}{n} \right) \frac{1}{n} \sum_{i=1}^n \|\nabla f_i(\phi_i^k) - \nabla f_i(x^*)\|^2 \\
&\quad + \frac{2L}{n^2} \sum_{i=1}^n D_{f_i}(x^k, x^*) \\
&= \left(1 - \frac{1}{n} \right) \sigma_k^2 + \frac{2L}{n} D_f(x^k, x^*).
\end{aligned}$$

F.1.6 N-SAGA

Note that it can in practice happen that instead of $\nabla f_i(x)$ one can query $g_i(x, \xi)$ such that $\mathbb{E}_\xi g_i(\cdot, \xi) = \nabla f_i(\cdot)$ and $\mathbb{E}_\xi \|g_i(\cdot, \xi)\|^2 \leq \sigma^2$. This leads to a variant of SAGA which only uses noisy estimates of the stochastic gradients $\nabla_i(\cdot)$. We call this variant N-SAGA (see Algorithm 49).

Algorithm 49 Noisy SAGA (N-SAGA)**Require:** learning rate $\alpha > 0$, starting point $x^0 \in \mathbb{R}^d$ Set $\psi_j^0 = x^0$ for each $j \in [0]$ **for** $k = 0, 1, 2, \dots$ **do**Sample $j \in [n]$ uniformly at random and ζ Set $g_j^{k+1} = g_j(x^k, \xi)$ and $g_i^{k+1} = g_i^k$ for $i \neq j$

$$g^k = g_j(x^k, \xi) - g_j^k + \frac{1}{n} \sum_{i=1}^n g_i^k$$

$$x^{k+1} = \text{prox}_{\alpha\psi}(x^k - \alpha g^k)$$

end for**Lemma F.1.14.** *We have*

$$\mathbb{E} \left[\|g^k - \nabla f(x^*)\|^2 \mid x^k \right] \leq 4LD_f(x^k, x^*) + 2\sigma_k^2 + 2\sigma^2, \quad (\text{F.24})$$

and

$$\mathbb{E} [\sigma_{k+1}^2 \mid x^k] \leq \left(1 - \frac{1}{n}\right) \sigma_k^2 + \frac{2L}{n} D_f(x^k, x^*) + \frac{\sigma^2}{n}, \quad (\text{F.25})$$

where $\sigma_k^2 \stackrel{\text{def}}{=} \frac{1}{n} \sum_{i=1}^n \|g_i^k - \nabla f_i(x^*)\|^2$.

Corollary F.1.15. *Let $\alpha = \frac{1}{6L}$. Then, iterates of Algorithm 49 satisfy*

$$\mathbb{E} V^k \leq \left(1 - \min\left(\frac{\mu}{6L}, \frac{1}{2n}\right)\right)^k V^0 + \frac{\sigma^2}{L \min(\mu, \frac{3L}{n})}.$$

Analogous results can be obtained for LSVRG.

Proof of Lemma F.1.14

Let all expectations be conditioned on x^k . By definition of g^k we have

$$\begin{aligned}
& \mathbb{E} \left[\|g^k - \nabla f(x^*)\|^2 \right] \\
& \leq \mathbb{E} \left[\left\| g_j(x^k, \zeta) - g_j^k + \frac{1}{n} \sum_{i=1}^n g_i^k - \nabla f(x^*) \right\|^2 \right] \\
& = \mathbb{E} \left[\left\| g_j(x^k, \zeta) - \nabla f_j(x^*) + \nabla f_j(x^*) - g_j^k + \frac{1}{n} \sum_{i=1}^n g_i^k - \nabla f(x^*) \right\|^2 \right] \\
& \stackrel{(F.61)}{\leq} 2\mathbb{E} \left[\|g_j(x^k, \zeta) - \nabla f_j(x^*)\|^2 \right] \\
& \quad + 2\mathbb{E} \left[\|\nabla f_j(x^*) - g_j^k - \mathbb{E}[\nabla f_j(x^*) - g_j^k]\|^2 \right] \\
& \stackrel{(F.63)}{\leq} 2\mathbb{E} \left[\|g_j(x^k, \zeta) - \nabla f_j(x^*)\|^2 \right] + 2\mathbb{E} \left[\|\nabla f_j(x^*) - g_j^k\|^2 \right] \\
& = 2\mathbb{E} \left[\|g_j(x^k, \zeta) - \nabla f_j(x^*)\|^2 \right] + 2 \underbrace{\frac{1}{n} \sum_{i=1}^n \|g_i^k - \nabla f_i(x^*)\|^2}_{\sigma_k^2} \\
& \stackrel{(F.63)}{\leq} 2\mathbb{E} \left[\|\nabla f_j(x^k) - \nabla f_j(x^*)\|^2 \right] + 2\sigma^2 + 2\sigma_k^2 \\
& \stackrel{(F.23)}{\leq} 4LD_f(x^k, x^*) + 2\sigma_k^2 + 2\sigma^2
\end{aligned}$$

For the second inequality, we have

$$\begin{aligned}
\mathbb{E}[\sigma_{k+1}^2] &= \frac{1}{n} \sum_{i=1}^n \mathbb{E} \left[\|g_i^{k+1} - \nabla f_i(x^*)\|^2 \right] \\
&= \frac{1}{n} \sum_{i=1}^n \left(\frac{n-1}{n} \|g_i^k - \nabla f_i(x^*)\|^2 + \frac{1}{n} \mathbb{E} \left[\|g_i(x^k, \zeta) - \nabla f_i(x^*)\|^2 \right] \right) \\
&\leq \frac{1}{n} \sum_{i=1}^n \left(\frac{n-1}{n} \|g_i^k - \nabla f_i(x^*)\|^2 + \frac{1}{n} \|\nabla f_i(x^k) - \nabla f_i(x^*)\|^2 + \frac{\sigma^2}{n} \right) \\
&\stackrel{(F.23)}{\leq} \left(1 - \frac{1}{n} \right) \sigma_k^2 + \frac{2L}{n} D_f(x^k, x^*) + \frac{\sigma^2}{n}.
\end{aligned}$$

F.1.7 SEGA

We show that the framework recovers the simplest version of SEGA (i.e., setup from Theorem D1 from [77]) in the proximal setting³.

³General version for arbitrary gradient sketches instead of partial derivatives can be recovered as well, however, we omit it for simplicity

Algorithm 50 SEGA [77]**Require:** learning rate $\alpha > 0$, starting point $x^0 \in \mathbb{R}^d$ Set $h^0 = 0$ **for** $k = 0, 1, 2, \dots$ **do**Sample $j \in [d]$ uniformly at randomSet $h^{k+1} = h^k + e_i(\nabla_i f(x^k) - h_i^k)$ $g^k = de_i(\nabla_i f(x^k) - h_i^k) + h^k$ $x^{k+1} = \text{prox}_{\alpha\psi}(x^k - \alpha g^k)$ **end for****Lemma F.1.16.** (Consequence of Lemmas A.3., A.4. from [77]) We have

$$\mathbb{E} [\|g^k - \nabla f(x^*)\|^2 | x^k] \leq 2d \|\nabla f(x^k) - \nabla f(x^*)\|^2 + 2d\sigma_k^2$$

and

$$\mathbb{E} [\sigma_{k+1}^2 | x^k] = \left(1 - \frac{1}{d}\right) \sigma_k^2 + \frac{1}{d} \|\nabla f(x^k) - \nabla f(x^*)\|^2,$$

where $\sigma_k^2 \stackrel{\text{def}}{=} \|h^k - \nabla f(x^*)\|^2$.

Given that we have from convexity and smoothness $\|\nabla f(x^k) - \nabla f(x^*)\|^2 \leq 2LD_f(x^k, x^*)$, Assumption 6.3.1 holds the parameter choice as per Table 6.2. Setting further $M = 4d^2$, we get the next corollary.

Corollary F.1.17. SEGA with $\alpha = \frac{1}{6dL}$ satisfies

$$\mathbb{E} V^k \leq \left(1 - \frac{\mu}{6dL}\right)^k V^0.$$

F.1.8 N-SEGA**Algorithm 51** Noisy SEGA (N-SEGA)**Require:** learning rate $\alpha > 0$, starting point $x^0 \in \mathbb{R}^d$ Set $h^0 = 0$ **for** $k = 0, 1, 2, \dots$ **do**Sample $i \in [d]$ uniformly at random and sample ξ Set $h^{k+1} = h^k + e_i(g_i(x, \xi) - h_i^k)$ $g^k = de_i(g_i(x, \xi) - h_i^k) + h^k$ $x^{k+1} = x^k - \alpha g^k$ **end for**

Here we assume that $g_i(x, \zeta)$ is a noisy estimate of the partial derivative $\nabla_i f(x)$ such that $\mathbb{E}_\zeta g_i(x, \zeta) = \nabla_i f(x)$ and $\mathbb{E}_\zeta |g_i(x, \zeta) - \nabla_i f(x)|^2 \leq \frac{\sigma^2}{d}$.

Lemma F.1.18. *The following inequalities hold:*

$$\mathbb{E} \left[\|g^k - \nabla f(x^*)\|^2 \right] \leq 4dLD_f(x^k, x^*) + 2d\sigma_k^2 + 2d\sigma^2,$$

$$\mathbb{E} [\sigma_{k+1}^2] \leq \left(1 - \frac{1}{d}\right) \sigma_k^2 + \frac{2L}{d} D_f(x^k, x^*) + \frac{\sigma^2}{d},$$

where $\sigma_k^2 = \|h^k - \nabla f(x^*)\|^2$.

Corollary F.1.19. *Let $\alpha = \frac{1}{6Ld}$. Applying Theorem 6.3.4 with $M = 4d^2$, iterates of Algorithm 51 satisfy*

$$\mathbb{E} V^k \leq \left(1 - \frac{\mu}{6dL}\right)^k V^0 + \frac{\sigma^2}{L\mu}.$$

Proof of Lemma F.1.18

Let all expectations be conditioned on x^k . For the first bound, we write

$$g^k - \nabla f(x^*) = \underbrace{h^k - \nabla f(x^*) - dh_i^k e_i + d\nabla_i f(x^*) e_i}_a + \underbrace{dg_i(x^k, \xi) e_i - d\nabla_i f(x^*) e_i}_b.$$

Let us bound the expectation of each term individually. The first term can be bounded as

$$\begin{aligned} \mathbb{E} \|a\|^2 &= \mathbb{E} \|(\mathbf{I} - de_i e_i^\top) (h^k - \nabla f(x^*))\|_2^2 \\ &= (d-1) \|h^k - \nabla f(x^*)\|^2 \\ &\leq d \|h^k - \nabla f(x^*)\|^2. \end{aligned}$$

The second term can be bounded as

$$\begin{aligned} \mathbb{E} \|b\|^2 &= \mathbb{E}_i \mathbb{E}_\xi \|dg_i(x, \xi) e_i - d\nabla f_i(x^*) e_i\|^2 \\ &= \mathbb{E}_i \mathbb{E}_\xi \|dg_i(x^k, \xi) e_i - d\nabla_i f(x^k) e_i\|^2 + \mathbb{E}_i \|d\nabla_i f(x^k) e_i - d\nabla f_i(x^*) e_i\|^2 \\ &\leq d\sigma^2 + d \|\nabla f(x^k) - \nabla f(x^*)\|^2 \\ &\leq d\sigma^2 + 2LdD_f(x^k, x^*), \end{aligned}$$

where in the last step we used L -smoothness of f . It remains to combine the two bounds.

For the second bound, we have

$$\begin{aligned}
\mathbb{E} \|h^{k+1} - \nabla f(x^*)\|^2 &= \mathbb{E} \|h^k + g_i(x^k, \xi)e_i - h_i^k - \nabla f(x^*)\|^2 \\
&= \mathbb{E} \|(\mathbf{I} - e_i e_i^\top) h^k + g_i(x^k, \xi)e_i - \nabla f(x^*)\|^2 \\
&= \mathbb{E} \|(\mathbf{I} - e_i e_i^\top) (h^k - \nabla f(x^*))\|^2 + \mathbb{E} \|g_i(x^k, \xi)e_i - \nabla_i f(x^*)e_i\|^2 \\
&= \left(1 - \frac{1}{d}\right) \|h^k - \nabla f(x^*)\|^2 + \mathbb{E} \|g_i(x^k, \xi)e_i - \nabla_i f(x^*)e_i\|^2 \\
&\quad + \mathbb{E} \|\nabla_i f(x^k)e_i - \nabla_i f(x^*)e_i\|^2 \\
&= \left(1 - \frac{1}{d}\right) \|h^k - \nabla f(x^*)\|^2 + \frac{\sigma^2}{d} + \frac{1}{d} \|\nabla f(x^k) - \nabla f(x^*)\|^2 \\
&\leq \left(1 - \frac{1}{d}\right) \|h^k - \nabla f(x^*)\|^2 + \frac{\sigma^2}{d} + \frac{2L}{d} D_f(x^k, x^*).
\end{aligned}$$

F.1.9 SVRG

Algorithm 52 SVRG [88]

Require: learning rate $\alpha > 0$, epoch length m , starting point $x^0 \in \mathbb{R}^d$

```

 $\phi = x^0$ 
for  $s = 0, 1, 2, \dots$  do
  for  $k = 0, 1, 2, \dots, m-1$  do
    Sample  $i \in \{1, \dots, n\}$  uniformly at random
     $g^k = \nabla f_i(x^k) - \nabla f_i(\phi) + \nabla f(\phi)$ 
     $x^{k+1} = \text{prox}_{\alpha\psi}(x^k - \alpha g^k)$ 
  end for
   $\phi = x^0 = \frac{1}{m} \sum_{k=1}^m x^k$ 
end for

```

Let $\sigma_k^2 \stackrel{\text{def}}{=} \frac{1}{n} \sum_{i=1}^n \|\nabla f_i(\phi) - \nabla f_i(x^*)\|^2$. We will show that Lemma 6.3.3 recovers per-epoch analysis of SVRG in a special case.

Lemma F.1.20. *For $k \bmod m \neq 0$ we have*

$$\mathbb{E} \left[\|g^k - \nabla f(x^*)\|^2 \mid x^k \right] \leq 4LD_f(x^k, x^*) + 2\sigma_k^2 \quad (\text{F.26})$$

and

$$\mathbb{E} [\sigma_{k+1}^2 \mid x^k] = \sigma_{k+1}^2 = \sigma_k^2. \quad (\text{F.27})$$

Proof. The proof of (F.26) is identical to the proof of (F.20). Next, (F.27) holds since σ_k does not depend on k . \square

Thus, Assumption 6.3.1 holds with parameter choice as per Table 6.2 and Lemma 6.3.3 implies the next corollary.

Corollary F.1.21.

$$\mathbb{E} \|x^{k+1} - x^*\|^2 + \alpha(1 - 2\alpha L)\mathbb{E} D_f(x^k, x^*) \leq (1 - \alpha\mu)\mathbb{E} \|x^k - x^*\|^2 + 2\alpha^2\mathbb{E}\sigma_k^2. \quad (\text{F.28})$$

Recovering SVRG rate

Summing (F.28) for $k = 0, \dots, m-1$ using $\sigma_k = \sigma_0$ we arrive at

$$\begin{aligned} \mathbb{E} \|x^m - x^*\|^2 + \sum_{k=1}^m \alpha(1 - 2\alpha L)\mathbb{E} D_f(x^k, x^*) &\leq (1 - \alpha\mu)\mathbb{E} \|x^0 - x^*\|^2 + 2m\alpha^2\mathbb{E}\sigma_0^2 \\ &\leq 2(\mu^{-1} + 2m\alpha^2 L) D_f(x^0, x^*). \end{aligned}$$

Since D_f is convex in the first argument, we have

$$m\alpha(1 - 2\alpha L)D_f\left(\frac{1}{m}\sum_{k=1}^m x^k, x^*\right) \leq \|x^m - x^*\|^2 + \sum_{k=1}^m \alpha(1 - 2\alpha L)D_f(x^k, x^*)$$

and thus

$$D_f\left(\frac{1}{m}\sum_{k=1}^m x^k, x^*\right) \leq \frac{2(\mu^{-1} + 2m\alpha^2 L)}{m\alpha(1 - 2\alpha L)} D_f(x^0, x^*),$$

which recovers rate from Theorem 1 in [88].

F.1.10 LSVRG

In this section we show that our approach also covers LSVRG analysis from [83, 106] (see Algorithm 53) with a minor extension – it allows for proximable regularizer ψ . Consider the finite-sum minimization problem

$$f(x) = \frac{1}{n} \sum_{i=1}^n f_i(x) + \psi(x), \quad (\text{F.29})$$

where each f_i convex and L -smooth for each i and f is μ -strongly convex.

Algorithm 53 LSVRG ([83, 106])

Require: learning rate $\alpha > 0$, probability $p \in (0, 1]$, starting point $x^0 \in \mathbb{R}^d$

$w^0 = x^0$

for $k = 0, 1, 2, \dots$ **do**

 Sample $i \in \{1, \dots, n\}$ uniformly at random

$g^k = \nabla f_i(x^k) - \nabla f_i(w^k) + \nabla f(w^k)$

$x^{k+1} = x^k - \alpha g^k$

$w^{k+1} = \begin{cases} x^k & \text{with probability } p \\ w^k & \text{with probability } 1 - p \end{cases}$

end for

Note that the gradient estimator is again unbiased, i.e. $\mathbb{E}[g^k | x^k] = \nabla f(x^k)$. Next, Lemma F.1.22 provides with the remaining constants for Assumption 6.3.1. The corresponding choice is stated in Table 6.2.

Lemma F.1.22 (Lemma 4.2 and Lemma 4.3 from [106] extended to prox setup). *We have*

$$\mathbb{E}[\|g^k - \nabla f(x^*)\|^2 | x^k] \leq 4LD_f(x^k, x^*) + 2\sigma_k^2 \quad (\text{F.30})$$

and

$$\mathbb{E}[\sigma_{k+1}^2 | x^k] \leq (1-p)\sigma_k^2 + 2LpD_f(x^k, x^*), \quad (\text{F.31})$$

where $\sigma_k^2 \stackrel{\text{def}}{=} \frac{1}{n} \sum_{i=1}^n \|\nabla f_i(w^k) - \nabla f_i(x^*)\|^2$.

Next, applying Theorem 6.3.4 on Algorithm 53 with $M = \frac{4}{p}$ we get Corollary F.1.23.

Corollary F.1.23. LSVRG with $\alpha = \frac{1}{6L}$ satisfies

$$\mathbb{E}V^k \leq \left(1 - \min\left\{\frac{\mu}{6L}, \frac{p}{2}\right\}\right)^k V^0. \quad (\text{F.32})$$

Proof of Lemma F.1.22

Let all expectations be conditioned on x^k . Using definition of g^k

$$\begin{aligned} & \mathbb{E}[\|g^k - \nabla f(x^*)\|^2] \\ \stackrel{\text{Alg. 53}}{=} & \mathbb{E}[\|\nabla f_i(x^k) - \nabla f_i(x^*) + \nabla f_i(x^*) - \nabla f_i(w^k) + \nabla f(w^k) - \nabla f(x^*)\|^2] \\ \stackrel{(\text{F.61})}{\leq} & 2\mathbb{E}[\|\nabla f_i(x^k) - \nabla f_i(x^*)\|^2] \\ & + 2\mathbb{E}[\|\nabla f_i(x^*) - \nabla f_i(w^k) - \mathbb{E}[\nabla f_i(x^*) - \nabla f_i(w^k) | x^k]\|^2] \\ \stackrel{(\text{F.23}), (\text{F.63})}{\leq} & 4LD_f(x^k, x^*) + 2\mathbb{E}[\|\nabla f_i(w^k) - \nabla f_i(x^*)\|^2] \\ = & 4LD_f(x^k, x^*) + 2\sigma_k^2. \end{aligned}$$

For the second bound, we shall have

$$\begin{aligned} \mathbb{E}[\sigma_{k+1}^2] & \stackrel{\text{Alg. 53}}{=} (1-p)\sigma_k^2 + \frac{p}{n} \sum_{i=1}^n \|\nabla f_i(x^k) - \nabla f_i(x^*)\|^2 \\ & \stackrel{(\text{F.23})}{\leq} (1-p)\sigma_k^2 + 2LpD_f(x^k, x^*). \end{aligned}$$

F.1.11 DIANA

In this section we consider a distributed setup where each function f_i from (6.3) is owned by i th machine (thus, we have all together n machines).

Algorithm 54 DIANA [136, 85]

Require: learning rates $\gamma > 0$ and $\alpha > 0$, initial vectors $x^0, h_1^0, \dots, h_n^0 \in \mathbb{R}^d$ and $h^0 = \frac{1}{n} \sum_{i=1}^n h_i^0$

```

1: for  $k = 0, 1, 2, \dots$  do
2:   Broadcast  $x^k$  to all workers
3:   for  $i = 1, \dots, n$  in parallel do
4:     Sample  $g_i^k$  such that  $\mathbb{E}[g_i^k \mid x^k] = \nabla f_i(x^k)$ 
5:      $\Delta_i^k = g_i^k - h_i^k$ 
6:     Sample  $\hat{\Delta}_i^k \sim Q(\Delta_i^k)$ 
7:      $h_i^{k+1} = h_i^k + \gamma \hat{\Delta}_i^k$ 
8:      $\hat{g}_i^k = h_i^k + \hat{\Delta}_i^k$ 
9:   end for
10:   $\hat{\Delta}^k = \frac{1}{n} \sum_{i=1}^n \hat{\Delta}_i^k$ 
11:   $g^k = \frac{1}{n} \sum_{i=1}^n \hat{g}_i^k = h^k + \hat{\Delta}^k$ 
12:   $x^{k+1} = \text{prox}_{\alpha\psi}(x^k - \alpha g^k)$ 
13:   $h^{k+1} = \frac{1}{n} \sum_{i=1}^n h_i^{k+1} = h^k + \gamma \hat{\Delta}^k$ 
14: end for

```

We show that our approach covers the analysis of DIANA from [136, 85]. DIANA is a specific algorithm for distributed optimization with *quantization* – lossy compression of gradient updates, which reduces the communication between the server and workers⁴.

In particular, DIANA quantizes gradient differences instead of the actual gradients. This trick allows for the linear convergence to the optimum once the full gradients are evaluated on each machine, unlike other popular quantization methods such as QSGD [2] or TernGrad [213]. In this case, DIANA behaves as variance reduced method – it reduces a variance that was injected due to the quantization. However, DIANA also allows for evaluation of stochastic gradients on each machine, as we shall further see.

First of all, we introduce the notion of quantization operator.

Definition F.1.24 (Quantization). *We say that $\hat{\Delta}$ is a quantization of vector $\Delta \in \mathbb{R}^d$ and write $\hat{\Delta} \sim Q(\Delta)$ if*

$$\mathbb{E}\hat{\Delta} = \Delta, \quad \mathbb{E}\|\hat{\Delta} - \Delta\|^2 \leq \omega \|\Delta\|^2 \quad (\text{F.33})$$

for some $\omega > 0$.

The aforementioned method is applied to solve problem (6.1)+(6.3) where each f_i is convex and L -smooth and f is μ -strongly convex.

Lemma F.1.25 (Lemma 1 and consequence of Lemma 2 from [85]). *Suppose that $\gamma \leq$*

⁴It is a well-known problem in distributed optimization that the communication between machines often takes more time than actual computation.

$\frac{1}{1+\omega}$. For all iterations $k \geq 0$ of Algorithm 54 it holds

$$\mathbb{E}[g^k | x^k] = \nabla f(x^k), \quad (\text{F.34})$$

$$\begin{aligned} \mathbb{E}[\|g^k - \nabla f(x^*)\|^2 | x^k] &\leq \left(1 + \frac{2\omega}{n}\right) \frac{1}{n} \sum_{i=1}^n \|\nabla f_i(x^k) - \nabla f_i(x^*)\|^2 \\ &\quad + \frac{2\omega\sigma_k^2}{n} + \frac{(1+\omega)\sigma^2}{n}, \end{aligned} \quad (\text{F.35})$$

$$\mathbb{E}[\sigma_{k+1}^2 | x^k] \leq (1-\gamma)\sigma_k^2 + \frac{\gamma}{n} \sum_{i=1}^n \|\nabla f_i(x^k) - \nabla f_i(x^*)\|^2 + \gamma\sigma^2. \quad (\text{F.36})$$

where $\sigma_k^2 = \frac{1}{n} \sum_{i=1}^n \|h_i^k - \nabla f_i(x^*)\|^2$ and σ^2 is such that $\frac{1}{n} \sum_{i=1}^n \mathbb{E}[\|g_i^k - \nabla f_i(x^*)\|^2 | x^k] \leq \sigma^2$.

Bounding further $\frac{1}{n} \sum_{i=1}^n \|\nabla f_i(x^k) - \nabla f_i(x^*)\|^2 \leq 2LD_f(x^k, x^*)$ in the above Lemma, we see that Assumption 6.3.1 as per Table 6.2 is valid. Thus, as a special case of Theorem 6.3.4, we obtain the following corollary.

Corollary F.1.26. Assume that f_i is convex and L -smooth for all $i \in [n]$ and f is μ strongly convex, $\gamma \leq \frac{1}{\omega+1}$, $\alpha \leq \frac{1}{(1+\frac{2\omega}{n})L+ML\gamma}$ where $M > \frac{2\omega}{n\gamma}$. Then the iterates of DIANA satisfy

$$\mathbb{E}[V^k] \leq \max \left\{ (1 - \alpha\mu)^k, \left(1 + \frac{2\omega}{nM} - \gamma\right)^k \right\} V^0 + \frac{\left(\frac{1+\omega}{n} + M\gamma\right) \sigma^2 \alpha^2}{\min \left\{ \alpha\mu, \gamma - \frac{2\omega}{nM} \right\}}, \quad (\text{F.37})$$

where the Lyapunov function V^k is defined by $V^k \stackrel{\text{def}}{=} \|x^k - x^*\|^2 + M\alpha^2\sigma_k^2$. For the particular choice $\gamma = \frac{1}{\omega+1}$, $M = \frac{4\omega(\omega+1)}{n}$, $\alpha = \frac{1}{(1+\frac{6\omega}{n})L}$, then DIANA converges to a solution neighborhood and the leading iteration complexity term is

$$\max \left\{ \frac{1}{\alpha\mu}, \frac{1}{\gamma - \frac{2\omega}{nM}} \right\} = \max \left\{ \kappa + \kappa \frac{6\omega}{n}, 2(\omega+1) \right\}, \quad (\text{F.38})$$

where $\kappa = \frac{L}{\mu}$.

As mentioned, once the full (deterministic) gradients are evaluated on each machine, DIANA converges linearly to the exact optimum. In particular, in such case we have $\sigma^2 = 0$. Corollary F.1.27 states the result in the case when $n = 1$, i.e. there is only a single node⁵. For completeness, we present the mentioned simple case of DIANA as Algorithm 55.

Corollary F.1.27. Assume that f_i is μ -strongly convex and L -smooth for all $i \in [n]$, $\gamma \leq \frac{1}{\omega+1}$, $\alpha \leq \frac{1}{(1+2\omega)L+ML\gamma}$ where $M > \frac{2\omega}{\gamma}$. Then the stochastic gradient \hat{g}^k and the objective function f satisfy Assumption 6.3.1 with $A = (1+2\omega)L$, $B = 2\omega$, $\sigma_k^2 =$

⁵node = machine

Algorithm 55 DIANA: 1 node & exact gradients [136, 85]**Require:** learning rates $\gamma > 0$ and $\alpha > 0$, initial vectors $x^0, h^0 \in \mathbb{R}^d$

```

1: for  $k = 0, 1, 2, \dots$  do
2:    $\Delta^k = \nabla f(x^k) - h^k$ 
3:   Sample  $\hat{\Delta}^k \sim Q(\Delta^k)$ 
4:    $h^{k+1} = h^k + \gamma \hat{\Delta}^k$ 
5:    $g^k = h^k + \hat{\Delta}^k$ 
6:    $x^{k+1} = \text{prox}_{\alpha\psi}(x^k - \alpha g^k)$ 
7: end for

```

 $\|h^k - h^*\|^2, \rho = \gamma, C = L\gamma, D_1 = 0, D_2 = 0$ and

$$\mathbb{E}[V^k] \leq \max \left\{ (1 - \alpha\mu)^k, \left(1 + \frac{2\omega}{M} - \gamma\right)^k \right\} V^0, \quad (\text{F.39})$$

where the Lyapunov function V^k is defined by $V^k \stackrel{\text{def}}{=} \|x^k - x^*\|^2 + M\alpha^2\sigma_k^2$. For the particular choice $\gamma = \frac{1}{\omega+1}$, $M = 4\omega(\omega+1)$, $\alpha = \frac{1}{(1+6\omega)L}$ the leading term in the iteration complexity bound is

$$\max \left\{ \frac{1}{\alpha\mu}, \frac{1}{\gamma - \frac{2\omega}{M}} \right\} = \max \{ \kappa + 6\kappa\omega, 2(\omega+1) \}, \quad (\text{F.40})$$

where $\kappa = \frac{L}{\mu}$.

F.1.12 Q-SGD-SR

In this section, we consider a quantized version of SGD-SR.

Algorithm 56 Q-SGD-SR**Require:** learning rate $\alpha > 0$, starting point $x^0 \in \mathbb{R}^d$, distribution \mathcal{D} over $\xi \in \mathbb{R}^n$ such that $\mathbb{E}[\xi]$ is vector of ones

```

for  $k = 0, 1, 2, \dots$  do
  Sample  $\xi \sim \mathcal{D}$ 
   $g^k \sim Q(\nabla f_\xi(x^k))$ 
   $x^{k+1} = \text{prox}_{\alpha\psi}(x^k - \alpha g^k)$ 
end for

```

Lemma F.1.28 (Generalization of Lemma 2.4, [60]). *If $(f, \mathcal{D}) \sim ES(\mathcal{L})$, then*

$$\mathbb{E}[\|g^k - \nabla f(x^*)\|^2] \leq 4\mathcal{L}(1 + \omega)D_f(x^k, x^*) + 2\sigma^2(1 + \omega). \quad (\text{F.41})$$

where $\sigma^2 \stackrel{\text{def}}{=} \mathbb{E}[\|\nabla f_\xi(x^*)\|^2]$.

A direct consequence of Theorem 6.3.4 in this setup is Corollary F.1.29.

Corollary F.1.29. Assume that $f(x)$ is μ -strongly quasi-convex and $(f, \mathcal{D}) \sim ES(\mathcal{L})$. Then Q-SGD-SR with $\alpha^k \equiv \alpha \leq \frac{1}{2(1+\omega)\mathcal{L}}$ satisfies

$$\mathbb{E} [\|x^k - x^*\|^2] \leq (1 - \alpha\mu)^k \|x^0 - x^*\|^2 + \frac{2\alpha(1+\omega)\sigma^2}{\mu}. \quad (\text{F.42})$$

Proof of Lemma F.1.28

In this proof all expectations are conditioned on x^k . First of all, from Lemma F.1.4 we have

$$\mathbb{E} [\|\nabla f_\xi(x^k) - \nabla f(x^*)\|^2] \leq 4\mathcal{L}D_f(x^k, x^*) + 2\sigma^2.$$

The remaining step is to understand how quantization of $\nabla f_\xi(x^k)$ changes the above inequality if we put $g^k \sim Q(\nabla f_\xi(x^k))$ instead of $\nabla f_\xi(x^k)$. Let us denote mathematical expectation with respect randomness coming from quantization by $\mathbb{E}_Q[\cdot]$. Using tower property of mathematical expectation we get

$$\begin{aligned} \mathbb{E} [\|g^k - \nabla f(x^*)\|^2] &= \mathbb{E}_{\mathcal{D}} [\mathbb{E}_Q \|g^k - \nabla f(x^*)\|^2] \\ &\stackrel{(\text{F.63})}{=} \mathbb{E} [\|g^k - \nabla f_\xi(x^k)\|^2] + \mathbb{E} [\|\nabla f_\xi(x^k) - \nabla f(x^*)\|^2] \\ &\stackrel{(\text{F.41})}{\leq} \mathbb{E} [\|g^k - \nabla f_\xi(x^k)\|^2] + 4\mathcal{L}D_f(x^k, x^*) + 2\sigma^2. \end{aligned}$$

Next, we estimate the first term in the last row of the previous inequality

$$\begin{aligned} \mathbb{E} [\|g^k - \nabla f_\xi(x^k)\|^2] &\stackrel{(\text{F.33})}{\leq} \omega \mathbb{E} [\|\nabla f_\xi(x^k)\|^2] \\ &\stackrel{(\text{F.61})}{\leq} 2\omega \mathbb{E} [\|\nabla f_\xi(x^k) - \nabla f_\xi(x^*)\|^2] + 2\omega \mathbb{E} [\|\nabla f_\xi(x^*)\|^2] \\ &\leq 4\omega \mathcal{L}D_f(x^k, x^*) + 2\omega\sigma^2. \end{aligned}$$

Putting all together we get the result.

F.1.13 VR-DIANA

Corollary F.1.26 shows that once each machine evaluates a stochastic gradient instead of the full gradient, DIANA converges linearly only to a certain neighborhood. In contrast, VR-DIANA [85] uses a variance reduction trick within each machine, which enables linear convergence to the exact solution. In this section, we show that our approach recovers VR-DIANA as well.

The aforementioned method is applied to solve problem (6.1)+(6.3) where each f_i is also of a finite sum structure, as in (6.4), with each $f_{ij}(x)$ being convex and L -smooth, and $f_i(x)$ being μ -strongly convex. Note that $\nabla f(x^*) = 0$ and, in particular, $D_f(x, x^*) = f(x) - f(x^*)$ since the problem is considered without regularization.

Algorithm 57 VR-DIANA based on LSVRG (Variant 1), SAGA (Variant 2), [85]

Require: learning rates $\gamma > 0$ and $\alpha > 0$, initial vectors $x^0, h_1^0, \dots, h_n^0, h^0 = \frac{1}{n} \sum_{i=1}^n h_i^0$

- 1: **for** $k = 0, 1, 2, \dots$ **do**
- 2: Sample random $u^k = \begin{cases} 1, & \text{with probability } \frac{1}{m} \\ 0, & \text{with probability } 1 - \frac{1}{m} \end{cases}$ ▷ only for Variant 1
- 3: Broadcast x^k, u^k to all workers
- 4: **for** $i = 1, \dots, n$ in parallel **do** ▷ Worker side
- 5: Pick random $j_i^k \sim_{\text{u.a.r.}} [m]$
- 6: $\mu_i^k = \frac{1}{m} \sum_{j=1}^m \nabla f_{ij}(w_{ij}^k)$
- 7: $g_i^k = \nabla f_{ij_i^k}(x^k) - \nabla f_{ij_i^k}(w_{ij_i^k}^k) + \mu_i^k$
- 8: $\hat{\Delta}_i^k = Q(g_i^k - h_i^k)$
- 9: $h_i^{k+1} = h_i^k + \gamma \hat{\Delta}_i^k$
- 10: **for** $j = 1, \dots, m$ **do**
- 11: $w_{ij}^{k+1} = \begin{cases} x^k, & \text{if } u^k = 1 \\ w_{ij}^k, & \text{if } u^k = 0 \end{cases}$ ▷ Variant 1 (LSVRG): update epoch gradient if
- $u^k = 1$
- 12: $w_{ij}^{k+1} = \begin{cases} x^k, & j = j_i^k \\ w_{ij}^k, & j \neq j_i^k \end{cases}$ ▷ Variant 2 (SAGA): update gradient table
- 13: **end for**
- 14: **end for**
- 15: $h^{k+1} = h^k + \frac{\gamma}{n} \sum_{i=1}^n \hat{\Delta}_i^k$ ▷ Gather quantized updates
- 16: $g^k = \frac{1}{n} \sum_{i=1}^n (\hat{\Delta}_i^k + h_i^k)$
- 17: $x^{k+1} = x^k - \alpha g^k$
- 18: **end for**

Lemma F.1.30 (Lemmas 3, 5, 6 and 7 from [85]). *Let $\gamma \leq \frac{1}{\omega+1}$. Then for all iterates $k \geq 0$ of Algorithm 57 the following inequalities hold:*

$$\mathbb{E}[g^k | x^k] = \nabla f(x^k), \quad (\text{F.43})$$

$$\mathbb{E}[H^{k+1} | x^k] \leq (1 - \gamma) H^k + \frac{2\gamma}{m} D^k + 8\gamma L n (f(x^k) - f(x^*)), \quad (\text{F.44})$$

$$\mathbb{E}[D^{k+1} | x^k] \leq \left(1 - \frac{1}{m}\right) D^k + 2L n (f(x^k) - f(x^*)), \quad (\text{F.45})$$

$$\mathbb{E}[\|g^k\|^2 | x^k] \leq 2L \left(1 + \frac{4\omega + 2}{n}\right) (f(x^k) - f(x^*)) + \frac{2\omega}{n^2} \frac{D^k}{m} + \frac{2(\omega + 1)}{n^2} H^k, \quad (\text{F.46})$$

where $H^k = \sum_{i=1}^n \|h_i^k - \nabla f_i(x^*)\|^2$ and $D^k = \sum_{i=1}^n \sum_{j=1}^m \|\nabla f_{ij}(w_{ij}^k) - \nabla f_{ij}(x^*)\|^2$.

Corollary F.1.31. *Let $\gamma \leq \min\{\frac{1}{3m}, \frac{1}{\omega+1}\}$. Then stochastic gradient \hat{g}^k (Algorithm 57) and the objective function f satisfy Assumption 6.3.1 with $A = (1 + \frac{4\omega+2}{n})L, B =$*

$\frac{2(\omega+1)}{n}, \rho = \gamma, C = L \left(\frac{1}{m} + 4\gamma \right), D_1 = 0, D_2 = 0$ and

$$\sigma_k^2 = \frac{H^k}{n} + \frac{D^k}{nm} = \frac{1}{n} \sum_{i=1}^n \|h_i^k - \nabla f_i(x^*)\|^2 + \frac{1}{nm} \sum_{i=1}^n \sum_{j=1}^m \|\nabla f_{ij}(w_{ij}^k) - \nabla f_{ij}(x^*)\|^2.$$

Proof. Indeed, (6.7) holds due to (F.43). Inequality (6.8) follows from (F.46) with $A = \left(1 + \frac{4\omega+2}{n}\right) L, B = \frac{2(\omega+1)}{n}, D_1 = 0, \sigma_k^2 = \frac{H^k}{n} + \frac{D^k}{nm}$ if we take into account that $\frac{2\omega}{n^2} \frac{D^k}{m} + \frac{2(\omega+1)}{n^2} H^k \leq \frac{2(\omega+1)}{n} \left(\frac{D^k}{nm} + \frac{H^k}{n} \right)$. Finally, summing inequalities (F.44) and (F.45) and using $\gamma \leq \frac{1}{3m}$

$$\begin{aligned} \mathbb{E} [\sigma_k^2 \mid x^k] &= \frac{1}{n} \mathbb{E} [H^{k+1} \mid x^k] + \frac{1}{nm} \mathbb{E} [D^{k+1} \mid x^k] \\ &\stackrel{(F.44)+(F.45)}{\leq} (1-\gamma) \frac{H^k}{n} + \left(1 + 2\gamma - \frac{1}{m}\right) \frac{D^k}{nm} + 2L \left(\frac{1}{m} + 4\gamma\right) (f(x^k) - f(x^*)) \\ &\leq (1-\gamma) \sigma_k^2 + 2L \left(\frac{1}{m} + 4\gamma\right) (f(x^k) - f(x^*)) \end{aligned}$$

we get (6.9) with $\rho = \gamma, C = L \left(\frac{1}{m} + 4\gamma \right), D_2 = 0$. \square

Corollary F.1.32. Assume that f_i is μ -strongly convex and f_{ij} is convex and L -smooth for all $i \in [n], j \in [m], \gamma \leq \min \left\{ \frac{1}{3m}, \frac{1}{\omega+1} \right\}, \alpha \leq \frac{1}{\left(1 + \frac{4\omega+2}{n}\right)L + ML\left(\frac{1}{m} + 4\gamma\right)}$ where $M > \frac{2(\omega+1)}{n\gamma}$. Then the iterates of VR-DIANA satisfy

$$\mathbb{E} [V^k] \leq \max \left\{ (1 - \alpha\mu)^k, \left(1 + \frac{2(\omega+1)}{nM} - \gamma\right)^k \right\} V^0, \quad (F.47)$$

where the Lyapunov function V^k is defined by $V^k \stackrel{\text{def}}{=} \|x^k - x^*\|^2 + M\alpha^2\sigma_k^2$. Further, if we set $\gamma = \min \left\{ \frac{1}{3m}, \frac{1}{\omega+1} \right\}, M = \frac{4(\omega+1)}{n\gamma}, \alpha = \frac{1}{\left(1 + \frac{20\omega+18}{n} + \frac{4\omega+4}{n\gamma m}\right)L}$, then to achieve precision

$$\mathbb{E} [\|x^k - x^*\|^2] \leq \varepsilon V^0 \text{ VR-DIANA needs}$$

$$\mathcal{O} \left(\max \left\{ \kappa + \kappa \frac{\omega+1}{n} + \kappa \frac{(\omega+1) \max \{m, \omega+1\}}{nm}, m, \omega+1 \right\} \log \frac{1}{\varepsilon} \right)$$

iterations, where $\kappa = \frac{L}{\mu}$.

Proof. Using Corollary F.1.31 we apply Theorem 6.3.4 and get the result. \square

Remark 31. VR-DIANA can be easily extended to the proximal setup in our framework.

F.1.14 JacSketch

In this section, we show that our approach covers the analysis of JacSketch from [65]. JacSketch is a generalization of SAGA in the following manner. SAGA observes every iteration $\nabla f_i(x)$ for random index i and uses it to build both stochastic gradient as well as the control variates on the stochastic gradient in order to progressively decrease

variance. In contrast, JacSketch observes every iteration the random sketch of the Jacobian, which is again used to build both stochastic gradient as well as the control variates on the stochastic gradient.

For simplicity, we do not consider proximal setup, since [65] does not either.

We first introduce the necessary notation (same as in [65]). Denote first the Jacobian the objective

$$\nabla \mathbf{F}(x) \stackrel{\text{def}}{=} [\nabla f_1(x), \dots, \nabla f_n(x)] \in \mathbb{R}^{d \times n}. \quad (\text{F.48})$$

Every iteration of the method, a random sketch of Jacobian $\nabla F(x^k)\mathbf{S}$ (where $\mathbf{S} \sim \mathcal{D}$) is observed. Then, the method builds a variable \mathbf{J}^k , which is the current Jacobian estimate, updated using so-called sketch and project iteration [61]:

$$\mathbf{J}^{k+1} = \mathbf{J}^k(\mathbf{I} - \Pi_{\mathbf{S}_k}) + \nabla \mathbf{F}(x^k)\Pi_{\mathbf{S}_k},$$

where $\Pi_{\mathbf{S}}$ is a projection under \mathbf{W} norm⁶ ($\mathbf{W} \in \mathbb{R}^{n \times n}$ is some positive definite weight matrix) defined as $\Pi_{\mathbf{S}} \stackrel{\text{def}}{=} \mathbf{S}(\mathbf{S}^\top \mathbf{W} \mathbf{S})^\dagger \mathbf{S}^\top \mathbf{W}$ ⁷.

Further, in order to construct unbiased stochastic gradient, an access to the random scalar $\theta_{\mathbf{S}}$ such that

$$\mathbb{E}[\theta_{\mathbf{S}} \Pi_{\mathbf{S}}] e = e, \quad (\text{F.49})$$

where e is the vector of all ones.

Next, the simplest option for the choice of the stochastic gradient is $\nabla f_{\mathbf{S}}(x)$ – an unbiased estimate of ∇f directly constructed using $\mathbf{S}, \theta_{\mathbf{S}}$:

$$\nabla f_{\mathbf{S}}(x) = \frac{\theta_{\mathbf{S}}}{n} \nabla \mathbf{F}(x) \Pi_{\mathbf{S}} e. \quad (\text{F.50})$$

However, one can build a smarter estimate $\nabla f_{\mathbf{S}, \mathbf{J}}(x)$ via control variates constructed from \mathbf{J} :

$$\nabla f_{\mathbf{S}, \mathbf{J}}(x) = \frac{\theta_{\mathbf{S}}}{n} (\nabla \mathbf{F}(x) - \mathbf{J}) \Pi_{\mathbf{S}} e + \frac{1}{n} \mathbf{J} e. \quad (\text{F.51})$$

The resulting algorithm is stated as Algorithm 58.

Algorithm 58 JacSketch [65]

Require: $(\mathcal{D}, \mathbf{W}, \theta_{\mathbf{S}})$, $x^0 \in \mathbb{R}^d$, Jacobian estimate $\mathbf{J}^0 \in \mathbb{R}^{d \times n}$, stepsize $\alpha > 0$

- 1: **for** $k = 0, 1, 2, \dots$ **do**
 - 2: Sample a fresh copy $\mathbf{S}_k \sim \mathcal{D}$
 - 3: $\mathbf{J}^{k+1} = \mathbf{J}^k(\mathbf{I} - \Pi_{\mathbf{S}_k}) + \nabla \mathbf{F}(x^k) \Pi_{\mathbf{S}_k}$
 - 4: $g^k = \nabla f_{\mathbf{S}_k, \mathbf{J}^k}(x^k)$
 - 5: $x^{k+1} = x^k - \alpha g^k$
 - 6: **end for**
-

Next we present Lemma F.1.33 which directly justifies the parameter choice from

⁶Weighted Frobenius norm of matrix $\mathbf{X} \in \mathbb{R}^{n \times n}$ with a positive definite weight matrix $\mathbf{W} \in \mathbb{R}^{n \times n}$ is defined as $\|\mathbf{X}\|_{\mathbf{W}^{-1}} \stackrel{\text{def}}{=} \sqrt{\text{Tr}(\mathbf{X} \mathbf{W}^{-1} \mathbf{X}^\top)}$.

⁷Symbol \dagger stands for Moore-Penrose pseudoinverse.

Table 6.1.

Lemma F.1.33 (Lemmas 2.5, 3.9 and 3.10 from [65]). *Suppose that there are constants $\mathcal{L}_1, \mathcal{L}_2 > 0$ such that*

$$\begin{aligned} \mathbb{E} [\|\nabla f_{\mathbf{S}}(x) - \nabla f_{\mathbf{S}}(x^*)\|_2^2] &\leq 2\mathcal{L}_1(f(x) - f(x^*)), \quad \forall x \in \mathbb{R}^d \\ \mathbb{E} [\|(\nabla \mathbf{F}(x) - \nabla \mathbf{F}(x^*))\Pi_{\mathbf{S}}\|_{\mathbf{W}^{-1}}^2] &\leq 2\mathcal{L}_2(f(x) - f(x^*)), \quad \forall x \in \mathbb{R}^d, \end{aligned}$$

Then

$$\mathbb{E} [\|\mathbf{J}^{k+1} - \nabla \mathbf{F}(x^*)\|_{\mathbf{W}^{-1}}^2] \leq (1 - \lambda_{\min}) \|\mathbf{J}^k - \nabla \mathbf{F}(x^*)\|_{\mathbf{W}^{-1}}^2 + 2\mathcal{L}_2(f(x^k) - f(x^*)), \quad (\text{F.52})$$

$$\mathbb{E} [\|g^k\|_2^2] \leq 4\mathcal{L}_1(f(x^k) - f(x^*)) + 2\frac{\lambda_{\max}}{n^2} \|\mathbf{J}^k - \nabla \mathbf{F}(x^*)\|_{\mathbf{W}^{-1}}^2, \quad (\text{F.53})$$

where $\lambda_{\min} = \lambda_{\min}(\mathbb{E}[\Pi_{\mathbf{S}}])$ and $\lambda_{\max} = \lambda_{\max}(\mathbf{W}^{1/2}(\mathbb{E}[\theta_{\mathbf{S}}^2 \Pi_{\mathbf{S}} e e^\top \Pi_{\mathbf{S}}] - e e^\top) \mathbf{W}^{1/2})$. Further, $\mathbb{E}[\nabla f_{\mathbf{S}, \mathbf{J}}(x)] = \nabla f(x)$.

Thus, as a direct consequence of Theorem 6.3.4, we obtain the next corollary.

Corollary F.1.34. *Consider the setup from Lemma F.1.33. Suppose that f is μ -strongly convex and choose $\alpha \leq \min \left\{ \frac{1}{\mu}, \frac{1}{2\mathcal{L}_1 + M \frac{\mathcal{L}_2}{n}} \right\}$ where $M > \frac{2\lambda_{\max}}{n\lambda_{\min}}$. Then the iterates of JacSketch satisfy*

$$\mathbb{E} [V^k] \leq \max \left\{ (1 - \alpha\mu)^k, \left(1 + \frac{2\lambda_{\max}}{nM} - \lambda_{\min} \right)^k \right\} V^0. \quad (\text{F.54})$$

F.1.15 Interpolation between methods

Given that a set of stochastic gradients satisfy Assumption 6.3.1, we show that an any convex combination of the mentioned stochastic gradients satisfy Assumption 6.3.1 as well.

Lemma F.1.35. *Assume that sequences of stochastic gradients $\{g_1^k\}_{k \geq 0}, \dots, \{g_m^k\}_{k \geq 0}$ at the common iterates $\{x^k\}_{k \geq 0}$ satisfy the Assumption 6.3.1 with parameters*

$$A(j), B(j), \{\sigma_k^2(j)\}_{k \geq 0}, C(j), \rho(j), D_1(j), D_2(j), \quad j \in [m]$$

respectively. Then for any vector $\tau = (\tau_1, \dots, \tau_m)^\top$ such as $\sum_{j=1}^m \tau_j = 1$ and $\tau_j \geq 0, j \in [m]$

stochastic gradient $g_\tau^k = \sum_{j=1}^m \tau_j g_j^k$ satisfies the Assumption 6.3.1 with parameters:

$$\begin{aligned} A_\tau &= \sum_{j=1}^m \tau_j A(j), \quad B_\tau = 1, \quad \sigma_{\tau,k}^2 = \sum_{j=1}^m B(j) \tau_j \sigma_k^2(j), \quad \rho_\tau = \min_{j \in [m]} \rho(j), \\ C_\tau &= \sum_{j=1}^m \tau_j C(j) B(j), \quad D_{\tau,1} = \sum_{j=1}^m \tau_j D_1(j), \quad D_{\tau,2} = \sum_{j=1}^m \tau_j D_2(j) B(j). \end{aligned} \quad (\text{F.55})$$

Furthermore, if stochastic gradients g_1^k, \dots, g_m^k are independent for all k , Assumption 6.3.1 is satisfied with parameters

$$\begin{aligned} A_\tau &= L + \sum_{j=1}^m \tau_j^2 A(j), \quad B_\tau = 1, \quad \sigma_{\tau,k}^2 = \sum_{j=1}^m B(j) \tau_j^2 \sigma_k^2(j), \quad \rho_\tau = \min_{j \in [m]} \rho(j), \\ C_\tau &= \sum_{j=1}^m \tau_j^2 C(j) B(j), \quad D_{\tau,1} = \sum_{j=1}^m \tau_j^2 D_1(j), \quad D_{\tau,2} = \sum_{j=1}^m \tau_j^2 D_2(j) B(j). \end{aligned} \quad (\text{F.56})$$

What is more, instead of taking convex combination one can choose stochastic gradient at random. Lemma F.1.36 provides the result.

Lemma F.1.36. Assume that sequences of stochastic gradients $\{g_1^k\}_{k \geq 0}, \dots, \{g_m^k\}_{k \geq 0}$ at the common iterates $\{x^k\}_{k \geq 0}$ satisfy the Assumption 6.3.1 with parameters

$$A(j), B(j), \{\sigma_k^2(j)\}_{k \geq 0}, C(j), \rho(j), D_1(j), D_2(j), \quad j \in [m],$$

respectively. Then for any vector $\tau = (\tau_1, \dots, \tau_m)^\top$ such as $\sum_{j=1}^m \tau_j = 1$ and $\tau_j \geq 0, j \in [m]$ stochastic gradient g_τ^k which equals g_j^k with probability τ_j satisfies the Assumption 6.3.1 with parameters:

$$\begin{aligned} A_\tau &= \sum_{j=1}^m \tau_j A(j), \quad B_\tau = 1, \quad \sigma_{\tau,k}^2 = \sum_{j=1}^m \tau_j B(j) \sigma_k^2(j), \quad \rho_\tau = \min_{j \in [m]} \rho(j), \\ C_\tau &= \sum_{j=1}^m \tau_j B(j) C(j), \quad D_{\tau,1} = \sum_{j=1}^m \tau_j D_1(j), \quad D_{\tau,2} = \sum_{j=1}^m B(j) \tau_j D_2(j). \end{aligned} \quad (\text{F.57})$$

Furthermore, if stochastic gradients g_1^k, \dots, g_m^k are independent for all k , Assumption 6.3.1 is satisfied with parameters

$$\begin{aligned} A_\tau &= L + \sum_{j=1}^m \tau_j^2 A(j), \quad B_\tau = 1, \quad \sigma_{\tau,k}^2 = \sum_{j=1}^m B(j) \tau_j^2 \sigma_k^2(j), \quad \rho_\tau = \min_{j \in [m]} \rho(j), \\ C_\tau &= \sum_{j=1}^m \tau_j^2 C(j) B(j), \quad D_{\tau,1} = \sum_{j=1}^m \tau_j^2 D_1(j), \quad D_{\tau,2} = \sum_{j=1}^m \tau_j^2 D_2(j) B(j). \end{aligned} \quad (\text{F.58})$$

Example 15 (τ -L-SVRG). Consider the following method — τ -L-SVRG — which interpolates between vanilla SGD and LSVRG. When $\tau = 0$ the Algorithm 59 becomes LSVRG and when $\tau = 1$ it is just SGD with uniform sampling. Notice that Lemmas F.1.22 and F.1.4 still hold as they does not depend on the update rule for x^{k+1} .

Thus, sequences $\{g_{SGD}^k\}_{k \geq 0}$ and $\{g_{L-SVRG}^k\}_{k \geq 0}$ satisfy the conditions of Lemma F.1.35 and, as a consequence, stochastic gradient g^k from τ -L-SVRG meets the Assumption 6.3.1

Algorithm 59 τ -L-SVRG

Require: learning rate $\alpha > 0$, probability $p \in (0, 1]$, starting point $x^0 \in \mathbb{R}^d$, convex combination parameter $\tau \in [0, 1]$

$$w^0 = x^0$$

for $k = 0, 1, 2, \dots$ **do**

 Sample $i \in \{1, \dots, n\}$ uniformly at random

$$g_{\text{LSVRG}}^k = \nabla f_i(x^k) - \nabla f_i(w^k) + \nabla f(w^k)$$

 Sample $j \in \{1, \dots, n\}$ uniformly at random

$$g_{\text{SGD}}^k = \nabla f_j(x^k)$$

$$g^k = \tau g_{\text{SGD}}^k + (1 - \tau) g_{\text{LSVRG}}^k$$

$$x^{k+1} = x^k - \alpha g^k$$

$$w^{k+1} = \begin{cases} x^k & \text{with probability } p \\ w^k & \text{with probability } 1 - p \end{cases}$$

end for

with the following parameters:

$$A_\tau = L + 2\tau^2\mathcal{L} + 2(1 - \tau)^2L, \quad B_\tau = 1, \quad \sigma_{\tau,k}^2 = 2\frac{(1 - \tau)^2}{n} \sum_{i=1}^n \|\nabla f_i(w^k) - \nabla f_i(x^*)\|^2, \\ \rho_\tau = p, \quad C_\tau = 2(1 - \tau)^2Lp, \quad D_{\tau,1} = 2\tau^2\sigma^2, \quad D_{\tau,2} = 0.$$

Remark 32. Similar interpolation with the analogous analysis can be considered between SGD and SAGA, or SGD and SVRG.

Proof of Lemma F.1.35

Indeed, (6.7) holds due to linearity of mathematical expectation. Next, summing inequalities (6.8) for g_1^k, \dots, g_m^k and using convexity of $\|\cdot\|^2$ we get

$$\begin{aligned} \mathbb{E} \left[\|g_\tau^k - \nabla f(x^*)\|^2 \mid x^k \right] &\leq \sum_{j=1}^m \tau_j \mathbb{E} \left[\|g_j^k - \nabla f(x^*)\|^2 \mid x^k \right] \\ &\stackrel{(6.8)}{\leq} 2 \sum_{j=1}^m \tau_j A(j) D_f(x^k, x^*) + \sum_{j=1}^m B(j) \tau_j \sigma_k^2(j) + \sum_{j=1}^m \tau_j D_1(j), \end{aligned}$$

which implies (6.8) for g_τ^k with $A_\tau = \sum_{j=1}^m \tau_j A(j)$, $B_\tau = 1$, $\sigma_{\tau,k}^2 = \sum_{j=1}^m \tau_j B(j) \sigma_k^2(j)$, $D_{\tau,1} = \sum_{j=1}^m \tau_j D_1(j)$. Finally, summing (6.9) for g_1^k, \dots, g_m^k gives us

$$\mathbb{E} [\sigma_{\tau,k+1}^2 \mid \sigma_{\tau,k}^2] \stackrel{(6.9)}{\leq} \left(1 - \min_{j \in [m]} \rho(j) \right) \sigma_{\tau,k}^2 + 2 \sum_{j=1}^m \tau_j B(j) C(j) D_f(x^k, x^*) + \sum_{j=1}^m \tau_j B(j) D_2(j),$$

which is exactly (6.9) for $\sigma_{\tau,k}^2$ with $\rho = \min_{j \in [m]} \rho(j)$, $C_\tau = \sum_{j=1}^m \tau_j C(j)$, $D_{\tau,2} = \sum_{j=1}^m \tau_j D_2(j)$.

Next, for independent gradients we have

$$\begin{aligned}
& \mathbb{E} \left[\|g_\tau^k - \nabla f(x^*)\|^2 \mid x^k \right] \\
&= \sum_{j=1}^m \tau_j^2 \mathbb{E} \left[\|g_j^k - \nabla f(x^*)\|^2 \mid x^k \right] + 2 \sum_{i < j} \tau_i \tau_j \mathbb{E} \langle g_j^k - \nabla f(x^*), g_i^k - \nabla f(x^*) \rangle \\
&= \sum_{j=1}^m \tau_j^2 \mathbb{E} \left[\|g_j^k - \nabla f(x^*)\|^2 \mid x^k \right] + 2 \sum_{i < j} \tau_i \tau_j \|\nabla f(x^k) - \nabla f(x^*)\|^2 \\
&\leq \sum_{j=1}^m \tau_j^2 \mathbb{E} \left[\|g_j^k - \nabla f(x^*)\|^2 \mid x^k \right] + \left(\sum_{j=1}^m \tau_j \right)^2 \|\nabla f(x^k) - \nabla f(x^*)\|^2 \\
&= \sum_{j=1}^m \tau_j^2 \mathbb{E} \left[\|g_j^k - \nabla f(x^*)\|^2 \mid x^k \right] + \|\nabla f(x^k) - \nabla f(x^*)\|^2 \\
&\leq \sum_{j=1}^m \tau_j^2 \mathbb{E} \left[\|g_j^k - \nabla f(x^*)\|^2 \mid x^k \right] + 2LD_f(x^k, x^*). \tag{F.59}
\end{aligned}$$

and further the bounds follow.

Proof of Lemma F.1.36

Indeed, (6.7) holds due to linearity and tower property of mathematical expectation. Next, using tower property of mathematical expectation and inequalities (6.8) for g_1^k, \dots, g_m^k we get

$$\begin{aligned}
\mathbb{E} \left[\|g_\tau^k - \nabla f(x^*)\|^2 \mid x^k \right] &= \mathbb{E} \left[\mathbb{E}_\tau \left[\|g_\tau^k - \nabla f(x^*)\|^2 \mid x^k \right] \right] = \sum_{j=1}^m \tau_j \mathbb{E} \left[\|g_j^k - \nabla f(x^*)\|^2 \mid x^k \right] \\
&\stackrel{(6.8)}{\leq} 2 \sum_{j=1}^m \tau_j A(j) D_f(x^k, x^*) + \sum_{j=1}^m B(j) \tau_j \sigma_k^2(j) + \sum_{j=1}^m \tau_j D_1(j),
\end{aligned}$$

which implies (6.8) for g_τ^k with $A_\tau = \sum_{j=1}^m \tau_j A(j)$, $B_\tau = 1$, $\sigma_{\tau,k}^2 = \sum_{j=1}^m \tau_j B(j) \sigma_k^2(j)$, $D_{\tau,1} = \sum_{j=1}^m \tau_j D_1(j)$. Finally, summing (6.9) for g_1^k, \dots, g_m^k gives us

$$\mathbb{E} \left[\sigma_{\tau,k+1}^2 \mid \sigma_{\tau,k}^2 \right] \stackrel{(6.9)}{\leq} \left(1 - \min_{j \in [m]} \rho(j) \right) \sigma_{\tau,k}^2 + 2 \sum_{j=1}^m \tau_j B(j) C(j) D_f(x^k, x^*) + \sum_{j=1}^m \tau_j B(j) D_2(j),$$

which is exactly (6.9) for $\sigma_{\tau,k}^2$ with $\rho = \min_{j \in [m]} \rho(j)$, $C_\tau = \sum_{j=1}^m \tau_j B(j) C(j)$, $D_{\tau,2} = \sum_{j=1}^m \tau_j B(j) D_2(j)$.

To show (F.58), it suffices to combine above bounds with the trick (F.59).

Remark 33. Recently, [204] demonstrated in that the convex combination of SGD and SARAH [160] performs very well on non-convex problems.

F.2 Proofs for Section 6.3

F.2.1 Basic facts and inequalities

For all $a, b \in \mathbb{R}^d$ and $\xi > 0$ the following inequalities holds:

$$\langle a, b \rangle \leq \frac{\|a\|^2}{2\xi} + \frac{\xi \|b\|^2}{2}, \quad (\text{F.60})$$

$$\|a + b\|^2 \leq 2\|a\|^2 + 2\|b\|^2, \quad (\text{F.61})$$

and

$$\frac{1}{2} \|a\|^2 - \|b\|^2 \leq \|a + b\|^2. \quad (\text{F.62})$$

For a random vector $\xi \in \mathbb{R}^d$ and any $x \in \mathbb{R}^d$ the variance can be decomposed as

$$\mathbb{E} [\|\xi - \mathbb{E}\xi\|^2] = \mathbb{E} [\|\xi - x\|^2] - \|\mathbb{E}\xi - x\|^2. \quad (\text{F.63})$$

F.2.2 Proof of Lemma 6.3.3

We start with estimating the first term of the Lyapunov function. Let $r^k = x^k - x^*$. Then

$$\begin{aligned} \|r^{k+1}\|^2 &= \|\text{prox}_{\alpha\psi}(x^k - \alpha g^k) - \text{prox}_{\alpha\psi}(x^* - \alpha \nabla f(x^*))\|^2 \\ &\leq \|x^k - x^* - \alpha(g^k - \nabla f(x^*))\|^2 \\ &= \|r^k\|^2 - 2\alpha \langle r^k, g^k - \nabla f(x^*) \rangle + \alpha^2 \|g^k - \nabla f(x^*)\|^2. \end{aligned}$$

Taking expectation conditioned on x^k we get

$$\begin{aligned} \mathbb{E} [\|r^{k+1}\|^2 \mid x^k] &= \|r^k\|^2 - 2\alpha \langle r^k, \nabla f(x^k) - \nabla f(x^*) \rangle + \alpha^2 \mathbb{E} [\|g^k - \nabla f(x^*)\|^2 \mid x^k] \\ &\stackrel{(6.12)}{\leq} (1 - \alpha\mu) \|r^k\|^2 - 2\alpha D_f(x^k, x^*) + \alpha^2 \mathbb{E} [\|g^k - \nabla f(x^*)\|^2 \mid x^k] \\ &\stackrel{(6.7)+(6.8)}{\leq} (1 - \alpha\mu) \|r^k\|^2 + 2\alpha (A\alpha - 1) D_f(x^k, x^*) + B\alpha^2 \sigma_k^2 + \alpha^2 D_1. \end{aligned}$$

Using this we estimate the full expectation of V^{k+1} in the following way:

$$\begin{aligned} &\mathbb{E} \|x^{k+1} - x^*\|^2 + M\alpha^2 \mathbb{E} \sigma_{k+1}^2 \\ &\stackrel{(6.9)}{\leq} (1 - \alpha\mu) \mathbb{E} \|x^k - x^*\|^2 + 2\alpha (A\alpha - 1) D_f(x^k, x^*) + B\alpha^2 \mathbb{E} \sigma_k^2 \\ &\quad + (1 - \rho) M\alpha^2 \mathbb{E} \sigma_k^2 + 2CM\alpha^2 \mathbb{E} [D_f(x^k, x^*)] + (D_1 + MD_2)\alpha^2 \\ &= (1 - \alpha\mu) \mathbb{E} \|x^k - x^*\|^2 + \left(1 + \frac{B}{M} - \rho\right) M\alpha^2 \mathbb{E} \sigma_k^2 \\ &\quad + 2\alpha (\alpha(A + CM) - 1) \mathbb{E} [D_f(x^k, x^*)] + (D_1 + MD_2)\alpha^2. \end{aligned}$$

It remains to rearrange the terms.

F.2.3 Proof of Theorem 6.3.4

Note first that due to (6.13) we have $2\alpha(1 - \alpha(A + CM))\mathbb{E}D_f(x^k, x^*) > 0$, thus we can omit the term.

Unrolling the recurrence from Lemma 6.3.3 and using the Lyapunov function notation gives us

$$\begin{aligned}
 \mathbb{E}V^k &\leq \max \left\{ (1 - \alpha\mu)^k, \left(1 + \frac{B}{M} - \rho\right)^k \right\} V^0 \\
 &\quad + (D_1 + MD_2)\alpha^2 \sum_{l=0}^{k-1} \max \left\{ (1 - \alpha\mu)^l, \left(1 + \frac{B}{M} - \rho\right)^l \right\} \\
 &\leq \max \left\{ (1 - \alpha\mu)^k, \left(1 + \frac{B}{M} - \rho\right)^k \right\} V^0 \\
 &\quad + (D_1 + MD_2)\alpha^2 \sum_{l=0}^{\infty} \max \left\{ (1 - \alpha\mu)^l, \left(1 + \frac{B}{M} - \rho\right)^l \right\} \\
 &\leq \max \left\{ (1 - \alpha\mu)^k, \left(1 + \frac{B}{M} - \rho\right)^k \right\} V^0 + \frac{(D_1 + MD_2)\alpha^2}{\min \left\{ \alpha\mu, \rho - \frac{B}{M} \right\}}.
 \end{aligned}$$

Appendix G

Appendix for Chapter 7

G.1 Missing lemmas and proofs: SAGA/LSVRG is a special case of SEGA/SVRCD

G.1.1 Proof of Lemma 7.4.3

Let $\mathbf{W}' \stackrel{\text{def}}{=} \frac{1}{n} \tilde{\mathbf{e}} \tilde{\mathbf{e}}^\top \otimes \mathbf{I}$ and denote $\mathbf{D}_\mathbf{B}(\tilde{\mathbf{M}}) \stackrel{\text{def}}{=} \text{BlockDiag}(\tilde{\mathbf{M}}_1, \dots, \tilde{\mathbf{M}}_n)$ for simplicity. Now clearly $x^0 \in \text{Range}(\mathbf{W}')$, while \mathbf{W}' is a projection matrix such that $I(x) < \infty$ if and only if $\mathbf{W}'x = x$. Consequently, $\mathbf{W} = \mathbf{W}'$. Next, if $x, y \in \text{Range}(\mathbf{W})$, there is $\tilde{x}, \tilde{y} \in \mathbb{R}^{\tilde{d}}$ such that $x = U(\tilde{x}), y = U(\tilde{y})$. Therefore we can write

$$\begin{aligned} f(x) &= f(\mathbf{W}(x)) = \frac{1}{n} \sum_{j=1}^n \tilde{f}_j(\tilde{x}) \\ &\geq \frac{1}{n} \sum_{j=1}^n \tilde{f}_j(\tilde{y}) + \left\langle \nabla \left(\frac{1}{n} \sum_{j=1}^n \tilde{f}_j(\tilde{y}) \right), \tilde{x} - \tilde{y} \right\rangle + \frac{\tilde{\mu}}{2} \|\tilde{x} - \tilde{y}\|^2 \\ &= f(y) + \langle \nabla f(y), x - y \rangle + \frac{\tilde{\mu}}{2n} \|x - y\|^2. \end{aligned}$$

Similarly,

$$\begin{aligned} f(x) &= \frac{1}{n} \sum_{j=1}^n \tilde{f}_j(\tilde{x}) \\ &\leq \frac{1}{n} \sum_{j=1}^n \tilde{f}_j(\tilde{y}) + \left\langle \nabla \left(\frac{1}{n} \sum_{j=1}^n \tilde{f}_j(\tilde{y}) \right), \tilde{x} - \tilde{y} \right\rangle + \sum_{j=1}^n \frac{1}{2n} \|\tilde{x} - \tilde{y}\|_{\tilde{\mathbf{M}}_j}^2 \\ &= f(y) + \langle \nabla f(y), x - y \rangle + \frac{1}{2n} \|x - y\|_{\mathbf{D}_\mathbf{B}(\tilde{\mathbf{M}})}^2. \end{aligned}$$

Thus we conclude $\mu = \frac{\tilde{\mu}}{n}$ and $\mathbf{M} = \frac{1}{n}\mathbf{D}_B(\tilde{\mathbf{M}})$. Further, for any $h \in \mathbb{R}^d$, we have:

$$\begin{aligned}
& h^\top \mathbf{M}^{\frac{1}{2}} \mathbb{E} \left[\sum_{i \in S} \mathbf{p}_i^{-1} \mathbf{e}_i \mathbf{e}_i^\top \mathbf{W} \sum_{i \in S} \mathbf{p}_i^{-1} \mathbf{e}_i \mathbf{e}_i^\top \right] \mathbf{M}^{\frac{1}{2}} h \\
&= \frac{1}{n} \|\mathbf{D}_B(\tilde{\mathbf{M}})^{\frac{1}{2}} h\|^2 \mathbb{E} \left[\left(\sum_{i \in \tilde{S}} \tilde{\mathbf{p}}_i^{-1} \left(\sum_{j \in R_i} \mathbf{e}_j \mathbf{e}_j^\top \right) \right) \mathbf{W} \left(\sum_{i \in \tilde{S}} \tilde{\mathbf{p}}_i^{-1} \left(\sum_{j \in R_i} \mathbf{e}_j \mathbf{e}_j^\top \right) \right) \right] \\
&= \frac{1}{n} \mathbb{E} \left[\left\| \sum_{i \in \tilde{S}} \tilde{\mathbf{M}}_i^{\frac{1}{2}} \tilde{\mathbf{p}}_i^{-1} h_{R_i} \right\|^2 \right] \\
&\stackrel{(7.12)}{\leq} \frac{1}{n} \sum_{i=1}^n \tilde{\mathbf{p}}_i \tilde{\mathbf{v}}_i \|h_{R_i}\|^2
\end{aligned}$$

and thus (7.6) holds with $v = \frac{1}{n} \tilde{\mathbf{v}}$ as desired.

G.1.2 Proof of Lemma 7.4.4

Denote $\text{Vec}(\cdot)$ to be the vectorization operator, i.e., operator which takes a matrix as an input, and returns a vector constructed by a column-wise stacking of the matrix columns. We will show both

$$h^k = \frac{1}{n} \text{Vec}(\mathbf{J}^k) \quad (\text{G.1})$$

and (7.14) using mathematical induction. Clearly, if $k = 0$ both (G.1) and (7.14) hold. Now, let us proceed with the second induction step.

$$\begin{aligned}
x^{k+1} &= \text{prox}_{\alpha\psi}(x^k - \alpha g^k) = \arg \min_{x \in \mathbb{R}^d} \alpha I(x) + \alpha \tilde{\psi}(x_{R_1}) + \|x - (x^k - \alpha g^k)\|^2 \\
&= \arg \min_{x \in \mathbb{R}^d} \alpha I(x) + \alpha \tilde{\psi}(x_{R_1}) + \left\| x - x^k + \alpha \left(h^k + \sum_{i \in S} \frac{1}{\mathbf{p}_i} (\nabla_i f(x^k) - h_i^k) \mathbf{e}_i \right) \right\|^2 \\
&= \arg \min_{x = \mathbf{W}x} \alpha \tilde{\psi}(x_{R_1}) + \left\| x - x^k + \alpha \left(h^k + \sum_{i \in S} \frac{1}{\mathbf{p}_i} (\nabla_i f(x^k) - h_i^k) \mathbf{e}_i \right) \right\|^2 \\
&= \arg \min_{x = \mathbf{W}x} \alpha \tilde{\psi}(x_{R_1}) + \left\| x - x^k + \alpha \left(h^k + \sum_{i \in S} \frac{1}{\mathbf{p}_i} (\nabla_i f(x^k) - h_i^k) \mathbf{e}_i \right) \right\|_{\mathbf{W}}^2 \\
&\stackrel{(7.14)}{=} U \left(\arg \min_{\tilde{x} \in \mathbb{R}^d} \alpha \tilde{\psi}(\tilde{x}) + \frac{1}{n} \left\| n\tilde{x} - n\tilde{x}^k + \alpha \left(\sum_{i=1}^n h_{R_i}^k + \sum_{i \in \tilde{S}} \frac{1}{\tilde{\mathbf{p}}_i} \left(\frac{1}{n} \nabla \tilde{f}_i(\tilde{x}^k) - h_{R_i}^k \right) \right) \right\|^2 \right) \\
&\stackrel{(\text{G.1})}{=} U \left(\arg \min_{\tilde{x} \in \mathbb{R}^d} \alpha \tilde{\psi}(\tilde{x}) + \frac{1}{n} \left\| n\tilde{x} - n\tilde{x}^k + \alpha \left(\frac{1}{n} \mathbf{J}^k \tilde{\mathbf{e}} + \frac{1}{n} \sum_{i \in \tilde{S}} \frac{1}{\tilde{\mathbf{p}}_i} \left((\nabla \tilde{f}_i(\tilde{x}^k) - \mathbf{J}_{:,i}^k) \right) \right) \right\|^2 \right) \\
&= U \left(\arg \min_{\tilde{x} \in \mathbb{R}^d} \tilde{\alpha} \tilde{\psi}(\tilde{x}) + \left\| \tilde{x} - \tilde{x}^k + \tilde{\alpha} \left(\frac{1}{n} \mathbf{J}^k \tilde{\mathbf{e}} + \frac{1}{n} \sum_{i \in \tilde{S}} \frac{1}{\tilde{\mathbf{p}}_i} \left((\nabla \tilde{f}_i(\tilde{x}^k) - \mathbf{J}_{:,i}^k) \right) \right) \right\|^2 \right) \\
&= U(\tilde{x}^{k+1}). \quad (\text{G.2})
\end{aligned}$$

It remains to notice that since $x^{k+1} = U(\tilde{x}^k)$, we have $h^{k+1} = \frac{1}{n} \text{Vec}(\mathbf{J}^{k+1})$ as desired.

G.2 Missing lemmas and proofs: ASVRCD

G.2.1 Technical lemmas

We first start with two key technical lemmas.

Lemma G.2.1. *Suppose that*

$$\eta \leq \frac{1}{2L}. \quad (\text{G.3})$$

Then, for all $x \in \text{Range}(\mathbf{W})$ the following inequality holds:

$$\begin{aligned} & \frac{1}{\eta} \mathbb{E} [\langle x - x^k, x^k - y^{k+1} \rangle] \\ & \leq \mathbb{E} \left[F(x) - F(y^{k+1}) - \frac{1}{4\eta} \|y^{k+1} - x^k\|^2 + \frac{\eta}{2} \|g^k - \nabla f(x^k)\|_{\mathbf{W}}^2 \right] \\ & \quad - D_f(x, x^k). \end{aligned} \quad (\text{G.4})$$

Proof. From the definition of y^{k+1} we get

$$y^{k+1} = x^k - \eta g^k - \eta \Delta,$$

where $\Delta \in \partial\psi(y^{k+1})$. Therefore,

$$\begin{aligned} & \mathbb{E} \left[\frac{1}{\eta} \langle x - x^k, x^k - y^{k+1} \rangle \right] \\ & = \mathbb{E} [\langle x - x^k, g^k + \Delta \rangle] \\ & = \langle x - x^k, \nabla f(x^k) \rangle + \mathbb{E} [\langle x - y^{k+1}, \Delta \rangle + \langle y^{k+1} - x^k, \Delta \rangle] \\ & \leq f(x) - f(x^k) - D_f(x, x^k) + \mathbb{E} [\psi(x) - \psi(y^{k+1})] + \mathbb{E} [\langle y^{k+1} - x^k, \Delta \rangle] \end{aligned} \quad (\text{G.5})$$

Now, we use the fact that f is L -smooth over the set where iterates live (i.e., over $\{x^0 + \text{Range}(\mathbf{W})\}$):

$$\begin{aligned} f(y^{k+1}) & \leq f(x^k) + \langle \nabla f(x^k), y^{k+1} - x^k \rangle + \frac{L}{2} \|y^{k+1} - x^k\|^2 \\ & = f(x^k) + \langle \mathbf{W} \nabla f(x^k), y^{k+1} - x^k \rangle + \frac{L}{2} \|y^{k+1} - x^k\|^2. \end{aligned} \quad (\text{G.6})$$

Thus, we have

$$\begin{aligned}
& \mathbb{E} \left[\frac{1}{\eta} \langle x - x^k, x^k - y^{k+1} \rangle \right] \\
& \stackrel{(G.5)+(G.6)}{\leq} \mathbb{E} \left[F(x) - F(y^{k+1}) + \langle y^{k+1} - x^k, \mathbf{W}(\Delta + \nabla f(x^k)) \rangle + \frac{L}{2} \|y^{k+1} - x^k\|^2 \right] \\
& \quad - D_f(x, x^k) \\
& = \mathbb{E} \left[F(x) - F(y^{k+1}) + \langle y^{k+1} - x^k, \mathbf{W}(\nabla f(x^k) - g^k) \rangle - \frac{1}{\eta} \|y^{k+1} - x^k\|^2 \right] \\
& \quad + \mathbb{E} \left[\frac{L}{2} \|y^{k+1} - x^k\|^2 \right] - D_f(x, x^k) \\
& \leq \mathbb{E} \left[F(x) - F(y^{k+1}) + \frac{\eta}{2} \|\nabla f(x^k) - g^k\|_{\mathbf{W}}^2 - \frac{1}{2\eta} \|y^{k+1} - x^k\|^2 + \frac{L}{2} \|y^{k+1} - x^k\|^2 \right] \\
& \quad - D_f(x, x^k) \\
& \stackrel{(G.3)}{\leq} \mathbb{E} \left[F(x) - F(y^{k+1}) - \frac{1}{4\eta} \|y^{k+1} - x^k\|^2 + \frac{\eta}{2} \|\nabla f(x^k) - g^k\|_{\mathbf{W}}^2 \right] - D_f(x, x^k),
\end{aligned}$$

which concludes the proof. \square

Lemma G.2.2. *Suppose, the following choice of parameters is used:*

$$\eta = \frac{1}{4} \max\{\mathcal{L}', L\}^{-1}, \quad \gamma = \frac{1}{\max\{2\mu, 4\theta_1/\eta\}}, \quad \beta = 1 - \gamma\mu, \quad \theta_2 = \frac{\mathcal{L}'}{2 \max\{L, \mathcal{L}'\}}.$$

Then the following inequality holds:

$$\begin{aligned}
& \mathbb{E} \left[\|z^{k+1} - x^*\|^2 + \frac{2\gamma\beta}{\theta_1} [F(y^{k+1}) - F(x^*)] \right] \\
& \leq \beta \|z^k - x^*\|^2 + \frac{2\gamma\beta\theta_2}{\theta_1} [F(w^k) - F(x^*)] + \frac{2\gamma\beta(1 - \theta_1 - \theta_2)}{\theta_1} [F(y^k) - F(x^*)].
\end{aligned} \tag{G.7}$$

Proof.

$$\begin{aligned}
& \mathbb{E} \left[\|z^{k+1} - x^*\|^2 \right] \\
&= \mathbb{E} \left[\left\| \beta z^k + (1 - \beta)x^k - x^* + \frac{\gamma}{\eta}(y^{k+1} - x^k) \right\|^2 \right] \\
&\leq \beta \|z^k - x^*\|^2 + (1 - \beta) \|x^k - x^*\|^2 + \frac{\gamma^2}{\eta^2} \mathbb{E} \left[\|y^{k+1} - x^k\|^2 \right] \\
&\quad + \frac{2\gamma}{\eta} \mathbb{E} [\langle y^{k+1} - x^k, \beta z^k + (1 - \beta)x^k - x^* \rangle] \\
&= \beta \|z^k - x^*\|^2 + (1 - \beta) \|x^k - x^*\|^2 + \frac{\gamma^2}{\eta^2} \mathbb{E} \left[\|y^{k+1} - x^k\|^2 \right] \\
&\quad + \frac{2\gamma}{\eta} \mathbb{E} [\langle y^{k+1} - x^k, x^k - x^* \rangle] + \frac{2\gamma\beta}{\eta} \mathbb{E} [\langle y^{k+1} - x^k, z^k - x^k \rangle] \\
&= \beta \|z^k - x^*\|^2 + (1 - \beta) \|x^k - x^*\|^2 + \frac{\gamma^2}{\eta^2} \mathbb{E} \left[\|y^{k+1} - x^k\|^2 \right] + \frac{2\gamma}{\eta} \mathbb{E} [\langle x^k - y^{k+1}, x^* - x^k \rangle] \\
&\quad + \frac{2\gamma\beta\theta_2}{\eta\theta_1} \mathbb{E} [\langle x^k - y^{k+1}, w^k - x^k \rangle] + \frac{2\gamma\beta(1 - \theta_1 - \theta_2)}{\eta\theta_1} \mathbb{E} [\langle x^k - y^{k+1}, y^k - x^k \rangle] \\
&\stackrel{(G.4)}{\leq} \beta \|z^k - x^*\|^2 + (1 - \beta) \|x^k - x^*\|^2 + \frac{\gamma^2}{\eta^2} \mathbb{E} \left[\|y^{k+1} - x^k\|^2 \right] \\
&\quad + 2\gamma \mathbb{E} \left[F(x^*) - F(y^{k+1}) - \frac{1}{4\eta} \|y^{k+1} - x^k\|^2 - D_f(x^*, x^k) + \frac{\eta}{2} \|g^k - \nabla f(x^k)\|_{\mathbf{W}}^2 \right] \\
&\quad + \frac{2\gamma\beta\theta_2}{\theta_1} \mathbb{E} \left[F(w^k) - F(y^{k+1}) - \frac{1}{4\eta} \|y^{k+1} - x^k\|^2 - D_f(w^k, x^k) + \frac{\eta}{2} \|g^k - \nabla f(x^k)\|_{\mathbf{W}}^2 \right] \\
&\quad + \frac{2\gamma\beta(1 - \theta_1 - \theta_2)}{\theta_1} \mathbb{E} \left[F(y^k) - F(y^{k+1}) - \frac{1}{4\eta} \|y^{k+1} - x^k\|^2 + \frac{\eta}{2} \|g^k - \nabla f(x^k)\|_{\mathbf{W}}^2 \right] \\
&\stackrel{(7.3)}{\leq} \beta \|z^k - x^*\|^2 + (1 - \beta - \gamma\mu) \|x^k - x^*\|^2 + \frac{\gamma^2}{\eta^2} \mathbb{E} \left[\|y^{k+1} - x^k\|^2 \right] \\
&\quad + 2\gamma\beta \mathbb{E} \left[F(x^*) - F(y^{k+1}) - \frac{1}{4\eta} \|y^{k+1} - x^k\|^2 \right] + \eta\gamma \mathbb{E} \left[\|g^k - \nabla f(x^k)\|_{\mathbf{W}}^2 \right] \\
&\quad + \frac{2\gamma\beta\theta_2}{\theta_1} \mathbb{E} \left[F(w^k) - F(y^{k+1}) - \frac{1}{4\eta} \|y^{k+1} - x^k\|^2 - D_f(w^k, x^k) + \frac{\eta}{2} \|g^k - \nabla f(x^k)\|_{\mathbf{W}}^2 \right] \\
&\quad + \frac{2\gamma\beta(1 - \theta_1 - \theta_2)}{\theta_1} \mathbb{E} \left[F(y^k) - F(y^{k+1}) - \frac{1}{4\eta} \|y^{k+1} - x^k\|^2 + \frac{\eta}{2} \|g^k - \nabla f(x^k)\|_{\mathbf{W}}^2 \right].
\end{aligned}$$

Using $\beta = 1 - \gamma\mu$ we get

$$\begin{aligned}
\mathbb{E} \left[\|z^{k+1} - x^*\|^2 \right] &\leq \beta \|z^k - x^*\|^2 + \left[\frac{\gamma^2}{\eta^2} - \frac{\gamma\beta}{2\eta\theta_1} \right] \mathbb{E} \left[\|y^{k+1} - x^k\|^2 \right] + \frac{\eta\gamma}{\theta_1} \mathbb{E} \left[\|g^k - \nabla f(x^k)\|_{\mathbf{W}}^2 \right] \\
&\quad - \frac{2\gamma\beta\theta_2}{\theta_1} D_f(w^k, x^k) + 2\gamma\beta \mathbb{E} [F(x^*) - F(y^{k+1})] \\
&\quad + \frac{2\gamma\beta\theta_2}{\theta_1} \mathbb{E} [F(w^k) - F(y^{k+1})] + \frac{2\gamma\beta(1 - \theta_1 - \theta_2)}{\theta_1} \mathbb{E} [F(y^k) - F(y^{k+1})].
\end{aligned}$$

Using stepsize $\gamma \leq \frac{\beta\eta}{2\theta_1}$ we get

$$\begin{aligned} \mathbb{E} \left[\|z^{k+1} - x^*\|^2 \right] &\leq \beta \|z^k - x^*\|^2 + \frac{\eta\gamma}{\theta_1} \mathbb{E} \left[\|g^k - \nabla f(x^k)\|_{\mathbf{W}}^2 \right] - \frac{2\gamma\beta\theta_2}{\theta_1} D_f(w^k, x^k) \\ &\quad + 2\gamma\beta \mathbb{E} [F(x^*) - F(y^{k+1})] + \frac{2\gamma\beta\theta_2}{\theta_1} \mathbb{E} [F(w^k) - F(y^{k+1})] + \\ &\quad \frac{2\gamma\beta(1 - \theta_1 - \theta_2)}{\theta_1} \mathbb{E} [F(y^k) - F(y^{k+1})]. \end{aligned}$$

Now, using the expected smoothness from inequality (7.16):

$$\mathbb{E} \left[\|g^k - \nabla f(x^k)\|_{\mathbf{W}}^2 \right] \leq 2\mathcal{L}' D_f(w^k, x^k) \quad (\text{G.8})$$

and stepsize $\eta \leq \frac{\beta\theta_2}{\mathcal{L}'}$ we get

$$\begin{aligned} \mathbb{E} \left[\|z^{k+1} - x^*\|^2 \right] &\leq \beta \|z^k - x^*\|^2 + \frac{2\mathcal{L}'\eta\gamma}{\theta_1} D_f(w^k, x^k) - \frac{2\gamma\beta\theta_2}{\theta_1} D_f(w^k, x^k) + 2\gamma\beta \mathbb{E} [F(x^*) - F(y^{k+1})] \\ &\quad + \frac{2\gamma\beta\theta_2}{\theta_1} \mathbb{E} [F(w^k) - F(y^{k+1})] + \frac{2\gamma\beta(1 - \theta_1 - \theta_2)}{\theta_1} \mathbb{E} [F(y^k) - F(y^{k+1})] \\ &\leq \beta \|z^k - x^*\|^2 + 2\gamma\beta \mathbb{E} [F(x^*) - F(y^{k+1})] + \frac{2\gamma\beta\theta_2}{\theta_1} \mathbb{E} [F(w^k) - F(y^{k+1})] \\ &\quad + \frac{2\gamma\beta(1 - \theta_1 - \theta_2)}{\theta_1} \mathbb{E} [F(y^k) - F(y^{k+1})] \\ &= \beta \|z^k - x^*\|^2 - \frac{2\gamma\beta}{\theta_1} \mathbb{E} [F(y^{k+1}) - F(x^*)] + \frac{2\gamma\beta\theta_2}{\theta_1} [F(w^k) - F(x^*)] \\ &\quad + \frac{2\gamma\beta(1 - \theta_1 - \theta_2)}{\theta_1} [F(y^k) - F(x^*)]. \end{aligned}$$

It remains to rearrange the terms. □

G.2.2 Proof of Theorem 7.5.1

One can easily show that

$$\mathbb{E} [F(w^{k+1})] = \rho F(y^k) + (1 - \rho) F(w^k). \quad (\text{G.9})$$

Using that, we obtain

$$\begin{aligned}
\mathbb{E} [\Psi^{k+1}] &\stackrel{(G.7)+(G.9)}{\leq} \beta \|z^k - x^*\|^2 + \frac{2\gamma\beta\theta_2}{\theta_1} [F(w^k) - F(x^*)] + \frac{2\gamma\beta(1-\theta_1-\theta_2)}{\theta_1} [F(y^k) - F(x^*)] \\
&\quad + \frac{(2\theta_2 + \theta_1)\gamma\beta}{\theta_1\rho} [\rho F(y^k) + (1-\rho)F(w^k) - F(x^*)] \\
&= \beta \|z^k - x^*\|^2 + \frac{2\gamma\beta(1-\theta_1/2)}{\theta_1} [F(y^k) - F(x^*)] \\
&\quad + \frac{(2\theta_2 + \theta_1)\gamma\beta}{\theta_1\rho} \left[1 - \rho + \frac{2\rho\theta_2}{2\theta_2 + \theta_1}\right] [F(w^k) - F(x^*)] \\
&\leq \max \left\{1 - \frac{1}{\max\{2, 4\theta_1/(\eta\mu)\}}, 1 - \frac{\theta_1}{2}, 1 - \frac{\rho\theta_1}{2\max\{2\theta_2, \theta_1\}}\right\} \Psi^k \\
&= \left[1 - \max \left\{\frac{2}{\rho}, \frac{4}{\theta_1} \max \left\{\frac{1}{2}, \frac{\theta_2}{\rho}\right\}, \frac{4\theta_1}{\eta\mu}\right\}^{-1}\right] \Psi^k.
\end{aligned}$$

Using $\theta_1 = \min \left\{\frac{1}{2}, \sqrt{\eta\mu \max \left\{\frac{1}{2}, \frac{\theta_2}{\rho}\right\}}\right\}$ we get

$$\begin{aligned}
\mathbb{E} [\Psi^{k+1}] &\leq \left[1 - \max \left\{\frac{2}{\rho}, 8 \max \left\{\frac{1}{2}, \frac{\theta_2}{\rho}\right\}, 4\sqrt{\frac{\max \left\{\frac{1}{2}, \frac{\theta_2}{\rho}\right\}}{\eta\mu}}\right\}^{-1}\right] \Psi^k \\
&\leq \left[1 - \frac{1}{4} \max \left\{\frac{1}{\rho}, \sqrt{\frac{2 \max \left\{L, \frac{\mathcal{L}'}{\rho}\right\}}{\mu}}\right\}^{-1}\right] \Psi^k,
\end{aligned}$$

as desired.

G.2.3 Proof of Lemma 7.5.2

To establish that that we can choose $\mathcal{L}' = \mathcal{L}$, it suffices to see

$$\begin{aligned}
\mathbb{E} \left[\left\| g^k - \nabla f(x^k) \right\|_{\mathbf{W}}^2 \right] &= \mathbb{E} \left[\left\| \sum_{i \in S} \frac{1}{\mathbf{p}_i} (\nabla_i f(x^k) - \nabla_i f(w^k)) e_i + \nabla f(w^k) - \nabla f(x^k) \right\|_{\mathbf{W}}^2 \right] \\
&\leq \mathbb{E} \left[\left\| \sum_{i \in S} \frac{1}{\mathbf{p}_i} (\nabla_i f(x^k) - \nabla_i f(w^k)) e_i \right\|_{\mathbf{W}}^2 \right] \\
&\stackrel{(7.7)}{\leq} \mathcal{L} \left\| \nabla f(x^k) - \nabla f(w^k) \right\|_{\mathbf{M}^{-1}}^2 \\
&\stackrel{(G.17)}{\leq} 2\mathcal{L} D_f(w^k, x^k).
\end{aligned}$$

Next, to establish $\mathcal{L} \geq L$, let $\mathbf{Q} \stackrel{\text{def}}{=} \sum_{i \in S} \frac{1}{\mathbf{p}_i} \mathbf{e}_i \mathbf{e}_i^\top \mathbf{W}$. Consequently, we get

$$\begin{aligned}
\mathcal{L} &\stackrel{(7.7)}{=} \lambda_{\max} \left(\mathbf{M}^{\frac{1}{2}} \mathbb{E} \left[\sum_{i \in S} \frac{1}{p_i} \mathbf{e}_i \mathbf{e}_i^\top \mathbf{W} \sum_{i \in S} \frac{1}{p_i} \mathbf{e}_i \mathbf{e}_i^\top \right] \mathbf{M}^{\frac{1}{2}} \right) \\
&= \lambda_{\max} \left(\mathbf{M}^{\frac{1}{2}} \mathbb{E} \left[\sum_{i \in S} \frac{1}{p_i} \mathbf{e}_i \mathbf{e}_i^\top \mathbf{W}^2 \sum_{i \in S} \frac{1}{p_i} \mathbf{e}_i \mathbf{e}_i^\top \right] \mathbf{M}^{\frac{1}{2}} \right) \\
&= \lambda_{\max} \left(\mathbf{M}^{\frac{1}{2}} \mathbb{E} [\mathbf{Q} \mathbf{Q}^\top] \mathbf{M}^{\frac{1}{2}} \right) \\
&\geq \lambda_{\max} \left(\mathbf{M}^{\frac{1}{2}} \mathbb{E} [\mathbf{Q}] \mathbb{E} [\mathbf{Q}^\top] \mathbf{M}^{\frac{1}{2}} \right) \\
&= \lambda_{\max} \left(\mathbf{M}^{\frac{1}{2}} \mathbf{W}^2 \mathbf{M}^{\frac{1}{2}} \right) \\
&= \lambda_{\max} \left(\mathbf{M}^{\frac{1}{2}} \mathbf{W} \mathbf{M}^{\frac{1}{2}} \right) \\
&= L,
\end{aligned}$$

as desired.

G.2.4 Proof of Lemma 7.5.3

Let us look first at $\mathbf{W} = \mathbf{I}$. In such case, it is easy to see that

$$\begin{aligned}
\mathbb{E} \left[\|g^k - \nabla f(x^k)\|_{\mathbf{W}}^2 \right] &= \mathbb{E} \left[\|g^k - \nabla f(x^k)\|^2 \right] \\
&\leq \mathbb{E} \left[\|d(\nabla_i f(x^k) - \nabla_i f(w^k)) e_i\|^2 \right] \\
&= d \|\nabla f(x^k) - \nabla f(w^k)\|^2 \\
&\leq 2d \lambda_{\max} \mathbf{M} D_f(w^k, x^k),
\end{aligned}$$

i.e., we can choose $\mathcal{L}' = d \lambda_{\max} \mathbf{M}$. Noting that $\lambda_{\max} \mathbf{M} \geq L$, the iteration complexity of Algorithm 17 is $\mathcal{O} \left(d \sqrt{\frac{L}{\mu}} \log \frac{1}{\epsilon} \right)$. On the other hand, if $\mathbf{W} = \frac{1}{d} \tilde{\mathbf{e}} \tilde{\mathbf{e}}^\top$, we have

$$\begin{aligned}
\mathbb{E} \left[\|g^k - \nabla f(x^k)\|_{\mathbf{W}}^2 \right] &= \mathbb{E} \left[\|g^k - \nabla f(x^k)\|_{\frac{1}{d} \tilde{\mathbf{e}} \tilde{\mathbf{e}}^\top}^2 \right] \\
&\leq \mathbb{E} \left[\|d(\nabla_i f(x^k) - \nabla_i f(w^k)) e_i\|_{\frac{1}{d} \tilde{\mathbf{e}} \tilde{\mathbf{e}}^\top}^2 \right] \\
&= \|\nabla f(x^k) - \nabla f(w^k)\|^2 \\
&\leq 2 \lambda_{\max} \mathbf{M} D_f(w^k, x^k),
\end{aligned}$$

and therefore $\mathcal{L}' = \lambda_{\max} \mathbf{M}$, which yields $\mathcal{O} \left(\sqrt{\frac{d \lambda_{\max} \mathbf{M}}{\mu}} \log \frac{1}{\epsilon} \right)$ convergence rate.

G.3 Missing lemmas and proofs: L-Katyusha as a particular case of ASVRCD

G.3.1 Proof of Lemma 7.6.3

Let us proceed by induction. We will show the following for all $k \geq 0$ we have

$$\begin{aligned} \tilde{x}^k &= x_{R_1}^k = \dots = x_{R_n}^k, & \tilde{y}^k &= y_{R_1}^k = \dots = y_{R_n}^k, \\ \tilde{z}^k &= z_{R_1}^k = \dots = z_{R_n}^k & \text{and} & \tilde{w}^k = w_{R_1}^k = \dots = w_{R_n}^k. \end{aligned} \quad (\text{G.10})$$

Clearly, for $k = 0$, the above claim holds. Let us proceed with the second induction step and assume that (G.10) holds for some $k \geq 0$. First, the update rule on $\{x^k\}$ for ASVRCD together with the update rule on $\{\tilde{x}^k\}$ yields

$$\tilde{x}^{k+1} = x_{R_1}^{k+1} = \dots = x_{R_n}^{k+1}. \quad (\text{G.11})$$

To show

$$\tilde{y}^{k+1} = y_{R_1}^{k+1} = \dots = y_{R_n}^{k+1}, \quad (\text{G.12})$$

we essentially repeat the proof of Lemma 7.4.4. In particular, it is sufficient to repeat the sequence of inequalities (G.2) where variables

$$(x^{k+1}, \tilde{x}^{k+1}, h^k, \mathbf{J}^k \alpha, \tilde{\alpha})$$

are replaced by

$$(y^{k+1}, \tilde{y}^{k+1}, \nabla f(w^k), [\nabla \tilde{f}_1(\tilde{w}^k), \dots, \nabla \tilde{f}_n(\tilde{w}^k)], \eta, \tilde{\eta}),$$

respectively.

Next, $\tilde{z}^{k+1} = z_{R_1}^{k+1} = \dots = z_{R_n}^{k+1}$ follows from (G.10), (G.11) and (G.12) together with the update rule (on $\{z^k\}$ and $\{\tilde{z}^k\}$) of both algorithms and the fact that $\frac{\gamma}{\eta} = \frac{\tilde{\gamma}}{\tilde{\eta}}$.

To finish the proof of the algorithms equivalence, we shall notice that $\tilde{w}^{k+1} = w_{R_1}^{k+1} = \dots = w_{R_n}^{k+1}$ follows from (G.10), (G.12) together with the update rule (on $\{w^k\}$ and $\{\tilde{w}^k\}$) of both algorithms.

To show $\mathcal{L}' = \frac{\tilde{\mathcal{L}}}{n}$ it is sufficient to see

$$\begin{aligned}
& \mathbb{E} \left[\left\| g^k - \nabla f(x^k) \right\|_{\mathbf{W}}^2 \right] \\
&= \mathbb{E} \left[\left\| \sum_{i \in \tilde{\mathcal{S}}} p_i^{-1} \left(\sum_{j \in R_i} \left(\nabla_j f(x^k) - \nabla_j f(w^k) \right) e_j \right) - \left(\nabla f(x^k) - \nabla f(w^k) \right) \right\|_{\mathbf{W}}^2 \right] \\
&= \mathbb{E} \left[\left\| \mathbf{W} \left(\sum_{i \in \tilde{\mathcal{S}}} p_i^{-1} \left(\sum_{j \in R_i} \left(\nabla_j f(x^k) - \nabla_j f(w^k) \right) e_j \right) \right) - \mathbf{W} \left(\nabla f(x^k) - \nabla f(w^k) \right) \right\|^2 \right] \\
&= \frac{1}{n} \mathbb{E} \left[\left\| \left(\frac{1}{n} \sum_{i \in \tilde{\mathcal{S}}} \tilde{p}_i^{-1} \left(\nabla \tilde{f}_i(\tilde{x}^k) - \nabla \tilde{f}_i(\tilde{w}^k) \right) \right) - \left(\nabla \tilde{f}(\tilde{x}^k) - \nabla \tilde{f}(\tilde{w}^k) \right) \right\|^2 \right] \\
&= \frac{1}{n} \mathbb{E} \left[\left\| \tilde{g}^k - \nabla \tilde{f}(\tilde{w}^k) \right\|^2 \right] \\
&\leq 2 \frac{\tilde{\mathcal{L}}}{n} D_{\tilde{f}}(\tilde{w}^k, \tilde{x}^k) \\
&= 2 \frac{\tilde{\mathcal{L}}}{n} \left(\frac{1}{n} \sum_{i=1}^n D_{\tilde{f}_i}(w_{R_i}^k, x_{R_i}^k) \right) \\
&= 2 \frac{\tilde{\mathcal{L}}}{n} D_f(w^k, x^k).
\end{aligned}$$

Lastly, if $x, y \in \text{Range}(\mathbf{W})$, there is $\tilde{x}, \tilde{y} \in \mathbb{R}^{\tilde{d}}$ such that $x = U(\tilde{x}), y = U(\tilde{y})$. Therefore we can write

$$\begin{aligned}
f(x) &= f(\mathbf{W}(x)) = \frac{1}{n} \sum_{j=1}^n \tilde{f}_j(\tilde{x}) \\
&\leq \frac{1}{n} \sum_{j=1}^n \tilde{f}_j(\tilde{y}) + \left\langle \nabla \left(\frac{1}{n} \sum_{j=1}^n \tilde{f}_j(\tilde{y}) \right), \tilde{x} - \tilde{y} \right\rangle + \frac{\mathcal{L}}{2} \|\tilde{x} - \tilde{y}\|^2 \\
&= f(y) + \langle \nabla f(y), x - y \rangle + \frac{\mathcal{L}}{2n} \|x - y\|^2,
\end{aligned}$$

and thus $L = \lambda_{\max} \left(\mathbf{M}^{\frac{1}{2}} \mathbf{W} \mathbf{M}^{\frac{1}{2}} \right) \leq n^{-1} \mathcal{L}$.

G.4 Tighter rates for GJS by exploiting prox and proof of Theorem 7.3.2

In this section, we show that specific nonsmooth function ψ might lead to faster convergence of variance reduced methods. We exploit the well-known fact that under some circumstances, a proximal operator might change the smoothness structure of the objective [73]. In particular, we consider GJS from Chapter 5. We generalize Theorem 5.4.2 therein, which allows for a tighter rate if ψ has a specific structure.

Theorem G.4.1 (Extension of Theorem 5.4.2 from Chapter 5). Define $f(x) \stackrel{\text{def}}{=} \frac{1}{n} \sum_{i=1}^n f_i(x)$. Let Assumption 7.2.1 hold and suppose that $\mathcal{M}^{\dagger \frac{1}{2}}$ commutes with \mathcal{S} . Next, let α and \mathcal{B} are such that for every $\mathbf{X} \in \mathbb{R}^{d \times n}$ we have

$$\frac{2\alpha}{n^2} \mathbb{E} [\|\mathcal{U} \mathbf{X} e\|_{\mathbf{W}}^2] + \left\| (\mathcal{I} - \mathbb{E}[\mathcal{S}])^{\frac{1}{2}} \mathcal{B} \mathcal{M}^{\dagger} \mathbf{X} \right\|^2 \leq (1 - \alpha \sigma') \|\mathcal{B} \mathcal{M}^{\dagger} \mathbf{X}\|^2, \quad (\text{G.13})$$

$$\frac{2\alpha}{n^2} \mathbb{E} [\|\mathcal{U} \mathbf{X} e\|_{\mathbf{W}}^2] + \left\| (\mathbb{E}[\mathcal{S}])^{\frac{1}{2}} \mathcal{B} \mathcal{M}^{\dagger} \mathbf{X} \right\|^2 \leq \frac{1}{n} \|\mathcal{M}^{\dagger} \mathbf{X}\|^2 \quad (\text{G.14})$$

and \mathcal{B} commutes with \mathcal{S} . Then for all $k \geq 0$, we have

$$\mathbb{E} [\Psi^k] \leq (1 - \alpha \sigma')^k \Psi^0,$$

where

$$\Psi^k \stackrel{\text{def}}{=} \|x^k - x^*\|^2 + \alpha \left\| \mathcal{B} \mathcal{M}^{\dagger \frac{1}{2}} (\mathbf{J}^k - \mathbf{G}(x^*)) \right\|^2.$$

G.4.1 Towards the proof of Theorem G.4.1

Lemma G.4.2. (Slight extension of Lemma E.2.6) Let \mathcal{U} be random linear operator which is identity in expectation. Let $\mathbf{G}(x)$ be Jacobian at x and $g^k = \frac{1}{n} \mathcal{U}(\mathbf{G}(x) - \mathbf{J}^k) e - \frac{1}{n} \mathbf{J}^k e$. Then for any $\mathbf{Q} \in \mathbb{R}^{d \times d}$, $\mathbf{Q} \succeq 0$ and all $k \geq 0$ we have

$$\mathbb{E} [\|g^k - \nabla f(x^*)\|_{\mathbf{Q}}^2] \leq \frac{2}{n^2} \mathbb{E} [\|\mathcal{U} (\mathbf{G}(x^k) - \mathbf{G}(x^*)) e\|_{\mathbf{Q}}^2] + \frac{2}{n^2} \mathbb{E} [\|\mathcal{U} (\mathbf{J}^k - \mathbf{G}(x^*)) e\|_{\mathbf{Q}}^2]. \quad (\text{G.15})$$

Proof. Since $\nabla f(x^*) = \frac{1}{n} \mathbf{G}(x^*) e$, we have

$$g^k - \nabla f(x^*) = \underbrace{\frac{1}{n} \mathcal{U} (\mathbf{G}(x^k) - \mathbf{G}(x^*)) e}_a + \underbrace{\frac{1}{n} (\mathbf{J}^k - \mathbf{G}(x^*)) e - \frac{1}{n} \mathcal{U} (\mathbf{J}^k - \mathbf{G}(x^*)) e}_b. \quad (\text{G.16})$$

Applying the bound $\|a + b\|_{\mathbf{Q}}^2 \leq 2 \|a\|_{\mathbf{Q}}^2 + 2 \|b\|_{\mathbf{Q}}^2$ to (G.16) and taking expectations, we get

$$\begin{aligned} \mathbb{E} [\|g^k - \nabla f(x^*)\|_{\mathbf{Q}}^2] &\leq \mathbb{E} \left[\frac{2}{n^2} \|\mathcal{U} (\mathbf{G}(x^k) - \mathbf{G}(x^*)) e\|_{\mathbf{Q}}^2 \right] \\ &\quad + \mathbb{E} \left[\frac{2}{n^2} \|(\mathbf{J}^k - \mathbf{G}(x^*)) e - \mathcal{U} (\mathbf{J}^k - \mathbf{G}(x^*)) e\|_{\mathbf{Q}}^2 \right] \\ &= \frac{2}{n^2} \mathbb{E} [\|\mathcal{U} (\mathbf{G}(x^k) - \mathbf{G}(x^*)) e\|_{\mathbf{Q}}^2] \\ &\quad + \frac{2}{n^2} \mathbb{E} [\|(\mathcal{I} - \mathcal{U}) (\mathbf{J}^k - \mathbf{G}(x^*)) e\|_{\mathbf{Q}}^2]. \end{aligned}$$

It remains to note that

$$\begin{aligned}\mathbb{E} \left[\left\| (\mathcal{I} - \mathcal{U})(\mathbf{J}^k - \mathbf{G}(x^*)) \mathbf{e} \right\|_{\mathbf{Q}}^2 \right] &= \mathbb{E} \left[\left\| \mathcal{U}(\mathbf{J}^k - \mathbf{G}(x^*)) \mathbf{e} \right\|_{\mathbf{Q}}^2 \right] - \left\| (\mathbf{J}^k - \mathbf{G}(x^*)) \mathbf{e} \right\|_{\mathbf{Q}}^2 \\ &\leq \mathbb{E} \left[\left\| \mathcal{U}(\mathbf{J}^k - \mathbf{G}(x^*)) \mathbf{e} \right\|_{\mathbf{Q}}^2 \right].\end{aligned}$$

□

Next, we restate two lemmas from the appendix of Chapter 5 which we need to show the convergence.

Lemma G.4.3. (Chapter E, Lemma E.2.3) Assume that function f_j are convex and \mathbf{M}_j -smooth. Then

$$D_{f_j}(x, y) \geq \frac{1}{2} \left\| \nabla f_j(x) - \nabla f_j(y) \right\|_{\mathbf{M}_j^\dagger}^2, \quad \forall x, y \in \mathbb{R}^d, 1 \leq j \leq n. \quad (\text{G.17})$$

If $x - y \in \text{Null}(\mathbf{M}_j)$, then

$$(i) \quad f_j(x) = f_j(y) + \langle \nabla f_j(y), x - y \rangle, \quad (\text{G.18})$$

$$(ii) \quad \nabla f_j(x) - \nabla f_j(y) \in \text{Null}(\mathbf{M}_j), \quad (\text{G.19})$$

$$(iii) \quad \langle \nabla f_j(x) - \nabla f_j(y), x - y \rangle = 0. \quad (\text{G.20})$$

If, in addition, f_j is bounded below, then $\nabla f_j(x) \in \text{Range}(\mathbf{M}_j)$ for all x .

Lemma G.4.4. (Chapter E, Lemma E.2.5) Let \mathcal{S} be a random projection operator and \mathcal{A} any deterministic linear operator commuting with \mathcal{S} , i.e., $\mathcal{A}\mathcal{S} = \mathcal{S}\mathcal{A}$. Further, let $\mathbf{X}, \mathbf{Y} \in \mathbb{R}^{d \times n}$ and define $\mathbf{Z} = (\mathcal{I} - \mathcal{S})\mathbf{X} + \mathcal{S}\mathbf{Y}$. Then

$$(i) \quad \mathcal{A}\mathbf{Z} = (\mathcal{I} - \mathcal{S})\mathcal{A}\mathbf{X} + \mathcal{S}\mathcal{A}\mathbf{Y},$$

$$(ii) \quad \|\mathcal{A}\mathbf{Z}\|^2 = \|(\mathcal{I} - \mathcal{S})\mathcal{A}\mathbf{X}\|^2 + \|\mathcal{S}\mathcal{A}\mathbf{Y}\|^2,$$

$$(iii) \quad \mathbb{E} \left[\|\mathcal{A}\mathbf{Z}\|^2 \right] = \|(\mathcal{I} - \mathbb{E}[\mathcal{S}])^{1/2} \mathcal{A}\mathbf{X}\|^2 + \left\| \mathbb{E}[\mathcal{S}]^{1/2} \mathcal{A}\mathbf{Y} \right\|^2, \text{ where the expectation is with respect to } \mathcal{S}.$$

Proof of Theorem G.4.1 For simplicity of notation, in this proof, all expectations are conditional on x^k , i.e., the expectation is taken with respect to the randomness of g^k . First notice that

$$\mathbb{E} [g^k] = \nabla f(x^k). \quad (\text{G.21})$$

For any differentiable function h let $D_h(x, y)$ to be Bregman distance with kernel h , i.e., $D_h(x, y) \stackrel{\text{def}}{=} h(x) - h(y) - \langle \nabla h(y), x - y \rangle$. Since

$$x^* = \text{prox}_{\alpha\psi}(x^* - \alpha \nabla f(x^*)), \quad (\text{G.22})$$

and since the prox operator is non-expansive, we have

$$\begin{aligned}
\mathbb{E} \left[\|x^{k+1} - x^*\|^2 \right] &\stackrel{(G.22)}{=} \mathbb{E} \left[\left\| \text{prox}_{\alpha\psi}(x^k - \alpha g^k) - \text{prox}_{\alpha\psi}(x^* - \alpha \nabla f(x^*)) \right\|^2 \right] \\
&\stackrel{(7.5)+(7.4)}{\leq} \mathbb{E} \left[\left\| x^k - x^* - \alpha \mathbf{W}(g^k - \nabla f(x^*)) \right\|^2 \right] \\
&\stackrel{(G.21)}{=} \left\| x^k - x^* \right\|^2 - 2\alpha \langle \nabla f(x^k) - \nabla f(x^*), x^k - x^* \rangle \\
&\quad + \alpha^2 \mathbb{E} \left[\left\| g^k - \nabla f(x^*) \right\|_{\mathbf{W}}^2 \right] \\
&\leq (1 - \alpha\sigma') \left\| x^k - x^* \right\|^2 + \alpha^2 \mathbb{E} \left[\left\| g^k - \nabla f(x^*) \right\|_{\mathbf{W}}^2 \right] \\
&\quad - 2\alpha D_f(x^k, x^*). \tag{G.23}
\end{aligned}$$

Since $f(x) = \frac{1}{n} \sum_{i=1}^n f_i(x)$, in view of (G.17) we have

$$\begin{aligned}
D_f(x^k, x^*) &= \frac{1}{n} \sum_{i=1}^n D_{f_i}(x^k, x^*) \stackrel{(G.17)}{\geq} \frac{1}{2n} \sum_{i=1}^n \left\| \nabla f_i(x^k) - \nabla f_i(x^*) \right\|_{\mathbf{M}_i^\dagger}^2 \\
&= \frac{1}{2n} \left\| \mathcal{M}^{\dagger \frac{1}{2}} (\mathbf{G}(x^k) - \mathbf{G}(x^*)) \right\|^2. \tag{G.24}
\end{aligned}$$

By combining (G.23) and (G.24), we get

$$\begin{aligned}
\mathbb{E} \left[\|x^{k+1} - x^*\|^2 \right] &\leq (1 - \alpha\sigma') \left\| x^k - x^* \right\|^2 + \alpha^2 \mathbb{E} \left[\left\| g^k - \nabla f(x^*) \right\|_{\mathbf{W}}^2 \right] \\
&\quad - \frac{\alpha}{n} \left\| \mathcal{M}^{\dagger \frac{1}{2}} (\mathbf{G}(x^k) - \mathbf{G}(x^*)) \right\|^2.
\end{aligned}$$

Next, applying Lemma G.4.2 with $\mathbf{Q} = \mathbf{W}$ leads to the estimate

$$\begin{aligned}
\mathbb{E} \left[\|x^{k+1} - x^*\|^2 \right] &\leq (1 - \alpha\sigma') \left\| x^k - x^* \right\|^2 - \frac{\alpha}{n} \left\| \mathcal{M}^{\dagger \frac{1}{2}} (\mathbf{G}(x^k) - \mathbf{G}(x^*)) \right\|^2 \\
&\quad + \frac{2\alpha^2}{n^2} \mathbb{E} \left[\left\| \mathcal{U} (\mathbf{G}(x^k) - \mathbf{G}(x^*)) e \right\|_{\mathbf{W}}^2 \right] \\
&\quad + \frac{2\alpha^2}{n^2} \mathbb{E} \left[\left\| \mathcal{U} (\mathbf{J}^k - \mathbf{G}(x^*)) e \right\|_{\mathbf{W}}^2 \right]. \tag{G.25}
\end{aligned}$$

Since, by assumption, both \mathcal{B} and $\mathcal{M}^{\dagger \frac{1}{2}}$ commute with \mathcal{S} , so does their composition $\mathcal{A} \stackrel{\text{def}}{=} \mathcal{B}\mathcal{M}^{\dagger \frac{1}{2}}$. Applying Lemma G.4.4, we get

$$\begin{aligned}
\mathbb{E} \left[\left\| \mathcal{B}\mathcal{M}^{\dagger \frac{1}{2}} (\mathbf{J}^{k+1} - \mathbf{G}(x^*)) \right\|^2 \right] &= \left\| (\mathcal{I} - \mathbb{E}[\mathcal{S}])^{\frac{1}{2}} \mathcal{B}\mathcal{M}^{\dagger \frac{1}{2}} (\mathbf{J}^k - \mathbf{G}(x^*)) \right\|^2 \\
&\quad + \left\| \mathbb{E}[\mathcal{S}]^{\frac{1}{2}} \mathcal{B}\mathcal{M}^{\dagger \frac{1}{2}} (\mathbf{G}(x^k) - \mathbf{G}(x^*)) \right\|^2. \tag{G.26}
\end{aligned}$$

Adding α -multiple of (G.26) for $\mathcal{C} = \mathcal{M}^{\dagger \frac{1}{2}}$ to (G.25) yields

$$\begin{aligned}
& \mathbb{E} \left[\|x^{k+1} - x^*\|^2 \right] + \alpha \mathbb{E} \left[\left\| \mathcal{B} \left(\mathcal{M}^{\dagger \frac{1}{2}} (\mathbf{J}^{k+1} - \mathbf{G}(x^*)) \right) \right\|_{\mathbf{W}}^2 \right] \\
& \leq (1 - \alpha\sigma') \|x^k - x^*\|^2 + \frac{2\alpha^2}{n^2} \mathbb{E} \left[\|\mathcal{U}(\mathbf{G}(x^k) - \mathbf{G}(x^*))e\|_{\mathbf{W}}^2 \right] + \\
& \quad \frac{2\alpha^2}{n^2} \mathbb{E} \left[\|\mathcal{U}(\mathbf{J}^k - \mathbf{G}(x^*))e\|_{\mathbf{W}}^2 \right] + \alpha \left\| (\mathcal{I} - \mathbb{E}[\mathcal{S}])^{\frac{1}{2}} \left(\mathcal{B} \left(\mathcal{M}^{\dagger \frac{1}{2}} (\mathbf{J}^k - \mathbf{G}(x^*)) \right) \right) \right\|^2 \\
& \quad + \alpha \left\| \mathbb{E}[\mathcal{S}]^{\frac{1}{2}} \left(\mathcal{B} \left(\mathcal{M}^{\dagger \frac{1}{2}} (\mathbf{G}(x^k) - \mathbf{G}(x^*)) \right) \right) \right\|^2 - \frac{\alpha}{n} \left\| \mathcal{M}^{\dagger \frac{1}{2}} (\mathbf{G}(x^k) - \mathbf{G}(x^*)) \right\|^2 \\
& \stackrel{(G.13)}{\leq} (1 - \alpha\sigma') \|x^k - x^*\|^2 + (1 - \alpha\sigma') \alpha \mathbb{E} \left[\left\| \mathcal{B} \left(\mathcal{M}^{\dagger \frac{1}{2}} (\mathbf{J}^k - \mathbf{G}(x^*)) \right) \right\|_{\mathbf{W}}^2 \right] \\
& \quad + \frac{2\alpha^2}{n^2} \mathbb{E} \left[\|\mathcal{U}(\mathbf{G}(x^k) - \mathbf{G}(x^*))e\|_{\mathbf{W}}^2 \right] + \alpha \left\| \mathbb{E}[\mathcal{S}]^{\frac{1}{2}} \left(\mathcal{B} \left(\mathcal{M}^{\dagger \frac{1}{2}} (\mathbf{G}(x^k) - \mathbf{G}(x^*)) \right) \right) \right\|^2 \\
& \quad - \frac{\alpha}{n} \left\| \mathcal{M}^{\dagger \frac{1}{2}} (\mathbf{G}(x^k) - \mathbf{G}(x^*)) \right\|^2 \\
& \stackrel{(G.14)}{\leq} (1 - \alpha\sigma') \left(\|x^k - x^*\|^2 + \alpha \mathbb{E} \left[\left\| \mathcal{B} \left(\mathcal{M}^{\dagger \frac{1}{2}} (\mathbf{J}^k - \mathbf{G}(x^*)) \right) \right\|_{\mathbf{W}}^2 \right] \right).
\end{aligned}$$

Above, we have used (G.13) with $\mathbf{X} = \mathbf{J}^k - \mathbf{G}(x^*)$ and (G.14) with $\mathbf{X} = \mathbf{G}(x^k) - \mathbf{G}(x^*)$.

G.4.2 Proof of Theorem 7.3.2

First, due to our choice of \mathcal{S} we have $\mathbb{E}[\mathcal{S}(x)] = \mathbf{Diag}(p)x$ and at the same time \mathcal{S} and $\mathcal{M}^{\dagger \frac{1}{2}}$ commute. Next, (7.6) implies

$$\mathbb{E} \left[\left\| \mathcal{U}(\mathbf{M}^{\frac{1}{2}}x) \right\|_{\mathbf{W}}^2 \right] = \|x\|_{\mathbf{M}^{\frac{1}{2}} \mathbb{E}[\sum_{i \in S} \mathbf{p}_i^{-1} \mathbf{e}_i \mathbf{e}_i^\top \mathbf{W} \sum_{i \in S} \mathbf{p}_i^{-1} \mathbf{e}_i \mathbf{e}_i^\top] \mathbf{M}^{\frac{1}{2}}}^2 \leq \|x\|_{p^{-1}ow}^2.$$

In order to satisfy (G.13) and (G.14) it remains to have (we substituted $y = \mathbf{M}^{\dagger \frac{1}{2}}x$):

$$2\alpha \|y\|_{p^{-1}ow}^2 + \left\| (\mathcal{I} - \mathbb{E}[\mathcal{S}])^{\frac{1}{2}} \mathcal{B}(y) \right\|^2 \leq (1 - \alpha\sigma) \|\mathcal{B}(y)\|^2, \quad (G.27)$$

$$2\alpha \|y\|_{p^{-1}ow}^2 + \left\| (\mathbb{E}[\mathcal{S}])^{\frac{1}{2}} \mathcal{B}(y) \right\|^2 \leq \|y\|^2. \quad (G.28)$$

Let us consider \mathcal{B} to be the operator corresponding to the left multiplication with matrix $\mathbf{Diag}(b)$. Thus for satisfy (G.13) it suffices to have for all $i \in [d]$:

$$2\alpha m_i \mathbf{p}_i^{-1} + b_i^2(1 - \mathbf{p}_i) \leq b_i^2(1 - \alpha\sigma) \quad \Rightarrow \quad 2\alpha m_i \mathbf{p}_i^{-1} + b_i^2\alpha\sigma \leq b_i^2 \mathbf{p}_i.$$

For (G.14) it suffices to have for all $i \in [d]$

$$2\alpha m_i \mathbf{p}_i^{-1} + b_i^2 \mathbf{p}_i \leq 1.$$

It remains to notice that choice $b_i^2 = \frac{1}{2\mathbf{p}_i}$ and $\alpha = \min_i \frac{\mathbf{p}_i}{4m_i + \sigma}$ is valid.

Appendix H

Appendix for Chapter 8

H.1 Remaining algorithms

H.1.1 Local GD with variance reduction

In this section, we present variance reduced local gradient descent with partial aggregation. In particular, the proposed algorithm (Algorithm 60) incorporates control variates to Algorithm 19. Therefore, the proposed method can be seen as a special case of Algorithm 20 with $m = 1$. We thus present it for pedagogical purposes only, as it might shed additional insights into our approach. In particular, the update rule of proposed method will be

$$x^{k+1} = x^k - \alpha g^k,$$

where

$$g^k \stackrel{\text{def}}{=} \begin{cases} p^{-1}(\lambda \nabla \Phi(x^k) - n^{-1} \Psi^k) + n^{-1} \mathbf{J}^k + n^{-1} \Psi^k & \text{with probability } p \\ (1-p)^{-1}(\nabla f(x^k) - n^{-1} \mathbf{J}^k) + n^{-1} \mathbf{J}^k + n^{-1} \Psi^k & \text{with probability } 1-p \end{cases}$$

for some control variates vectors $\mathbf{J}^k, \Psi^k \in \mathbb{R}^{nd}$. A quick check gives

$$\mathbb{E}[g^k | x^k] = \nabla f(x^k) + \lambda \nabla \Phi(x^k) = \nabla F(x^k),$$

thus the direction we are taking is unbiased regardless of the value of control variates \mathbf{J}^k, Ψ^k . The goal is to make control variates \mathbf{J}^k, Ψ^k correlated¹ with $n \nabla f(x^k)$ and $n \lambda \nabla \Phi(x^k)$. One possible solution to the problem is for \mathbf{J}^k, Ψ^k to track most recently observed values of $n \nabla f(\cdot)$ and $n \lambda \nabla \Phi(\cdot)$, which corresponds to the following update rule

$$(\Psi^{k+1}, \mathbf{J}^{k+1}) = \begin{cases} (n \lambda \nabla \Phi(x^k), \mathbf{J}^k) & \text{with probability } p \\ (\Psi^k, n \nabla f(x^k)) & \text{with probability } 1-p. \end{cases}$$

A specific, distributed implementation of the described method is presented as Algorithm 60. The only communication between the devices takes place when the average model \bar{x}^k is being computed (with probability p), which is analogous to standard local SGD. Therefore we aim to set p rather small.

Note that Algorithm 60 is a particular special case of SAGA with importance sampling [165]; thus, we obtain convergence rate of the method for free. We state it as Theorem H.1.1.

¹Specifically we aim to have $\text{Corr}[\mathbf{J}^k, n \nabla f(x^k)] \rightarrow 1$ and $\text{Corr}[n^{-1} \Psi^k, \lambda \nabla \Phi(x^k)] \rightarrow 1$ as $x^k \rightarrow x^*$.

Algorithm 60 Variance reduced local gradient descent

Input: $x_1^0 = \dots = x_n^0 \in \mathbb{R}^d$, stepsize α , probability p
 $\mathbf{J}_1^0 = \dots = \mathbf{J}_n^0 = \mathbf{\Psi}_1^0 = \dots = \mathbf{\Psi}_n^0 = \mathbf{0} \in \mathbb{R}^d$
for $k = 0, 1, 2, \dots$ **do**
 $\xi = 1$ with probability p and 0 with probability $1 - p$
 if ξ **then**
 All **Devices** $i = 1, \dots, n$:
 Compute $\nabla f_n(x_n^k)$
 $x_n^{k+1} = x_n^k - \alpha \left(n^{-1}(1-p)^{-1} \nabla f_n(x_n^k) - n^{-1} \frac{p}{1-p} \mathbf{J}_n^k + n^{-1} \mathbf{\Psi}_n^k \right)$
 Set $\mathbf{J}_n^{k+1} = \nabla f_n(x_n^k)$, $\mathbf{\Psi}_n^{k+1} = \mathbf{\Psi}_n^k$
 else
 Master computes the average $\bar{x}^k = \frac{1}{n} \sum_{i=1}^n x_i^k$
 Master does for all $i = 1, \dots, n$:
 Set $x_n^{k+1} = x_n^k - \alpha \left(\frac{\lambda}{np} (x_n^k - \bar{x}^k) - (p^{-1} - 1) n^{-1} \mathbf{\Psi}_n^k + n^{-1} \mathbf{J}_n^k \right)$
 Set $\mathbf{\Psi}_n^{k+1} = \lambda(x_n^k - \bar{x}^k)$, $\mathbf{J}_n^{k+1} = \mathbf{J}_n^k$
 end if
end for

Theorem H.1.1. *Let Assumption 8.3.1 hold. Set $\alpha = n \min \left\{ P \frac{(1-p)}{4L+\mu}, \frac{p}{4\lambda+\mu} \right\}$. Then, iteration complexity of Algorithm 60 is*

$$\max \left\{ \frac{4L + \mu}{\mu(1-p)}, \frac{4\lambda + \mu}{\mu p} \right\} \log \frac{1}{\varepsilon}.$$

Proof. Clearly,

$$F(x) = f(x) + \lambda \Phi(x) = \frac{1}{2} \left(\underbrace{2f(x)}_{\stackrel{\text{def}}{=} \textcolor{green}{f}(x)} + \underbrace{2\lambda \Phi(x)}_{\stackrel{\text{def}}{=} \textcolor{blue}{\psi}(x)} \right).$$

Note that $\textcolor{blue}{\psi}$ is $\frac{2\lambda}{n}$ -smooth and $\textcolor{green}{f}$ is $\frac{2L}{n}$ -smooth. At the same time, F is $\frac{\mu}{n}$ -strongly convex. Using convergence theorem of SAGA with importance sampling from [165, 52], we get

$$\mathbb{E} \left[F(x^k) + \frac{\alpha}{2} \Upsilon(\mathbf{J}^k, \mathbf{\Psi}^k) \right] \leq \left(1 - \alpha \frac{\mu}{n} \right)^k \left(F(x^0) + \frac{\alpha}{2} \Upsilon(\mathbf{J}^0, \mathbf{\Psi}^0) \right),$$

where

$$\Upsilon(\mathbf{J}^k, \mathbf{\Psi}^k) \stackrel{\text{def}}{=} \frac{4}{n^2} \sum_{n=1}^n (\|\mathbf{\Psi}_n^k - \lambda(x_n^* - \bar{x}^*)\|^2 + \|\mathbf{J}_n^k - \nabla f_n(x_n^*)\|^2)$$

and

$$\alpha = n \min \left\{ \frac{(1-p)}{4L + \mu}, \frac{p}{4\lambda + \mu} \right\},$$

as desired. □

Corollary H.1.2. *Iteration complexity of Algorithm 60 is minimized for $p = \frac{4\lambda+\mu}{4\lambda+4L+2\mu}$, which yields complexity $4\left(\frac{\lambda}{\mu} + \frac{L}{\mu} + \frac{1}{2}\right) \log \frac{1}{\varepsilon}$. The communication complexity is minimized for any $p \leq \frac{4\lambda+\mu}{4\lambda+4L+2\mu}$, in which case the total number of communication rounds to reach ε -solution is $\left(\frac{4\lambda}{\mu} + 1\right) \log \frac{1}{\varepsilon}$.*

As a direct consequence of Corollary H.1.2 we see that the optimal choice of p that minimizes both communication and number of iterations to reach ε solution of problem (8.2) is $p = \frac{4\lambda+\mu}{4\lambda+4L+2\mu}$.

Remark 34. While both Algorithm 60 and Algorithm 20 are a special case of SAGA, the practical version of variance reduced local SGD (presented in Section H.1.3) is not. In particular, we wish to run the SVRG-like method locally in order to avoid storing the full gradient table.² Therefore, variance reduced local SGD that will be proposed in Section H.1.3 is neither a special case of SAGA nor a special case of SVRG (or a variant of SVRG). However, it is still a special case of a GJS from Chapter 5.

As mentioned, Algorithm 20 is a generalization of Algorithm 60 when the local subproblem is a finite sum. Note that Algorithm 60 constructs a control variates for both local subproblem and aggregation function Φ and constructs corresponding unbiased gradient estimator. In contrast, Algorithm 20 constructs extra control variates within the local subproblem in order to reduce the variance of gradient estimator coming from the local subsampling.

H.1.2 Efficient implementation of L2SGD+

Here we present an efficient implementation of L2SGD+ as Algorithm 61 so that we do not have to communicate control variates. As a consequence, Algorithm 61 needs to communicate on average $p(1-p)k$ times per k iterations, while each communication consists of sending only local models to the master and back.

H.1.3 Local SGD with variance reduction – general method

In this section, we present a fully general variance reduced local SGD. We consider a more general instance of (8.2) where each local objective includes a possibly nonsmooth regularizer, which admits a cheap evaluation of proximal operator. In particular, the objective becomes

$$\min_{x \in \mathbb{R}^{dn}} F(x) \stackrel{\text{def}}{=} \underbrace{\frac{1}{N} \sum_{i=1}^n \left(\sum_{j=1}^{m_i} \textcolor{red}{f}_{i,j}(x_i) \right)}_{= \frac{N}{n} f_n(x)} + \underbrace{\lambda \frac{1}{2n} \sum_{i=1}^n \|x_i - \bar{x}\|^2}_{=\Phi(x)} + \underbrace{\sum_{i=1}^n \psi_i(x_i)}_{\stackrel{\text{def}}{=} \psi(x)}, \quad (\text{H.1})$$

²SAGA does not require storing a full gradient table for problems with linear models by memorizing the residuals. However, in full generality, SVRG-like methods are preferable.

Algorithm 61 L2SGD+: Loopless Local SGD with Variance Reduction (communication-efficient implementation)

Input: $x_1^0 = \dots = x_n^0 = \tilde{x} \in \mathbb{R}^d$, stepsize α , probability p
Initialize control variates $\mathbf{J}_n^0 = \mathbf{0} \in \mathbb{R}^{d \times m}$, $\Psi_n^0 = \mathbf{0} \in \mathbb{R}^d$ (for $n = 1, \dots, n$), initial coin toss $\xi^{-1} = 0$
for $k = 0, 1, 2, \dots$ **do**
 $\xi^k = 1$ with probability p and 0 with probability $1 - p$
 if $\xi^k = 0$ **then**
 All **Devices** $i = 1, \dots, n$:
 if $\xi^{k-1} = 1$ **then**
 Receive x_i^k, c from **Master**
 Reconstruct $\bar{x}^k = \bar{x}^{k-c}$ using x_i^k, x_i^{k-c}, c
 Set $x^k = x^k - c\alpha \frac{1}{nm} \mathbf{J}_n^k \mathbf{1}$, $\mathbf{J}_n^k = \mathbf{J}_n^{k-c}$, $\Psi_n^k = \lambda(x_i^{k-c} - \bar{x}^k)$,
 end if
 Sample $j \in \{1, \dots, m\}$ (uniformly at random)
 $g_n^k = \frac{1}{n(1-p)} \left(\nabla f_{i,j}(x_n^k) - (\mathbf{J}_n^k)_{:,j} \right) + \frac{\mathbf{J}_n^k \mathbf{1}}{nm} + \frac{\Psi_n^k}{n}$
 $x_n^{k+1} = x_n^k - \alpha g_n^k$
 Set $(\mathbf{J}_i^{k+1})_{:,j} = \nabla f_{i,j}(x_n^k)$, $\Psi_n^{k+1} = \Psi_n^k$,
 $(\mathbf{J}_i^{k+1})_{:,l} = (\mathbf{J}_i^{k+1})_{:,l}$ for all $l \neq j$
 else
 Master does for all $i = 1, \dots, n$:
 if $\xi^{k-1} = 0$ **then**
 Set $c = 0$
 Receive x_i^k from **Device** and set $\bar{x} = \frac{1}{n} \sum_{i=1}^n x_i^k$, $x_i^k = x_i^k$
 end if
 Set $x_n^{k+1} = x_n^k - \alpha \left(\frac{\lambda}{np} (x_n^k - \bar{x}) - \frac{p^{-1}-1}{n} \lambda (\tilde{x} - \bar{x}) \right)$
 Set $\tilde{x} = x_i^k$
 Set $c = c + 1$
 end if
 end for

where m_i is the number of data points owned by client i and $N = \sum_{i=1}^n m_i$.

In order to squeeze a faster convergence rate from minibatch samplings, we will assume that $f_{i,j}$ is smooth with respect to a matrix $\mathbf{M}_{i,j}$ (instead of scalar $\tilde{L}_{i,j} = \lambda_{\max} \mathbf{M}_{i,j}$).

Assumption H.1.3. Suppose that $f_{i,j}$ is $\mathbf{M}_{i,j}$ smooth ($\mathbf{M}_{i,j} \in \mathbb{R}^{d \times d}$, $\mathbf{M}_{i,j} \succ 0$) and convex for $1 \leq j \leq m_i$, $1 \leq i \leq n$, i.e., for all $x, y \in \mathbb{R}^d$ we have

$$f_{i,j}(y) + \langle \nabla f_{i,j}(y), x - y \rangle \leq f_{i,j}(x) \leq f_{i,j}(y) + \langle \nabla f_{i,j}(y), x - y \rangle + \frac{1}{2} \|y - x\|_{\mathbf{M}_{i,j}}^2. \quad (\text{H.2})$$

Furthermore, assume that ψ_i is convex for $1 \leq i \leq n$.

Our method (Algorithm 62) allows for arbitrary aggregation probability (same as Algorithms 60, 20), arbitrary sampling of clients (to model the inactive clients) and arbitrary

structure/sampling of the local objectives (i.e., arbitrary size of local datasets, arbitrary smoothness structure of each local objective and arbitrary subsampling strategy of each client). Moreover, it allows for the SVRG-like update rule of local control variates \mathbf{J}^k , which requires less storage given an efficient implementation.

To be specific, each device owns a distribution \mathcal{D}_i over subsets of m_i . When the aggregation is not performed (with probability $1 - p$), a subset of active devices \mathcal{S} is selected (\mathcal{S} follows arbitrary fixed distribution \mathcal{D}). Each of the active clients ($i \in \mathcal{S}$) samples a subset of local indices $S_i \sim \mathcal{D}_i$ and observe the corresponding part of local Jacobian $\mathbf{G}_i(x^k)_{(:,S_i)}$ (where $\mathbf{G}_i(x^k) \stackrel{\text{def}}{=} [\nabla f_{i,1}(x^k), \nabla f_{i,2}(x^k), \dots, \nabla f_{i,m_i}(x^k)]$). When the aggregation is performed (with probability p) we evaluate \bar{x}^k and distribute it to each device; using which each device computes a corresponding component of $\lambda \nabla \Phi(x^k)$. Those are the key components in constructing the unbiased gradient estimator (without control variates).

It remains to construct control variates and unbiased gradient estimator. If the aggregation is done, we just simply replace the last column of the gradient table. If the aggregation is not done, we have two options – either keep replacing the columns of the Jacobian table (in such case, we obtain a particular case of SAGA [37]) or do LSVRG-like replacement [83, 106] (in such case, the algorithm is a particular case of GJS from Chapter 5, but is not a special case of neither SAGA nor LSVRG. Note that LSVRG-like replacement is preferable in practice due to a better memory efficiency (one does not need to store the whole gradient table) for the models other than linear).

In order to keep the gradient estimate unbiased, it will be convenient to define vector $\mathbf{p}_i \in \mathbb{R}^{m_i}$ such that for each $j \in \{1, \dots, m_i\}$ we have $\mathbb{P}(j \in S_i) = \mathbf{p}_{i,j}$.

Next, to give a tight rate for any given pair of smoothness structure and sampling strategy, we use a rather standard tool called *Expected Separable Overapproximation* (ESO) assumption – it provides us with smoothness parameters of the objective which “account” for the given sampling strategy.

Assumption H.1.4. Suppose that there is $\mathbf{v}_i \in \mathbb{R}^{m_i}$ such for each client we have:

$$\mathbb{E} \left[\left\| \sum_{j \in S_i} \mathbf{M}_{i,j}^{\frac{1}{2}} h_{i,j} \right\|^2 \right] \leq \sum_{j=1}^{m_i} \mathbf{p}_{i,j} \mathbf{v}_{i,j} \|h_{i,j}\|^2, \quad (\text{H.3})$$

for all $1 \leq i \leq n$, $h_{i,j} \in \mathbb{R}^{m_i}$, and $j \in \{1, \dots, m_i\}$.

Lastly, denote \mathbf{p}_i to be the probability that worker i is active and $\mathbf{1}^{(N_t)} \in \mathbb{R}^{m_i}$ to be the vector of ones.

The resulting algorithm is stated as Algorithm 62.

Next, Theorems H.1.5 and H.1.6 present convergence rate of Algorithm 62 (SAGA and SVRG variant, respectively).

Theorem H.1.5. Suppose that Assumptions H.1.3 and H.1.4 hold. Let

$$\alpha = \min \left\{ \min_{j \in \{1, \dots, m_i\}, 1 \leq i \leq n} \frac{N(1-p)\mathbf{p}_{i,j}\mathbf{p}_i}{4v_j + N_n^\mu}, \frac{np}{4\lambda + \mu} \right\}.$$

Algorithm 62 L2SGD++: Loopless Local SGD with Variance Reduction and Partial Participation

Input: $x_1^0, \dots, x_n^0 \in \mathbb{R}^d$, # parallel units n , each of them owns m_i data points (for $1 \leq i \leq n$), distributions \mathcal{D}_i over subsets of $\{1, \dots, m_i\}$, distribution \mathcal{D} over subsets of $\{1, 2, \dots, n\}$, aggregation probability p , stepsize α
 $\mathbf{J}_i^0 = 0 \in \mathbb{R}^{d \times m_i}$, $\Psi_i^0 = 0 \in \mathbb{R}^d$ (for $i = 1, \dots, n$)

for $k = 0, 1, 2, \dots$ **do**
 $\xi = 1$ with probability p and 0 with probability $1 - p$
 if $\xi = 0$ **then**
 Sample $S \sim \mathcal{D}$
 All **Devices** $i \in S$:
 Sample $S_i \sim \mathcal{D}_i$; $S_i \subseteq \{1, \dots, m_i\}$ (independently on each machine)
 Observe $\nabla f_{i,j}(x_i^k)$ for all $j \in S_i$
 $g_i^k = \frac{1}{N(1-p)p_i} \left(\sum_{j \in S_i} p_{i,j}^{-1} \left(\nabla f_{i,j}(x_i^k) - (\mathbf{J}_i^k)_{:,j} \right) \right) + \frac{1}{N} \mathbf{J}_i^k \mathbf{1}^{(N_i)} + i^{-1} \Psi_i^k$
 $x_i^{k+1} = \text{prox}_{\alpha\psi_i}(x_i^k - \alpha g_i^k)$
 For all $j \in \{1, \dots, m_i\}$ set $\mathbf{J}_{:,j}^{k+1} = \begin{cases} \begin{cases} \nabla f_{i,j}(x_i^k) & \text{if } j \in S_i \\ \mathbf{J}_{:,j}^k & \text{otherwise} \end{cases} & \text{if SAGA} \\ \begin{cases} \nabla f_{i,j}(x_i^k); & \text{w. p. } p_i \\ \mathbf{J}_{:,j}^k & \text{otherwise} \end{cases} & \text{if L-SVRG} \end{cases}$
 Set $\Psi_i^{k+1} = \Psi_i^k$
 All **Devices** $i \notin S$:
 $g_i^k = \frac{1}{N} \mathbf{J}_i^k \mathbf{1}^{(N_i)} + i^{-1} \Psi_i^k$
 $x_i^{k+1} = \text{prox}_{\alpha\psi_i}(x_i^k - \alpha g_i^k)$
 Set $\mathbf{J}_i^{k+1} = \mathbf{J}_i^k$, $\Psi_i^{k+1} = \Psi_i^k$
 else
 Master computes the average $\bar{x}^k = \frac{1}{n} \sum_{i=1}^n x_i^k$
 Master does for all $i = 1, \dots, n$:
 $g_i^k = p^{-1} \lambda(x_i^k - \bar{x}^k) - (p^{-1} - 1) i^{-1} \Psi_i^k + \frac{1}{N} \mathbf{J}_i^k \mathbf{1}^{(N_i)}$
 Set $x_i^{k+1} = \text{prox}_{\alpha\psi_i}(x_i^k - \alpha g_i^k)$
 Set $\Psi_i^{k+1} = \lambda(x_i^k - \bar{x}^k)$, $\mathbf{J}_i^{k+1} = \mathbf{J}_i^k$
 end if
end for

Then the iteration complexity of Algorithm 62 (SAGA option) is

$$\max \left\{ \max_{j \in \{1, \dots, m_i\}, 1 \leq i \leq n} \left(\frac{4v_j \frac{n}{N} + \mu}{\mu(1-p)p_{i,j}p_i} \right), \frac{4\lambda + \mu}{p\mu} \right\} \log \frac{1}{\varepsilon}.$$

Theorem H.1.6. Suppose that Assumptions H.1.3 and H.1.4 hold. Let

$$\alpha = \min \left\{ \min_{j \in \{1, \dots, m_i\}, 1 \leq i \leq n} \frac{N(1-p)p_i}{4 \frac{v_j}{p_{i,j}} + N \frac{\mu}{n} p_i^{-1}}, \frac{pn}{4\lambda + \mu} \right\}.$$

Then the iteration complexity of Algorithm 62 (LSVRG option) is

$$\max \left\{ \max_{j \in \{1, \dots, m_i\}, 1 \leq i \leq n} \left(\frac{4v_j \frac{n}{N p_{i,j}} + \mu p_i^{-1}}{p_i \mu (1-p)} \right), \frac{4\lambda + \mu}{p\mu} \right\} \log \frac{1}{\varepsilon}.$$

Remark 35. Algorithm 60 is a special case of Algorithm 20 which is in turn special case of Algorithm 62. Similarly, Theorem 60 is a special case of Theorem 8.5.2 which is again special case of Theorem H.1.5.

H.1.4 Local stochastic algorithms

In this section, we present two more algorithms – Local SGD with partial variance reduction (Algorithm 64) and Local SGD without variance reduction (Algorithm 63). While Algorithm 63 uses no control variates at all (thus is essentially Algorithm 19 where local gradient descent steps are replaced with local SGD steps), Algorithm 64 constructs control variates for Φ only, resulting in locally drifted SGD algorithm (with the constant drift between each consecutive rounds of communication). While we do not present the convergence rates of the methods here, we shall notice they can be easily obtained using the framework from [55].

Algorithm 63 Loopless Local SGD (L2SGD)

Input: $x_1^0 = \dots = x_n^0 \in \mathbb{R}^d$, stepsize α , probability p
for $k = 0, 1, 2, \dots$ **do**
 $\xi = 1$ with probability p and 0 with probability $1 - p$
 if $\xi = 0$ **then**
 All **Devices** $i = 1, \dots, n$:
 Sample $j \in \{1, \dots, m\}$ (uniformly at random)
 $g_i^k = \frac{1}{n(1-p)} (\nabla f_{i,j}(x_i^k))$
 $x_i^{k+1} = x_i^k - \alpha g_i^k$
 else
 Master computes the average $\bar{x}^k = \frac{1}{n} \sum_{i=1}^n x_i^k$
 Master does for all $i = 1, \dots, n$:
 $g_i^k = \frac{\lambda}{np} (x_i^k - \bar{x}^k)$
 Set $x_i^{k+1} = x_i^k - \alpha g_i^k$
 end if
end for

Algorithm 64 Loopleless Local SGD with partial variance reduction (L2SGD2)

Input: $x_1^0 = \dots = x_n^0 \in \mathbb{R}^d$, stepsize α , probability p
 $\Psi_i^0 = 0 \in \mathbb{R}^d$ (for $i = 1, \dots, n$)
for $k = 0, 1, 2, \dots$ **do**
 $\xi = 1$ with probability p and 0 with probability $1 - p$
 if $\xi = 0$ **then**
 All **Devices** $i = 1, \dots, n$:
 Sample $j \in \{1, \dots, m\}$ (uniformly at random)
 $g_i^k = \frac{1}{n(1-p)} (\nabla f_{i,j}(x_i^k)) + \frac{1}{n} \Psi_i^k$
 $x_i^{k+1} = x_i^k - \alpha g_i^k$
 Set $\Psi_i^{k+1} = \Psi_i^k$
 else
 Master computes the average $\bar{x}^k = \frac{1}{n} \sum_{i=1}^n x_i^k$
 Master does for all $i = 1, \dots, n$:
 $g_i^k = \frac{\lambda}{np} (x_i^k - \bar{x}^k) - \frac{p-1}{n} \Psi_i^k$
 Set $x_i^{k+1} = x_i^k - \alpha g_i^k$
 Set $\Psi_i^{k+1} = \lambda(x_i^k - \bar{x}^k)$
 end if
end for

H.2 Missing lemmas and proofs

H.2.1 Gradient and Hessian of Φ

Lemma H.2.1. *Let \mathbf{I} be the $d \times d$ identity matrix and \mathbf{I}_n be $n \times n$ identity matrix. Then, we have*

$$\nabla^2 \Phi(x) = \frac{1}{n} \left(\mathbf{I}_n - \frac{1}{n} e e^\top \right) \otimes \mathbf{I} \quad \text{and} \quad \nabla \Phi(x) = \frac{1}{n} \begin{pmatrix} x - \begin{pmatrix} \bar{x} \\ \vdots \\ \bar{x} \\ \vdots \\ \bar{x} \end{pmatrix} \end{pmatrix}.$$

Furthermore, $L_\Phi = \frac{1}{n}$.

Proof. Let \mathbf{O} the $d \times d$ zero matrix and let

$$\mathbf{Q}_i \stackrel{\text{def}}{=} [\underbrace{\mathbf{O}, \dots, \mathbf{O}}_{i-1}, \mathbf{I}, \underbrace{\mathbf{O}, \dots, \mathbf{O}}_{n-i}] \in \mathbb{R}^{d \times dn}$$

and $\mathbf{Q} \stackrel{\text{def}}{=} [\mathbf{I}, \dots, \mathbf{I}] \in \mathbb{R}^{d \times dn}$. Note that $x_i = \mathbf{Q}_i x$, and $\bar{x} = \frac{1}{n} \mathbf{Q} x$. So,

$$\Phi(x) = \frac{1}{2n} \sum_{i=1}^n \left\| \mathbf{Q}_i x - \frac{1}{n} \mathbf{Q} x \right\|^2 = \frac{1}{2n} \sum_{i=1}^n \left\| \left(\mathbf{Q}_i - \frac{1}{n} \mathbf{Q} \right) x \right\|^2.$$

The Hessian of Φ is

$$\begin{aligned}
\nabla^2 \Phi(x) &= \frac{1}{n} \sum_{i=1}^n \left(\mathbf{Q}_i - \frac{1}{n} \mathbf{Q} \right)^\top \left(\mathbf{Q}_i - \frac{1}{n} \mathbf{Q} \right) \\
&= \frac{1}{n} \sum_{i=1}^n \left(\mathbf{Q}_i^\top \mathbf{Q}_i - \frac{1}{n} \mathbf{Q}_i^\top \mathbf{Q} - \frac{1}{n} \mathbf{Q}^\top \mathbf{Q}_i + \frac{1}{n^2} \mathbf{Q}^\top \mathbf{Q} \right) \\
&= \frac{1}{n} \sum_{i=1}^n \mathbf{Q}_i^\top \mathbf{Q}_i - \frac{1}{n} \sum_{i=1}^n \frac{1}{n} \mathbf{Q}_i^\top \mathbf{Q} - \frac{1}{n} \sum_{i=1}^n \frac{1}{n} \mathbf{Q}^\top \mathbf{Q}_i + \frac{1}{n} \sum_{i=1}^n \frac{1}{n^2} \mathbf{Q}^\top \mathbf{Q} \\
&= \frac{1}{n} \sum_{i=1}^n \mathbf{Q}_i^\top \mathbf{Q}_i - \frac{1}{n^2} \mathbf{Q}^\top \mathbf{Q} \\
&= \frac{1}{n} \begin{pmatrix} \left(1 - \frac{1}{n}\right) \mathbf{I} & -\frac{1}{n} \mathbf{I} & -\frac{1}{n} \mathbf{I} & \cdots & -\frac{1}{n} \mathbf{I} \\ -\frac{1}{n} \mathbf{I} & \left(1 - \frac{1}{n}\right) \mathbf{I} & -\frac{1}{n} \mathbf{I} & \cdots & -\frac{1}{n} \mathbf{I} \\ -\frac{1}{n} \mathbf{I} & -\frac{1}{n} \mathbf{I} & \left(1 - \frac{1}{n}\right) \mathbf{I} & \cdots & -\frac{1}{n} \mathbf{I} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ -\frac{1}{n} \mathbf{I} & -\frac{1}{n} \mathbf{I} & -\frac{1}{n} \mathbf{I} & \cdots & \left(1 - \frac{1}{n}\right) \mathbf{I} \end{pmatrix} \\
&= \frac{1}{n} \begin{pmatrix} \left(1 - \frac{1}{n}\right) & -\frac{1}{n} & -\frac{1}{n} & \cdots & -\frac{1}{n} \\ -\frac{1}{n} & \left(1 - \frac{1}{n}\right) & -\frac{1}{n} & \cdots & -\frac{1}{n} \\ -\frac{1}{n} & -\frac{1}{n} & \left(1 - \frac{1}{n}\right) & \cdots & -\frac{1}{n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ -\frac{1}{n} & -\frac{1}{n} & -\frac{1}{n} & \cdots & \left(1 - \frac{1}{n}\right) \end{pmatrix} \otimes \mathbf{I} \\
&= \frac{1}{n} \left(\mathbf{I}_n - \frac{1}{n} \mathbf{e} \mathbf{e}^\top \right) \otimes \mathbf{I}.
\end{aligned}$$

Notice that $\mathbf{I}_n - \frac{1}{n} \mathbf{e} \mathbf{e}^\top$ is a circulant matrix, with eigenvalues 1 (multiplicity $n - 1$) and 0 (multiplicity 1). Since the eigenvalues of a Kronecker product of two matrices are the products of pairs of eigenvalues of the these matrices, we have

$$\lambda_{\max}(\nabla^2 \Phi(x)) = \lambda_{\max} \left(\frac{1}{n} \left(\mathbf{I}_n - \frac{1}{n} \mathbf{e} \mathbf{e}^\top \right) \otimes \mathbf{I} \right) = \frac{1}{n} \lambda_{\max} \left(\mathbf{I}_n - \frac{1}{n} \mathbf{e} \mathbf{e}^\top \right) = \frac{1}{n}.$$

So, $L_\Phi = \frac{1}{n}$.

The gradient of Φ is given by

$$\begin{aligned}
\nabla\Phi(x) &= \frac{1}{n} \sum_{i=1}^n \left(\mathbf{Q}_i - \frac{1}{n} \mathbf{Q} \right)^\top \left(\mathbf{Q}_i - \frac{1}{n} \mathbf{Q} \right) x \\
&= \frac{1}{n} \sum_{i=1}^n \left(\mathbf{Q}_i^\top \mathbf{Q}_i - \frac{1}{n} \mathbf{Q}_i^\top \mathbf{Q} - \frac{1}{n} \mathbf{Q}^\top \mathbf{Q}_i + \frac{1}{n^2} \mathbf{Q}^\top \mathbf{Q} \right) x \\
&= \frac{1}{n} \sum_{i=1}^n \left[\begin{pmatrix} 0 \\ \vdots \\ 0 \\ x_i \\ 0 \\ \vdots \\ 0 \end{pmatrix} - \begin{pmatrix} 0 \\ \vdots \\ 0 \\ \bar{x} \\ 0 \\ \vdots \\ 0 \end{pmatrix} - \begin{pmatrix} x_i/n \\ \vdots \\ x_i/n \\ x_i/n \\ x_i/n \\ \vdots \\ x_i/n \end{pmatrix} + \begin{pmatrix} \bar{x}/n \\ \vdots \\ \bar{x}/n \\ \bar{x}/n \\ \bar{x}/n \\ \vdots \\ \bar{x}/n \end{pmatrix} \right] \\
&= \frac{1}{n} \left(\sum_{i=1}^n \begin{pmatrix} 0 \\ \vdots \\ 0 \\ x_i \\ 0 \\ \vdots \\ 0 \end{pmatrix} - \sum_{i=1}^n \begin{pmatrix} 0 \\ \vdots \\ 0 \\ \bar{x} \\ 0 \\ \vdots \\ 0 \end{pmatrix} - \sum_{i=1}^n \begin{pmatrix} x_i/n \\ \vdots \\ x_i/n \\ x_i/n \\ x_i/n \\ \vdots \\ x_i/n \end{pmatrix} + \sum_{i=1}^n \begin{pmatrix} \bar{x}/n \\ \vdots \\ \bar{x}/n \\ \bar{x}/n \\ \bar{x}/n \\ \vdots \\ \bar{x}/n \end{pmatrix} \right) \\
&= \frac{1}{n} \left(x - \begin{pmatrix} \bar{x} \\ \vdots \\ \bar{x} \\ \bar{x} \\ \bar{x} \\ \vdots \\ \bar{x} \end{pmatrix} - \begin{pmatrix} \bar{x} \\ \vdots \\ \bar{x} \\ \bar{x} \\ \bar{x} \\ \vdots \\ \bar{x} \end{pmatrix} + \begin{pmatrix} \bar{x} \\ \vdots \\ \bar{x} \\ \bar{x} \\ \bar{x} \\ \vdots \\ \bar{x} \end{pmatrix} \right) \\
&= \frac{1}{n} \left(x - \begin{pmatrix} \bar{x} \\ \vdots \\ \bar{x} \\ \bar{x} \\ \bar{x} \\ \vdots \\ \bar{x} \end{pmatrix} \right).
\end{aligned}$$

□

H.2.2 Proof of Theorem 8.3.2

For any $\lambda, \theta \geq 0$ we have

$$f(x(\lambda)) + \lambda\Phi(x(\lambda)) \leq f(x(\theta)) + \lambda\Phi(x(\theta)) \quad (\text{H.4})$$

$$f(x(\theta)) + \theta\Phi(x(\theta)) \leq f(x(\lambda)) + \theta\Phi(x(\lambda)). \quad (\text{H.5})$$

By adding inequalities (H.4) and (H.5), we get

$$(\theta - \lambda)(\Phi(x(\lambda)) - \Phi(x(\theta))) \geq 0,$$

which means that $\Phi(x(\lambda))$ is decreasing in λ . Assume $\lambda \geq \theta$. From the (H.5) we get

$$f(x(\lambda)) \geq f(x(\theta)) + \theta(\Phi(x(\theta)) - \Phi(x(\lambda))) \geq f(x(\theta)),$$

where the last inequality follows since $\theta \geq 0$ and since $\Phi(x(\theta)) \geq \Phi(x(\lambda))$. So, $f(x(\lambda))$ is increasing.

Notice that since Φ is a non-negative function and since $x(\lambda)$ minimizes F and $\Phi(x(\infty)) = 0$, we have

$$f(x(0)) \leq f(x(\lambda)) \leq f(x(\lambda)) + \lambda\Phi(x(\lambda)) \leq f(x(\infty)),$$

which implies (8.5) and (8.6).

H.2.3 Proof of Theorem 8.3.3

The equation $\nabla F(x^*(\lambda)) = 0$ can be equivalently written as

$$\nabla f_i(x_i^*(\lambda)) + \lambda(x_i^*(\lambda) - \bar{x}^*(\lambda)) = 0, \quad i = 1, 2, \dots, n,$$

which is identical to (8.7). Averaging these identities over i , we get $\bar{x}^*(\lambda) = \bar{x}^*(\lambda) - \frac{1}{\lambda} \frac{1}{n} \sum_{i=1}^n \nabla f_i(x_i^*(\lambda))$, which implies

$$\sum_{i=1}^n \nabla f_i(x_i^*(\lambda)) = 0.$$

Further, we have

$$\Phi(x^*(\lambda)) = \frac{1}{2n} \sum_{i=1}^n \|x_i^*(\lambda) - \bar{x}^*(\lambda)\|^2 = \frac{1}{2n\lambda^2} \sum_{i=1}^n \|\nabla f_i(x_i^*(\lambda))\|^2 = \frac{1}{2\lambda^2} \|\nabla f(x^*(\lambda))\|^2,$$

as desired.

H.2.4 Proof of Lemma 8.4.2

We first have

$$\begin{aligned}
\mathbb{E} [\|g(x) - G(x^*)\|^2] &= (1-p) \left\| \frac{\nabla f(x)}{1-p} - \frac{\nabla f(x^*)}{1-p} \right\|^2 + p \left\| \lambda \frac{\nabla \Phi(x)}{p} - \lambda \frac{\nabla \Phi(x^*)}{p} \right\|^2 \\
&= \frac{1}{1-p} \|\nabla f(x) - \nabla f(x^*)\|^2 + \frac{\lambda^2}{p} \|\nabla \Phi(x) - \nabla \Phi(x^*)\|^2 \\
&\leq \frac{2L_f}{1-p} D_f(x, x^*) + \frac{2\lambda^2 L_\Phi}{p} D_\Phi(x, x^*) \\
&= \frac{2L}{n(1-p)} D_f(x, x^*) + \frac{2\lambda^2}{np} D_\Phi(x, x^*).
\end{aligned}$$

Since $D_f + \lambda D_\Phi = D_F$ and $\nabla F(x^*) = 0$, we can continue:

$$\begin{aligned}
\mathbb{E} [\|g(x) - G(x^*)\|^2] &\leq \frac{2}{n} \max \left\{ \frac{L}{1-p}, \frac{\lambda}{p} \right\} D_F(x, x^*) \\
&= \frac{2}{n} \max \left\{ \frac{L}{1-p}, \frac{\lambda}{p} \right\} (F(x) - F(x^*)).
\end{aligned}$$

Next, note that

$$\begin{aligned}
\sigma^2 &= \frac{1}{n^2} \sum_{i=1}^n \left(\frac{1}{1-p} \|\nabla f_i(x_i^*)\|^2 + \frac{\lambda^2}{p} \|x_i^* - \bar{x}^*\|^2 \right) \\
&= \frac{1}{1-p} \|\nabla f(x^*)\|^2 + \frac{\lambda^2}{p} \|\nabla \Phi(x^*)\|^2 \\
&= (1-p) \left\| \frac{\nabla f(x^*)}{1-p} \right\|^2 + p \left\| \frac{\lambda \nabla \Phi(x^*)}{p} \right\|^2 \\
&= \mathbb{E} [\|G(x^*)\|^2].
\end{aligned} \tag{H.6}$$

Therefore, we have

$$\begin{aligned}
\mathbb{E} [\|g(x)\|^2] &\leq \mathbb{E} [\|g(x) - G(x^*)\|^2] + 2\mathbb{E} [\|G(x^*)\|^2] \\
&\stackrel{\text{Lemma 8.4.2+(H.6)}}{\leq} 4\mathcal{L}(F(x) - F(x^*)) + 2\sigma^2
\end{aligned}$$

as desired.

H.2.5 Proof of Theorem 8.4.3

Follows from Lemma 8.4.2 by applying Theorem 3.1 from [60].

H.2.6 Proof of Corollary 8.4.4

Firstly, to minimize the total number of iterations, it suffices to minimize \mathcal{L} which is achieved with $p^* = \frac{\lambda}{L+\lambda}$. Let us look at the communication. Fix $\varepsilon > 0$, choose $\alpha = \frac{1}{2\mathcal{L}}$

and let $k = \frac{2n\mathcal{L}}{\mu} \log \frac{1}{\varepsilon}$, so that

$$\left(1 - \frac{\mu}{2n\mathcal{L}}\right)^k \leq \varepsilon.$$

The expected number of communications to achieve this goal is equal to

$$\begin{aligned} \text{Comm}_p &\stackrel{\text{def}}{=} p(1-p)k \\ &= p(1-p) \frac{2 \max \left\{ \frac{L}{1-p}, \frac{\lambda}{p} \right\}}{\mu} \log \frac{1}{\varepsilon} \\ &= \frac{2 \max \{pL, (1-p)\lambda\}}{\mu} \log \frac{1}{\varepsilon}. \end{aligned}$$

The quantity Comm_p is minimized by choosing any p such that $pL = (1-p)\lambda$, i.e., for $p = \frac{\lambda}{\lambda+L} = p^*$, as desired. The optimal expected number of communications is therefore equal to

$$\text{Comm}_{p^*} = \frac{2\lambda}{\lambda+L} \frac{L}{\mu} \log \frac{1}{\varepsilon}.$$

H.2.7 Proof of Corollary 8.5.3

Firstly, to minimize the total number of iterations, it suffices to solve

$$\min_p \max \left\{ \frac{4\tilde{L} + \mu m}{(1-p)\mu}, \frac{4\lambda + \mu}{p\mu} \right\},$$

which is achieved with $p = p^* = \frac{4\lambda + \mu}{4\tilde{L} + 4\lambda + (m+1)\mu}$. The expected number of communications to reach ε -solution is

$$\begin{aligned} \text{Comm}_p &= p(1-p) \max \left\{ \frac{4\tilde{L} + \mu m}{(1-p)\mu}, \frac{4\lambda + \mu}{p\mu} \right\} \log \frac{1}{\varepsilon} \\ &= \frac{\max \{p(4\tilde{L} + \mu m), (1-p)(4\lambda + \mu)\}}{\mu} \log \frac{1}{\varepsilon}. \end{aligned}$$

Minimizing the above in p yield $p = p^* = \frac{4\lambda + \mu}{4\tilde{L} + 4\lambda + (m+1)\mu}$, as desired. The optimal expected number of communications is therefore equal to

$$\text{Comm}_{p^*} = \frac{4\lambda + \mu}{4\tilde{L} + 4\lambda + (m+1)\mu} \left(4\frac{\tilde{L}}{\mu} + m \right) \log \frac{1}{\varepsilon}.$$

H.2.8 Proof of Theorems 8.5.2, H.1.5, and H.1.6

Note first that Algorithm 20 is a special case of Algorithm 62, and Theorem 8.5.2 immediately follows from Theorem H.1.5. Therefore it suffices to show Theorems H.1.5,

and H.1.6. In order to do so, we will cast Algorithm 62 as a special case of GJS (Algorithm 14). As a consequence, Theorem H.1.5 will be a special cases of Theorem 5.4.2.

Variance reduced local SGD as special case of GJS

Let $\Omega(i, j) \stackrel{\text{def}}{=} j + \sum_{l=1}^{i-1} m_l$ In order to case problem (H.1) as (5.1), denote $n \stackrel{\text{def}}{=} N + 1$, $f_{\Omega(i, j)}(x) \stackrel{\text{def}}{=} \frac{N+1}{N} f_{i, j}(x_i)$ and $f_n \stackrel{\text{def}}{=} (N + 1)\Phi$. Therefore the objective (H.1) becomes

$$\min_{x \in \mathbb{R}^{Nd}} \Upsilon(x) \stackrel{\text{def}}{=} \frac{1}{n} \sum_{j=1}^n f_j(x) + \psi(x). \quad (\text{H.7})$$

Let $v \in \mathbb{R}^{n-1}$ be such that $v_{\Omega(i, j)} = \frac{N+1}{N} v_{i, j}$ and as a consequence of (H.3) we have

$$\mathbb{E} \left[\left\| \sum_{j \in S_i} M_{i, j}^{\frac{1}{2}} h_{i, j} \right\|^2 \right] \leq \sum_{j=1}^{m_i} p_{i, j} v_{\Omega(i, j)} \|h_{i, j}\|^2, \quad \forall 1 \leq i \leq n, \forall h_{i, j} \in \mathbb{R}^d, j \in \{1, \dots, m_i\} \quad (\text{H.8})$$

At the same time, Υ is $\mu \stackrel{\text{def}}{=} \frac{\mu}{n}$ strongly convex.

Proof of Theorem H.1.5 and Theorem H.1.6

Let $e \in \mathbb{R}^d$ be a vector of ones and $p^i \in \mathbb{R}^N$ is such that $p_j^i = p_{i, j}$ if $j \in \{1, \dots, m_i\}$, otherwise $p_j^i = 0$. Given the notation, random operator \mathcal{U} is chosen as

$$\mathcal{U}\mathbf{X} = \begin{cases} (1-p)^{-1} \sum_{i=1}^n \left(p_i^{-1} e \left((p^i)^{-1} \right)^\top \right) \circ \left(\mathbf{X}_{:, m_i} \left(\sum_{j \in S_i} e_j e_j^\top \right) \right) & \text{w.p. } (1-p) \\ p^{-1} \mathbf{X}_{:, n} & \text{w.p. } p \end{cases}$$

We next give two options on how to update Jacobian – first one is SAGA-like, second one is SVRG like.

$$\begin{aligned} \text{SAGA-like:} \quad (\mathcal{S}\mathbf{X})_{:, m_i} &= \begin{cases} \mathbf{X}_{:, S_i} = \mathbf{X}_{:, m_i} \left(\sum_{j \in S_i} e_j e_j^\top \right), & \text{w.p. } (1-p)p_i, \\ 0 & \text{w.p. } (1-p)(1-p_i) + p \end{cases} \\ (\mathcal{S}\mathbf{X})_{:, n} &= \begin{cases} \mathbf{X}_{:, n} & \text{w.p. } p \\ 0 & \text{w.p. } 1-p \end{cases} \\ \text{SVRG-like:} \quad (\mathcal{S}\mathbf{X})_{:, m_i} &= \begin{cases} \mathbf{X}_{:, m_i} b_i; \quad b_i = \begin{cases} 1 & \text{w.p. } p_i \\ 0 & \text{w.p. } 1-p_i \end{cases} & \text{w.p. } (1-p)p_i \\ 0 & \text{w.p. } (1-p)(1-p_i) + p \end{cases} \\ (\mathcal{S}\mathbf{X})_{:, n} &= \begin{cases} \mathbf{X}_{:, n} & \text{w.p. } p \\ 0 & \text{w.p. } 1-p \end{cases} \end{aligned}$$

We can now proceed with the proof of Theorem H.1.5 and Theorem H.1.6. As

$\nabla f_i(x) - \nabla f_i(y) \in \text{Range}(\mathbf{M}_i)$, we must have

$$\mathbf{G}(x^k) - \mathbf{G}(x^*) = \mathcal{M}^\dagger \mathcal{M} (\mathbf{G}(x^k) - \mathbf{G}(x^*)) \quad (\text{H.9})$$

and

$$\mathbf{J}^k - \mathbf{G}(x^*) = \mathcal{M}^\dagger \mathcal{M} (\mathbf{J}^k - \mathbf{G}(x^*)). \quad (\text{H.10})$$

Due to (H.10), (H.9), inequalities (5.12) and (5.13) with choice $\mathbf{Y} = \mathcal{M}^{\dagger \frac{1}{2}} \mathbf{X}$ become respectively:

$$\begin{aligned} & \frac{2\alpha}{n^2} p^{-1} \|\mathbf{M}_n^{\frac{1}{2}} \mathbf{Y}_{:,n}\|^2 + \frac{2\alpha^2}{n^2} (1-p)^{-1} \sum_{i=1}^n \mathbb{E} \left[\left\| \mathbf{p}_i^{-1} \sum_{j \in S_i} \mathbf{p}_{i,j}^{-1} \mathbf{M}_{i,j}^{\frac{1}{2}} \mathbf{Y}_{:,j} \right\|^2 \right] + \|(\mathcal{I} - \mathbb{E}[\mathcal{S}])^{\frac{1}{2}} \mathcal{B}(\mathbf{Y})\|^2 \\ & \leq (1 - \alpha\mu) \|\mathcal{B}(\mathbf{Y})\|^2 \end{aligned} \quad (\text{H.11})$$

$$\begin{aligned} & \frac{2\alpha}{n^2} p^{-1} \|\mathbf{M}_n^{\frac{1}{2}} \mathbf{Y}_{:,n}\|^2 + \frac{2\alpha^2}{n^2} (1-p)^{-1} \sum_{i=1}^n \mathbb{E} \left[\left\| \mathbf{p}_i^{-1} \sum_{j \in S_i} \mathbf{p}_{i,j}^{-1} \mathbf{M}_{i,j}^{\frac{1}{2}} \mathbf{Y}_{:,j} \right\|^2 \right] + \|(\mathbb{E}[\mathcal{S}])^{\frac{1}{2}} \mathcal{B}(\mathbf{Y})\|^2 \leq \frac{1}{n} \|\mathbf{Y}\|^2 \end{aligned} \quad (\text{H.12})$$

Above, we have used

$$\mathbb{E} [\|\mathcal{U}\mathbf{X}e\|^2] = \mathbb{E} [\|\mathcal{U}\mathcal{M}^{\frac{1}{2}} \mathbf{Y}e\|^2] = p^{-1} \|\mathbf{M}_n^{\frac{1}{2}} \mathbf{Y}_{:,n}\|^2 + (1-p)^{-1} \sum_{i=1}^n \mathbb{E} \left[\left\| \mathbf{p}_i^{-1} \sum_{j \in S_i} \mathbf{p}_{i,j}^{-1} \mathbf{M}_{i,j}^{\frac{1}{2}} \mathbf{Y}_{:,j} \right\|^2 \right].$$

Note that $\mathbb{E}[\mathcal{S}(\mathbf{X})] = \mathbf{X} \cdot \text{Diag}((1-p)(\mathbf{p} \circ \mathbf{p}), p)$ where $\mathbf{p} \in \mathbb{R}^{n-1}$ such that $\mathbf{p}_{\Omega(i,j)} = \mathbf{p}_{i,j}$. Using (H.8), setting \mathcal{B} to be right multiplication with $\text{Diag}(b)$ and noticing that $\lambda_{\max} \mathbf{M}_n = n\lambda$ it suffices to have

$$\begin{aligned} & \frac{2\alpha}{n} p^{-1} \lambda + (1-p) b_n^2 \leq (1 - \alpha\mu) b_n^2 \\ & \frac{2\alpha}{n^2} (1-p)^{-1} \mathbf{p}_{i,j}^{-1} \mathbf{p}_i^{-1} v_{\Omega(i,j)} + (1 - (1-p) \mathbf{p}_{i,j} \mathbf{p}_i) b_j^2 \leq (1 - \alpha\mu) b_j^2 \quad \forall j \in \{1, \dots, m_i\}, i \leq n \end{aligned}$$

$$\frac{2\alpha}{n} p^{-1} \lambda + p b_n^2 \leq \frac{1}{n}$$

$$\frac{2\alpha}{n^2} (1-p)^{-1} \mathbf{p}_{i,j}^{-1} \mathbf{p}_i^{-1} v_{\Omega(i,j)} + (1-p) \mathbf{p}_{i,j} \mathbf{p}_i b_j^2 \leq \frac{1}{n} \quad \forall j \in \{1, \dots, m_i\}, i \leq n$$

for SAGA case and

$$\frac{2\alpha}{n} p^{-1} \lambda + (1-p) b_n^2 \leq (1 - \alpha\mu) b_n^2$$

$$\frac{2\alpha}{n^2} (1-p)^{-1} \mathbf{p}_{i,j}^{-1} \mathbf{p}_i^{-1} v_{\Omega(i,j)} + (1 - (1-p) \mathbf{p}_i \mathbf{p}_i) b_j^2 \leq (1 - \alpha\mu) b_j^2 \quad \forall j \in \{1, \dots, m_i\}, i \leq n$$

$$\frac{2\alpha}{n} p^{-1} \lambda + p b_n^2 \leq \frac{1}{n}$$

$$\frac{2\alpha}{n^2}(1-p)^{-1}p_{i,j}^{-1}p_i^{-1}v_{\Omega(i,j)} + (1-p)p_i p_j b_j^2 \leq \frac{1}{n} \quad \forall j \in \{1, \dots, m_i\}, i \leq n$$

for LSVRG case.

It remains to notice that to satisfy the SAGA case, it suffices to set $b_n^2 = \frac{1}{2np}$, $b_{\Omega(i,j)}^2 = \frac{1}{2n(1-p)p_{i,j}p_i}$ (for $j \in \{1, \dots, m_i\}, i \leq n$) and $\alpha = \min \left\{ \min_{j \in \{1, \dots, m_i\}, 1 \leq i \leq n} \frac{n(1-p)p_{i,j}p_i}{4v_{\Omega(i,j)} + n\mu}, \frac{p}{4\lambda + \mu} \right\}$.

To satisfy LSVRG case, it remains to set $b_n^2 = \frac{1}{2np}$, $b_{\Omega(i,j)}^2 = \frac{1}{2n(1-p)p_{i,j}p_i}$ (for $j \in \{1, \dots, m_i\}, i \leq n$) and $\alpha = \min \left\{ \min_{j \in \{1, \dots, m_i\}, 1 \leq i \leq n} \frac{n(1-p)p_i}{4 \frac{v_{\Omega(i,j)}}{p_{i,j}} + n\mu p_i^{-1}}, \frac{p}{4\lambda + \mu} \right\}$.

The last step to establish is to recall that $n = N + 1$, $v_{\Omega(i,j)} = \frac{N+1}{N}v_{i,j}$ and $\mu = \frac{\mu}{n}$ and note that the iteration complexity is $\frac{1}{\alpha\mu} \log \frac{1}{\varepsilon} = \frac{n}{\alpha\mu} \log \frac{1}{\varepsilon}$.

Proof of Theorem 8.5.2

To obtain convergence rate of Theorem 8.5.2, it remains to use Theorem H.1.5 with $p_i = 1, m_i = m$ ($\forall i \leq n$), where each machine samples (when the aggregation is not performed) individual data points with probability $\frac{1}{m}$ and thus $p_j = \frac{1}{m}$ (for all $j \leq N$). The last remaining thing is to realize that $v_j = \tilde{L}$ for all $j \leq N$.

Appendix I

Appendix for Chapter 9

I.1 Missing lemmas and proofs from Section 9.3

I.1.1 Explicit update

Lemma I.1.1. *Let $x^+ = \arg \min_y \langle g', y - x \rangle + \frac{H'}{2} \|x - y\|^2 + \frac{M'}{6} \|x - y\|^3$, where $H', M' > 0$. Then we have*

$$x^+ = x - \frac{2g'}{H' + \sqrt{H'^2 + 2M'\|g'\|}} \quad (I.1)$$

Proof. By first-order optimality conditions we have $g' + H'(x^+ - x) + \frac{M'}{2} \|x^+ - x\| (x^+ - x) = 0$ which immediately yields

$$x^+ = x - \frac{g'}{H' + \frac{M'}{2} \|x^+ - x\|}. \quad (I.2)$$

Rearranging the terms and taking the norm we have $\frac{M'}{2} \|x^+ - x\|^2 + H' \|x^+ - x\| + \|g'\| = 0$. Solving the quadratic equation we arrive at

$$\|x^+ - x\| = \frac{\sqrt{H'^2 + 2M'\|g'\|} - H'}{M'}.$$

Plugging it back to (I.2), we get (I.1). □

I.1.2 Proof of Lemma 9.3.3

First, note that

$$\begin{aligned} & D_f(x^+, x) - \frac{1}{2} (x^+ - x)^\top \nabla^2 f(x) (x^+ - x) \\ &= \int_0^1 \langle \nabla f(x + t(x^+ - x)) - f(x), x^+ - x \rangle dt - \frac{1}{2} (x^+ - x)^\top \nabla^2 f(x) (x^+ - x) \\ &= \int_0^1 \int_0^1 \langle t \nabla^2 f(x + st(x^+ - x)), x^+ - x, x^+ - x \rangle ds dt - \frac{1}{2} (x^+ - x)^\top \nabla^2 f(x) (x^+ - x) \\ &= \int_0^1 \int_0^1 \langle t \nabla^2 f(x + st(x^+ - x)) - \nabla^2 f(x), x^+ - x, x^+ - x \rangle ds dt \\ &= \int_0^1 \int_0^1 \int_0^1 \langle t^2 s \nabla^3 f(x + rst(x^+ - x)), x^+ - x, x^+ - x, x^+ - x \rangle dr ds dt. \end{aligned}$$

Using (9.2) we get

$$\begin{aligned}
& |f(x^+) - f(x) + \langle \nabla f(x), \mathbf{S}h \rangle + \frac{1}{2}h^\top \nabla_{\mathbf{S}}^2 f(x)h| \\
& \stackrel{(9.2)}{=} \left| \int_0^1 \int_0^1 \int_0^1 \langle t^2 s \nabla^3 f(x + rst\mathbf{S}h), \mathbf{S}h, \mathbf{S}h, \mathbf{S}h \rangle dr ds dt \right| \\
& \stackrel{(9.4)}{\leq} \int_0^1 \int_0^1 \int_0^1 t^2 s M_{\mathbf{S}} \|\mathbf{S}h\|^3 dr ds dt \\
& = \frac{M_{\mathbf{S}}}{6} \|\mathbf{S}h\|^3.
\end{aligned}$$

I.1.3 Proof of Lemma 9.3.2

First, $M \geq M_{\mathbf{S}}$ is trivial. At the same time $M = M_{\mathbf{S}}$ if $\nabla^3 f(x)$ is identity tensor always (which is clearly feasible) – thus the inequality is tight.

To show sharpness of $M_{\mathbf{S}} \geq \left(\frac{\tau}{d}\right)^{\frac{3}{2}} M$, consider $f(x) = \frac{1}{6}(x^\top e)^3$. In this case, we have¹ $\nabla^3 f(x) = [e]^3$ and $\mathbf{S} = e_i$. In such case, $M = d^{\frac{3}{2}}$ and $M_{\mathbf{S}} = \tau^{\frac{3}{2}}$. Note that f is non-convex in this example. However, f is convex on a set where $x e \geq 0$, where the argument follows through.

I.2 Proofs for Section 9.6

I.2.1 Proof of Lemma 9.6.2

Let $\text{Tr}(\mathbf{A})$ be a trace of square matrix \mathbf{A} . We have

$$\begin{aligned}
\mathbb{E}[\tau(\mathbf{S})] &= \mathbb{E}[\text{Tr}(\mathbf{I}^{\tau(\mathbf{S})})] = \mathbb{E}[\text{Tr}(\mathbf{S}^\top \mathbf{S} (\mathbf{S}^\top \mathbf{S})^{-1})] = \mathbb{E}[\text{Tr}(\mathbf{S} (\mathbf{S}^\top \mathbf{S})^{-1} \mathbf{S}^\top)] \\
&= \text{Tr}(\mathbb{E}[\mathbf{S} (\mathbf{S}^\top \mathbf{S})^{-1} \mathbf{S}^\top]) \stackrel{(9.7)}{=} \text{Tr}\left(\frac{\tau}{d} \mathbf{I}^d\right) = \tau.
\end{aligned}$$

I.2.2 Proof of Lemma 9.6.7

For any $h' \in \mathbb{R}^d$ denote

$$\Omega_{\mathbf{S}}(x; h') \stackrel{\text{def}}{=} f(x) + \langle \nabla f(x), \mathbf{Z}h' \rangle + \frac{1}{2} \langle \nabla^2 f(x) \mathbf{Z}h', \mathbf{Z}h' \rangle + \frac{H}{6} \|\mathbf{Z}h'\|^3 + \psi(x + \mathbf{Z}h')$$

and

$$T_{\mathbf{S}}(x^k) \stackrel{\text{def}}{=} \arg \min_{h' \in \mathbb{R}^d} \Omega_{\mathbf{S}}(x; h').$$

Then, for any fixed $y \in \mathbb{R}^d$ we have

$$F(x^{k+1}) \stackrel{(9.5)}{\leq} \Omega_{\mathbf{S}}(x^k; T_{\mathbf{S}}(x^k)) \leq \Omega_{\mathbf{S}}(x^k; y - x^k).$$

¹By $[e] \in \mathbb{R}^{d \times d \times d}$ we mean third order product of vector e .

Therefore,

$$\begin{aligned}
\mathbb{E} [F(x^{k+1}) | x^k] &\leq \mathbb{E} [\Omega_{\mathbf{S}}(x^k; y - x^k)] \\
&= f(x^k) + \frac{\tau}{d} \langle \nabla f(x^k), y - x^k \rangle + \mathbb{E} \left[\frac{1}{2} \langle \mathbf{Z} \nabla^2 f(x^k) \mathbf{Z} (y - x^k), y - x^k \rangle \right] \\
&\quad + \frac{M}{6} \mathbb{E} [\|\mathbf{Z}(y - x^k)\|^3] + \mathbb{E} [\psi(x + \mathbf{Z}(y - x^k))].
\end{aligned}$$

Let us get rid of the expectations above. Firstly, we have

$$\begin{aligned}
\mathbb{E} [\psi(x + \mathbf{Z}(y - x^k))] &= \mathbb{E} [\langle \psi'((\mathbf{I}^d - \mathbf{Z})x^k + \mathbf{Z}y), e \rangle] \\
&= \mathbb{E} [\langle (\mathbf{I}^d - \mathbf{Z})\psi'(x^k), e \rangle] + \mathbb{E} [\langle \mathbf{Z}\psi'(y), e \rangle] \\
&= \left(1 - \frac{\tau}{d}\right) \psi(x^k) + \frac{\tau}{d} \psi(y).
\end{aligned}$$

For the cubed norm it can be estimated as follows

$$\mathbb{E} [\|\mathbf{Z}h'\|^3] \leq \|h'\| \cdot \mathbb{E} [\|\mathbf{Z}h'\|^2] = \frac{\tau}{d} \|h'\|^3, \quad \forall h' \in \mathbb{R}^d.$$

Lastly, note that

$$\begin{aligned}
&\mathbb{E} [\mathbf{Z} \nabla^2 f(x^k) \mathbf{Z}] \\
&= \mathbb{E} \left[\mathbf{Z} (\nabla^2 f(x^k))^{\frac{1}{2}} \right] \mathbb{E} \left[(\nabla^2 f(x^k))^{\frac{1}{2}} \mathbf{Z} \right] \\
&\quad + \mathbb{E} \left[\left(\mathbf{Z} (\nabla^2 f(x^k))^{\frac{1}{2}} - \mathbb{E} \left[\mathbf{Z} (\nabla^2 f(x^k))^{\frac{1}{2}} \right] \right) \left(\mathbf{Z} (\nabla^2 f(x^k))^{\frac{1}{2}} - \mathbb{E} \left[\mathbf{Z} (\nabla^2 f(x^k))^{\frac{1}{2}} \right] \right)^{\top} \right] \\
&= \frac{\tau^2}{d^2} \nabla^2 f(x^k) + \mathbb{E} \left[\left(\mathbf{Z} - \frac{\tau}{d} \mathbf{I}^d \right) \nabla^2 f(x^k) \left(\mathbf{Z} - \frac{\tau}{d} \mathbf{I}^d \right) \right] \\
&\leq \frac{\tau^2}{d^2} \nabla^2 f(x^k) + L \mathbb{E} \left[\left(\mathbf{Z} - \frac{\tau}{d} \mathbf{I}^d \right)^2 \right] \\
&= \frac{\tau^2}{d^2} \nabla^2 f(x^k) + \frac{\tau(d - \tau)}{d^2} L \mathbf{I}^d.
\end{aligned}$$

Therefore, we conclude

$$\begin{aligned}
\mathbb{E} [F(x^{k+1}) | x^k] &\leq f(x^k) + \frac{\tau}{d} \langle \nabla f(x^k), y - x^k \rangle + \frac{\tau(d - \tau)}{d^2} \cdot \frac{L}{2} \|y - x^k\|^2 \\
&\quad + \frac{\tau^2}{d^2} \cdot \frac{1}{2} \langle \nabla^2 f(x^k)(y - x^k), y - x^k \rangle + \frac{\tau}{d} \cdot \frac{M}{6} \|y - x^k\|^3 \\
&\quad + \frac{\tau}{d} \psi(y) + \left(1 - \frac{\tau}{d}\right) \psi(x^k).
\end{aligned}$$

Finally, by convexity and from Lipschitz continuity of the Hessian (9.5), we have the

following upper estimate:

$$\begin{aligned}
& \langle \nabla f(x^k), y - x^k \rangle + \frac{\tau}{d} \cdot \frac{1}{2} \langle \nabla^2 f(x^k)(y - x^k), y - x^k \rangle \\
&= \frac{d-\tau}{d} \langle \nabla f(x^k), y - x^k \rangle \\
&+ \frac{\tau}{d} \left(\langle \nabla f(x^k), y - x^k \rangle + \frac{1}{2} \langle \nabla^2 f(x^k)(y - x^k), y - x^k \rangle \right) \\
&\leq \frac{d-\tau}{d} \left(f(y) - f(x^k) \right) + \frac{\tau}{d} \left(f(y) - f(x^k) + \frac{M}{6} \|y - x^k\|^3 \right) \\
&\leq f(y) - f(x^k) + \frac{M}{6} \|y - x^k\|^3.
\end{aligned}$$

which completes the proof. \square

I.2.3 Proof of Theorem 9.6.8

Let us denote the following auxiliary sequences:

$$a_k \stackrel{\text{def}}{=} k^2, \quad A_k \stackrel{\text{def}}{=} A_0 + \sum_{i=1}^k a_i, \quad k \geq 1,$$

and

$$A_0 \stackrel{\text{def}}{=} \frac{4}{3} \left(\frac{d}{\tau} \right)^3.$$

Then, we have an estimate

$$A_k = A_0 + \sum_{i=1}^k i^2 \geq A_0 + \int_0^k x^2 dx = A_0 + \frac{k^3}{3}. \quad (1.3)$$

Now, let us fix iteration counter $k \geq 0$ and set

$$\alpha_k \stackrel{\text{def}}{=} \frac{d}{\tau} \frac{a_{k+1}}{A_{k+1}} \Leftrightarrow 1 - \frac{\tau}{d} \alpha_k = \frac{A_k}{A_{k+1}}.$$

Note that we have $\alpha_k \leq 1$ by the choice of A_0 , since it holds

$$\max_{\tau \geq 0} \frac{\tau^2}{A_0 + \frac{\tau^3}{3}} = \frac{\tau}{d}.$$

Let us plug $y \equiv \alpha_k x^* + (1 - \alpha_k)x^k$ into (9.8). By convexity we obtain

$$\begin{aligned}
& \mathbb{E} [F(x^{k+1}) | x^k] \\
& \leq \left(1 - \frac{\tau}{d}\right) F(x^k) + \frac{\tau}{d} \alpha_k F^* + \frac{\tau}{d} (1 - \alpha_k) F(x^k) \\
& \quad + \frac{\tau}{d} \left(\frac{d - \tau}{d} \frac{L \|x^k - x^*\|^2}{2} \alpha_k^2 + \frac{M \|x^k - x^*\|^3}{3} \alpha_k^3 \right) \\
& = \frac{A_k}{A_{k+1}} F(x^k) + \frac{a_{k+1}}{A_{k+1}} F^* + \frac{d - \tau}{\tau} \frac{L}{d} \frac{\|x^k - x^*\|^2}{2} \left(\frac{a_{k+1}}{A_{k+1}} \right)^2 \\
& \quad + \left(\frac{d}{\tau} \right)^2 \frac{M \|x^k - x^*\|^3}{3} \left(\frac{a_{k+1}}{A_{k+1}} \right)^3 \\
& \leq \frac{A_k}{A_{k+1}} F(x^k) + \frac{a_{k+1}}{A_{k+1}} F^* + \frac{d - \tau}{2\tau} L R^2 \left(\frac{a_{k+1}}{A_{k+1}} \right)^2 + \left(\frac{d}{\tau} \right)^2 \frac{M R^3}{3} \left(\frac{a_{k+1}}{A_{k+1}} \right)^3.
\end{aligned}$$

Therefore, for the residual $\delta_k \stackrel{\text{def}}{=} \mathbb{E} [F(x^k)] - F^*$ we have the following bound

$$A_{k+1} \delta_{k+1} \leq A_k \delta_k + \frac{d - \tau}{2\tau} L R^2 \frac{a_{k+1}^2}{A_{k+1}} + \left(\frac{d}{\tau} \right)^2 \frac{M R^3}{3} \frac{a_{k+1}^3}{A_{k+1}^2}, \quad k \geq 0.$$

Summing up these inequalities for different k , we obtain

$$A_k \delta_k \leq A_0 \delta_0 + \frac{d - \tau}{2\tau} L R^2 \sum_{i=1}^k \frac{a_i^2}{A_i} + \left(\frac{d}{\tau} \right)^2 \frac{M R^3}{3} \sum_{i=1}^k \frac{a_i^3}{A_i^2}, \quad k \geq 1.$$

To finish the proof it remains to notice that

$$\sum_{i=1}^k \frac{a_i^2}{A_i} \stackrel{(I.3)}{\leq} \sum_{i=1}^k \frac{i^4}{A_0 + \frac{1}{3}i^3} \leq 3 \sum_{i=1}^k i \leq 3k^2,$$

and

$$\sum_{i=1}^k \frac{a_i^3}{A_i^2} \stackrel{(I.3)}{\leq} \sum_{i=1}^k \frac{i^6}{(A_0 + \frac{1}{3}i^3)^2} \leq 9k.$$

□

I.2.4 Proof of Theorem 9.6.10

Given that Assumption 9.6.9 (strong convexity) is satisfied, the following inequality holds

$$\frac{\mu}{2} \|x - x^*\|^2 \leq F(x) - F^*, \quad \forall x \in \mathbb{R}^d,$$

and thus we have a bound for the radius of level sets (9.9):

$$R^2 \leq \frac{2}{\mu}(F(x^0) - F^*).$$

Combining the above with (9.10) we obtain the following convergence estimate for $k \geq 1$:

$$\mathbb{E}[F(x^k) - F^*] \leq \left(\frac{d - \tau}{\tau} \cdot \frac{18L}{\mu k} + \left(\frac{d}{\tau}\right)^2 \cdot \frac{18MR}{\mu k^2} + \frac{1}{1 + \frac{1}{4}\left(\frac{\tau}{d}k\right)^3} \right) \cdot (F(x^0) - F^*).$$

Therefore, we get the linear decrease of the expected residual

$$\mathbb{E}[F(x^k) - F^*] \leq \frac{1}{2}(F(x^0) - F^*),$$

as soon as the following three bounds for k are all reached:

1. $\frac{d - \tau}{\tau} \cdot \frac{18L}{\mu k} \leq \frac{1}{6} \Leftrightarrow k \geq 108 \frac{d - \tau}{\tau} \cdot \frac{L}{\mu}.$
2. $\left(\frac{d}{\tau}\right)^2 \cdot \frac{18MR}{\mu k^2} \leq \frac{1}{6} \Leftrightarrow k \geq \frac{d}{\tau} \sqrt{108 \frac{MR}{\mu}}.$
3. $\frac{1}{1 + \frac{1}{4}\left(\frac{\tau}{d}k\right)^3} \leq \frac{1}{6} \Leftrightarrow k \geq \frac{d}{\tau} 20^{1/3}.$

□

I.3 Proofs for Section 9.7

I.3.1 Several technical lemmas

It will be convenient to denote the Newton decrement as follows:

$$\lambda_f(x) \stackrel{\text{def}}{=} \left(\nabla f(x)^\top (\nabla^2 f(x))^{-1} \nabla f(x) \right)^{\frac{1}{2}} \quad (1.4)$$

and a sublevel set of x^0 as χ^0 ; i.e. $\chi^0 \stackrel{\text{def}}{=} \{x; f(x) \leq f(x^0)\}.$

Lemma I.3.1. (Local bounds) Suppose that x^0 is such that

$$f(x^0) - f(x^*) \leq \varrho^4 \frac{2(\min_{x \in \chi^0} \lambda_{\min} \nabla_{\mathbf{S}}^2 f(x))^4}{LM_{\mathbf{S}}^2 \|S\|^2}$$

for some $\varrho > 0$. Then, we have

$$\sqrt{\frac{M_{\mathbf{S}}}{2}} \|\mathbf{S}^\top \nabla f(x^k)\|^{\frac{1}{2}} I \preceq \varrho \nabla_{\mathbf{S}}^2 f(x^k). \quad (1.5)$$

Suppose further that $f(x^0) - f(x^*) \leq \varphi^2 \frac{\mu(\lambda_{\min} \nabla_{\mathbf{S}}^2 f(x^*))^2}{2M_{\mathbf{S}}^2}$ for some $\varphi > 0$. Then we have

$$(1 + \varphi)^{-1} \nabla_{\mathbf{S}}^2 f(x^*) \preceq \nabla_{\mathbf{S}}^2 f(x^k) \preceq (1 + \varphi) \nabla_{\mathbf{S}}^2 f(x^*). \quad (1.6)$$

Lastly, if $f(x^0) - f(x^*) \leq \omega^{-1} \left(\frac{2\mu^{\frac{3}{2}}}{(1+\gamma^{-1})M} \right)$ where $\omega(y) \stackrel{\text{def}}{=} y - \log(1+y)$ and $\gamma > 0$, we have

$$f(x^k) - f(x^*) \leq \frac{1}{2}(1+\gamma)\lambda_f(x^k)^2. \quad (1.7)$$

Proof. For the sake of simplicity, let $x = x^k$ and $\mathbf{S} = \mathbf{S}^k$ throughout this proof. For the first part, we have

$$\begin{aligned} \sqrt{\frac{M_{\mathbf{S}}}{2}} \|\mathbf{S}^\top \nabla f(x)\|^{\frac{1}{2}} \mathbf{I}^{\tau(\mathbf{S})} &\leq \sqrt{\frac{M_{\mathbf{S}}}{2}} \|\mathbf{S}\|^{\frac{1}{2}} \|\nabla f(x)\|^{\frac{1}{2}} \mathbf{I}^{\tau(\mathbf{S})} \\ &\leq \sqrt{\frac{M_{\mathbf{S}}}{2}} \|\mathbf{S}\|^{\frac{1}{2}} 2^{\frac{1}{4}} L^{\frac{1}{4}} (f(x^0) - f(x^*))^{\frac{1}{4}} \mathbf{I}^{\tau(\mathbf{S})} \\ &\leq \varrho \min_{x \in \chi^0} \lambda_{\min} \nabla_{\mathbf{S}}^2 f(x) \mathbf{I}^{\tau(\mathbf{S})} \\ &\leq \varrho \nabla_{\mathbf{S}}^2 f(x). \end{aligned}$$

For the second part, we have

$$\begin{aligned} \nabla_{\mathbf{S}}^2 f(x^k) - \nabla_{\mathbf{S}}^2 f(x^*) &\preceq M_{\mathbf{S}} \|x^k - x^*\| \mathbf{I}^{\tau(\mathbf{S})} \\ &\preceq M_{\mathbf{S}} \sqrt{\frac{2(f(x^k) - f(x^*))}{\mu}} \mathbf{I}^{\tau(\mathbf{S})} \\ &\preceq M_{\mathbf{S}} \sqrt{\frac{2(f(x^0) - f(x^*))}{\mu}} \mathbf{I}^{\tau(\mathbf{S})} \\ &\preceq \varphi \nabla_{\mathbf{S}}^2 f(x^*). \end{aligned}$$

Therefore, we can conclude that $\nabla_{\mathbf{S}}^2 f(x) \preceq (1+\varphi) \nabla_{\mathbf{S}}^2 f(x^*)$. Analogously we can show $\nabla_{\mathbf{S}}^2 f(x^*) \preceq (1+\varphi) \nabla_{\mathbf{S}}^2 f(x)$ and thus (1.6) follows.

Lastly, if $f(x^0) - f(x^*) \leq \omega \left(\frac{2\mu^{\frac{3}{2}}}{(1+\gamma^{-1})M} \right)$, then due to [154] we have

$$\omega(\lambda_f(x^k)) \leq f(x^k) - f(x^*) \leq f(x^0) - f(x^*) \leq \omega \left(\frac{2\mu^{\frac{3}{2}}}{(1+\gamma^{-1})M} \right)$$

and thus $\lambda_f(x^k) \leq \frac{2\mu^{\frac{3}{2}}}{(1+\gamma^{-1})M}$. Now (1.7) follows from Lemma 1.3.2 and Lemma 1.3.3. \square

Lemma 1.3.2. Function f is $\frac{M}{\mu^{\frac{3}{2}}}$ self-concordant.

Proof.

$$\frac{M}{\mu^{\frac{3}{2}}} \|u\|_{\nabla^2 f(x)}^3 \geq M \|u\|^3 \geq \nabla^3 f(x)[u, u, u]$$

\square

Lemma I.3.3. Consider any $\gamma \in \mathbb{R}^+$ and suppose that f is ς self-concordant. Then if $\lambda_f(x) < \frac{2}{(1+\gamma^{-1})\varsigma}$ we have

$$f(x) - f(x^*) \leq \frac{1}{2} (1 + \gamma) \lambda_f(x)^2 \quad (1.8)$$

Proof. Define $\omega_*(z) \stackrel{\text{def}}{=} -z - \ln(1 - z)$. Note first that, $h(x) \stackrel{\text{def}}{=} \frac{\varsigma^2}{4} f(x)$ is 2 self concordant [154]. As a consequence, if $\lambda_h(x) < 1$ we have [154]

$$h(x) - h(x^*) \leq \omega_*(\lambda_h(x)).$$

If further $\lambda_h(x) \leq \frac{1}{1+\gamma^{-1}}$ due to Lemma I.3.4, we get

$$\omega_*(\lambda_h(x)) \leq (1 + \gamma) \frac{\lambda_h(x)^2}{2}.$$

As $\lambda_h(x) = \frac{\varsigma}{2} \lambda_f(x)$, we get (1.8). □

Lemma I.3.4. Let $c \in \mathbb{R}^+$ and $0 \leq y \leq \frac{1}{1+c}$. Then we have $\omega_*(y) \leq (1 + \frac{1}{c}) \frac{y^2}{2}$.

Proof. Clearly $\omega_*(y) = \sum_{i=2}^{\infty} \frac{y^i}{i}$ and thus function $(1 + \frac{1}{c}) \frac{y^2}{2} - \omega_*(y)$ is non-increasing for $y \geq 0$. Therefore, it suffices to check verify $(1 + \frac{1}{c}) \frac{1}{2(1+c)^2} - \omega_*(\frac{1}{1+c}) \geq 0$, which is easy task for Mathematica, see Figure I.1.

```
In[67]:= FindInstance[ (1 + 1/x) / (2 * (1 + x)^2) + 1 / (1 + x) + Log[ (1 - 1 / (1 + x)) ] < 0 && x > 0,
{ x } ]
Out[67]= {}
```

Figure I.1: Proof of $(1 + \frac{1}{c}) \frac{1}{2(1+c)^2} - \omega_*(\frac{1}{1+c}) \geq 0$ for all $c > 0$. □

I.3.2 Proof of Lemma 9.7.1

Note that the update rule of SSCN yields immediately (using first-order optimality conditions)

$$-\mathbf{S}^\top \nabla f(x) = \left(\nabla_{\mathbf{S}}^2 f(x) + \frac{1}{2} M_{\mathbf{S}} \|x^+ - x\| \mathbf{I}^{\tau(\mathbf{S})} \right) (x^+ - x) \quad (1.9)$$

and therefore

$$\begin{aligned}
\|\mathbf{S}^\top \nabla f(x)\|^{\frac{1}{2}} &= \left((x^+ - x)^\top \left(\nabla_{\mathbf{S}}^2 f(x) + \frac{1}{2} M_{\mathbf{S}} \|x^+ - x\| I \right) (x^+ - x) \right)^{\frac{1}{4}} \\
&\geq \left((x^+ - x)^\top \left(\frac{1}{2} M_{\mathbf{S}} \|x^+ - x\| I \right) (x^+ - x) \right)^{\frac{1}{4}} \\
&= \sqrt{\frac{M_{\mathbf{S}}}{2}} \|x^+ - x\|.
\end{aligned} \tag{I.10}$$

Furthermore, taking dot product of (I.9) with $(x^+ - x)$ yields

$$\langle \mathbf{S}^\top \nabla f(x), x^+ - x \rangle + \langle \nabla_{\mathbf{S}}^2 f(x) (x^+ - x), x^+ - x \rangle + \frac{1}{2} M_{\mathbf{S}} \|x^+ - x\|^3 = 0$$

and thus

$$\begin{aligned}
f(x) - f(x^+) &\stackrel{(9.5)}{\geq} \langle \mathbf{S}^\top \nabla f(x), x - x^+ \rangle - \frac{1}{2} \langle \nabla_{\mathbf{S}}^2 f(x) (x^+ - x), x^+ - x \rangle - \frac{M_{\mathbf{S}}}{6} \|x^+ - x\|^3 \\
&= \frac{1}{2} \langle \nabla_{\mathbf{S}}^2 f(x) (x^+ - x), x^+ - x \rangle + \frac{M_{\mathbf{S}}}{3} \|x^+ - x\|^3 \\
&\stackrel{(*)}{\geq} \frac{1}{2} (x^+ - x)^\top \left(\nabla_{\mathbf{S}}^2 f(x) + \frac{1}{2} M_{\mathbf{S}} \|x^+ - x\| I \right) (x^+ - x) \\
&\stackrel{(I.9)}{=} \frac{1}{2} \nabla f(x)^\top \mathbf{S} \left(\nabla_{\mathbf{S}}^2 f(x) + \frac{1}{2} M_{\mathbf{S}} \|x^+ - x\| I \right)^{-1} \mathbf{S}^\top \nabla f(x) \\
&\stackrel{(I.10)}{\geq} \frac{1}{2} \nabla f(x)^\top \mathbf{S} \left(\nabla_{\mathbf{S}}^2 f(x) + \sqrt{\frac{M_{\mathbf{S}}}{2}} \|\mathbf{S}^\top \nabla f(x)\|^{\frac{1}{2}} I \right)^{-1} \mathbf{S}^\top \nabla f(x).
\end{aligned}$$

Above, in inequality $(*)$ we have used the fact that matrix $(\nabla_{\mathbf{S}}^2 f(x) + \frac{1}{2} M_{\mathbf{S}} \|x^+ - x\| \mathbf{I}^{\tau(\mathbf{S})})$ is invertible since f is strongly convex and thus $\nabla_{\mathbf{S}}^2 f(x) \succ 0$.

I.3.3 Proof of Theorem 9.7.2

First, suppose that $f(x^0) - f(x^*) \leq \varrho^4 \frac{2(\min_{x \in \mathcal{X}^0} \lambda_{\min} \nabla_{\mathbf{S}}^2 f(x))^4}{LM_{\mathbf{S}}^2 \|\mathbf{S}\|^2}$ for some $\varrho > 0$. Using the fact that $\nabla_{\mathbf{S}}^2 f(x)$ is invertible (\mathbf{S} has full column rank and $\nabla^2 f(x) \succ 0$) we have

$$\begin{aligned}
\mathbb{E} \left[\frac{1}{2} \|\mathbf{S}^\top \nabla f(x^k)\|_{(\mathbf{H}(x^k))^{-1}}^2 \right] &\stackrel{(I.5)}{\geq} \mathbb{E} \left[\frac{1}{2} \nabla f(x)^\top \mathbf{S} ((1 + \varrho) \nabla_{\mathbf{S}}^2 f(x))^{-1} \mathbf{S}^\top \nabla f(x) \right] \\
&= \frac{1}{2(1 + \varrho)} \nabla f(x)^\top \mathbb{E} \left[\mathbf{S} (\nabla_{\mathbf{S}}^2 f(x))^{-1} \mathbf{S}^\top \right] \nabla f(x).
\end{aligned} \tag{I.11}$$

If further $f(x^0) - f(x^*) \leq \varphi^2 \frac{\mu(\lambda_{\min} \nabla_{\mathbf{S}}^2 f(x^*))^2}{2M_{\mathbf{S}}^2}$ for some $\varphi > 0$ we get

$$\begin{aligned}
\mathbb{E} \left[\frac{1}{2} \|\mathbf{S}^\top \nabla f(x^k)\|_{(\mathbf{H}(x^k))^{-1}}^2 \right] &\stackrel{(1.11)}{\geq} \frac{\nabla f(x)^\top \mathbb{E} \left[\mathbf{S} (\nabla_{\mathbf{S}}^2 f(x))^{-1} \mathbf{S}^\top \right] \nabla f(x)}{2(1+\varrho)} \\
&\stackrel{(1.6)}{\geq} \frac{\nabla f(x)^\top \mathbb{E} \left[\mathbf{S} (\nabla_{\mathbf{S}}^2 f(x^*))^{-1} \mathbf{S}^\top \right] \nabla f(x)}{2(1+\varrho)(1+\varphi)} \\
&\stackrel{(9.13)}{\geq} \frac{\nabla f(x)^\top \left(\zeta (\nabla^2 f(x^*))^{-1} \right) \nabla f(x)}{2(1+\varrho)(1+\varphi)} \\
&\stackrel{(1.6)}{\geq} \frac{\zeta \lambda_f(x)^2}{2(1+\varrho)(1+\varphi)^2} \tag{1.12}
\end{aligned}$$

Lastly, if $f(x^0) - f(x^*) \leq \omega^{-1} \left(\frac{2\mu^{\frac{3}{2}}}{(1+\gamma^{-1})M} \right)$ where $\omega(y) \stackrel{\text{def}}{=} y - \log(1+y)$ and $\gamma > 0$, we get

$$\begin{aligned}
\mathbb{E} \left[\frac{1}{2} \|\mathbf{S}^\top \nabla f(x^k)\|_{(\mathbf{H}(x^k))^{-1}}^2 \right] &\stackrel{(1.12)}{\geq} \frac{\zeta \lambda_f(x)^2}{2(1+\varrho)(1+\varphi)^2} \\
&\stackrel{(1.7)}{\geq} \frac{\zeta(f(x) - f(x^*))}{(1+\varrho)(1+\varphi)^2(1+\gamma)}
\end{aligned}$$

and thus (9.14) follows. In particular for any $\varrho, \varphi, \gamma > 0$, we can choose

$$\delta = \min \left\{ \varrho^4 \frac{2 (\min_{x \in \chi^0} \lambda_{\min} \nabla_S^2 f(x))^4}{LM_S^2}, \varphi^2 \frac{\mu (\lambda_{\min} \nabla_S^2 f(x^*))^2}{2M_S^2}, \omega^{-1} \left(\frac{2\mu^{\frac{3}{2}}}{(1+\gamma^{-1})M} \right) \right\}$$

and

$$\varepsilon = 1 - \frac{1}{(1+\varrho)(1+\varphi)^2(1+\gamma)}.$$

□

Appendix J

Appendix for Chapter 10

J.1 Proofs for Section 10.3

J.1.1 Proof of Lemma 10.3.2

First note that \mathbf{Z} is a self-adjoint positive operator and thus so is $\mathbb{E}[\mathbf{Z}]$. Consequently.

$$\begin{aligned}
 \theta & \stackrel{(10.13)}{=} \inf_{x \in \text{Range}(\mathcal{A}^*)} \frac{\langle \mathbb{E}[\mathbf{Z}] x, x \rangle}{\langle x, x \rangle} \\
 & \stackrel{(10.12)}{=} \inf_{x \in \text{Range}(\mathbb{E}[\mathbf{Z}])} \frac{\langle \mathbb{E}[\mathbf{Z}] x, x \rangle}{\langle x, x \rangle} \\
 & \stackrel{\text{Lemma J.8.8 item 2}}{=} \inf_{x \in \mathcal{X}} \frac{\langle \mathbb{E}[\mathbf{Z}] \mathbb{E}[\mathbf{Z}]^\dagger x, \mathbb{E}[\mathbf{Z}]^\dagger x \rangle}{\langle \mathbb{E}[\mathbf{Z}]^\dagger x, \mathbb{E}[\mathbf{Z}]^\dagger x \rangle} \\
 & \stackrel{\text{Lemma J.8.8 item 1}}{=} \inf_{x \in \mathcal{X}} \frac{\langle \mathbb{E}[\mathbf{Z}]^\dagger x, x \rangle}{\langle \mathbb{E}[\mathbf{Z}]^\dagger x, \mathbb{E}[\mathbf{Z}]^\dagger x \rangle} \\
 & \stackrel{\text{Lemma J.8.4}}{=} \inf_{z \in \text{Range}((\mathbb{E}[\mathbf{Z}]^\dagger)^{1/2})} \frac{\langle z, z \rangle}{\langle \mathbb{E}[\mathbf{Z}]^\dagger z, z \rangle} \quad (\text{set } z = (\mathbb{E}[\mathbf{Z}]^\dagger)^{1/2} x) \\
 & \stackrel{(J.48)}{=} \frac{1}{\left\| \mathbb{E}[\mathbf{Z}]^\dagger \right\|}. \tag{J.1}
 \end{aligned}$$

For the bounds (10.14) we have that

$$\begin{aligned}
 \nu & \stackrel{(10.13)}{=} \sup_{x \in \text{Range}(\mathcal{A}^*)} \frac{\mathbb{E} \left[\langle \mathbb{E}[\mathbf{Z}]^\dagger \mathbf{Z} x, \mathbf{Z} x \rangle \right]}{\langle \mathbb{E}[\mathbf{Z}] x, x \rangle} \\
 & \leq \sup_{x \in \text{Range}(\mathcal{A}^*)} \frac{\left\| \mathbb{E}[\mathbf{Z}]^\dagger \right\| \mathbb{E}[\|\mathbf{Z} x\|_2^2]}{\langle \mathbb{E}[\mathbf{Z}] x, x \rangle} \\
 & = \left\| \mathbb{E}[\mathbf{Z}]^\dagger \right\| \\
 & \stackrel{(J.1)}{\leq} \frac{1}{\theta}.
 \end{aligned}$$

To bound ν from below we use that $\mathbb{E}[\mathbf{Z}]^\dagger$ is self adjoint together with that the map

$\mathbf{X} \mapsto \langle \mathbf{X} \mathbb{E}[\mathbf{Z}]^\dagger \mathbf{X} x, x \rangle$ is convex over the space of self-adjoint operators $\mathbf{X} \in L(\mathcal{X})$ and for a fixed $x \in \mathcal{X}$. Consequently by Jensen's inequality

$$\mathbb{E} \left[\langle \mathbf{Z} \mathbb{E}[\mathbf{Z}]^\dagger \mathbf{Z} x, x \rangle \right] \geq \left\langle \mathbb{E}[\mathbf{Z}] \mathbb{E}[\mathbf{Z}]^\dagger \mathbb{E}[\mathbf{Z}] x, x \right\rangle \stackrel{\text{Lemma J.8.8 item 1}}{=} \langle \mathbb{E}[\mathbf{Z}] x, x \rangle. \quad (\text{J.2})$$

Finally

$$\nu \stackrel{(\text{J.2})}{\geq} \sup_{x \in \text{Range}(\mathcal{A}^*)} \frac{\langle \mathbb{E}[\mathbf{Z}] x, x \rangle}{\langle \mathbb{E}[\mathbf{Z}] x, x \rangle} = 1.$$

Lastly, to show (10.15) we have

$$\begin{aligned} \text{Rank}(\mathcal{A}^*) &\stackrel{(10.12)}{=} \text{Rank}(\mathbb{E}[\mathbf{Z}]) \\ &\stackrel{\text{Lemma J.8.3} + \text{Lemma J.8.8 (5)}}{=} \text{Tr} \left(\mathbb{E}[\mathbf{Z}] \mathbb{E}[\mathbf{Z}]^\dagger \right) = \mathbb{E} \left[\text{Tr} \left(\mathbf{Z} \mathbb{E}[\mathbf{Z}]^\dagger \right) \right] \\ &= \mathbb{E} \left[\text{Tr} \left(\mathbf{Z} \mathbb{E}[\mathbf{Z}]^\dagger \mathbf{Z} \right) \right] \\ &\leq \nu \mathbb{E}[\text{Tr}(\mathbf{Z})] \stackrel{\text{Lemma J.8.3}}{=} \nu \mathbb{E}[\text{Rank}(\mathbf{Z})], \end{aligned}$$

where we used that $\langle \mathbb{E}[\mathbf{Z} \mathbb{E}[\mathbf{Z}]^\dagger \mathbf{Z}] u, u \rangle \leq \nu \langle \mathbb{E}[\mathbf{Z}] u, u \rangle$ for every $u \in \text{Range}(\mathbb{E}[\mathbf{Z}]) = \text{Range}(\mathcal{A}^*) = \mathcal{X}$. \square

Proof that $\mathbf{X} \mapsto \langle \mathbf{X} \mathbb{E}[\mathbf{Z}]^\dagger \mathbf{X} x, x \rangle = \|\mathbf{X} x\|_{\mathbb{E}[\mathbf{Z}]^\dagger}^2$ is convex: Let $\mathbf{G} = \mathbb{E}[\mathbf{Z}]^\dagger$ then

$$\begin{aligned} \|(\lambda \mathbf{X} + (1 - \lambda) \mathbf{Y}) x\|_{\mathbf{G}}^2 &= \lambda^2 \|\mathbf{X} x\|_{\mathbf{G}}^2 + (1 - \lambda)^2 \|\mathbf{Y} x\|_{\mathbf{G}}^2 + 2\lambda(1 - \lambda) \langle x \mathbf{X} \mathbf{G} \mathbf{Y} x, x \rangle \\ &= -\lambda(1 - \lambda) \|(\mathbf{X} - \mathbf{Y}) x\|_{\mathbf{G}}^2 \\ &\quad + \lambda \|\mathbf{X} x\|_{\mathbf{G}}^2 + (1 - \lambda) \|\mathbf{Y} x\|_{\mathbf{G}}^2 \\ &\leq \lambda \|\mathbf{X} x\|_{\mathbf{G}}^2 + (1 - \lambda) \|\mathbf{Y} x\|_{\mathbf{G}}^2. \end{aligned} \quad \square$$

J.1.2 Technical lemmas to prove Theorem 10.3.3

Lemma J.1.1. *For all $k \geq 0$, the vectors $y_k - x_*$, $x_k - x_*$ and $v_k - x_*$ belong to $\text{Range}(\mathcal{A}^*)$.*

Proof. Note that $x_0 = y_0 = x_0$ and in view of (10.8) we have $x_* \in x_0 + \text{Range}(\mathcal{A}^*)$. So $y_0 - x_* \in \text{Range}(\mathcal{A}^*)$, $v_0 - x_* \in \text{Range}(\mathcal{A}^*)$ and $x_0 - x_* \in \text{Range}(\mathcal{A}^*)$. Assume by induction that $y_k - x_* \in \text{Range}(\mathcal{A}^*)$, $v_k - x_* \in \text{Range}(\mathcal{A}^*)$ and $x_k - x_* \in \text{Range}(\mathcal{A}^*)$. Since $g_k \in \text{Range}(\mathcal{A}^*)$ and $x_{k+1} = y_k - g_k$ we have

$$x_{k+1} - x_* = (y_k - x_*) - g_k \in \text{Range}(\mathcal{A}^*).$$

Moreover,

$$v_{k+1} - x_* = \beta(v_k - x_*) + (1 - \beta)(y_k - x_*) - \gamma g_k \in \text{Range}(\mathcal{A}^*).$$

Finally

$$y_{k+1} - x_* = \eta v_{k+1} + (1 - \eta)x_{k+1} - x_* = \eta(v_{k+1} - x_*) + (1 - \eta)(x_{k+1} - x_*) \in \mathbf{Range}(\mathcal{A}^*).$$

□

Lemma J.1.2.

$$\mathbb{E} \left[\|\mathcal{Z}_k(y_k - x_*)\|_{\mathbb{E}[\mathcal{Z}]^\dagger}^2 \mid y_k \right] \leq \nu \|y_k - x_*\|_{\mathbb{E}[\mathcal{Z}]}^2 \quad (\text{J.3})$$

Proof. Since $y_k - x_* \in \mathbf{Range}(\mathcal{A}^*)$ we have that

$$\begin{aligned} \mathbb{E} \left[\|\mathcal{Z}_k(y_k - x_*)\|_{\mathbb{E}[\mathcal{Z}]^\dagger}^2 \mid y_k \right] &= \left\langle \mathbb{E} \left[\mathcal{Z}_k \mathbb{E}[\mathcal{Z}]^\dagger \mathcal{Z}_k \right] (y_k - x_*), (y_k - x_*) \right\rangle \\ &\stackrel{(10.13)}{\leq} \nu \langle \mathbb{E}[\mathcal{Z}] (y_k - x_*), (y_k - x_*) \rangle \\ &= \nu \|y_k - x_*\|_{\mathbb{E}[\mathcal{Z}]}^2. \end{aligned}$$

□

Lemma J.1.3.

$$\|y_k - x_*\|_{\mathbb{E}[\mathcal{Z}]}^2 = \|y_k - x_*\|^2 - \mathbb{E}[\|x_{k+1} - x_*\|^2 \mid y_k] \quad (\text{J.4})$$

Proof.

$$\begin{aligned} \mathbb{E}[\|x_{k+1} - x_*\|^2 \mid y_k] &= \mathbb{E}[\|(\mathcal{I} - \mathcal{Z}_k)(y_k - x_*)\|^2 \mid y_k] \\ &= \langle (\mathcal{I} - \mathbb{E}[\mathcal{Z}]) (y_k - x_*), y_k - x_* \rangle \\ &= \|y_k - x_*\|^2 - \|y_k - x_*\|_{\mathbb{E}[\mathcal{Z}]}^2. \end{aligned}$$

□

J.1.3 Proof of Theorem 10.3.3

Let $r_k \stackrel{\text{def}}{=} \|v_k - x_*\|_{\mathbb{E}[\mathcal{Z}]^\dagger}^2$. It follows that

$$\begin{aligned} r_{k+1}^2 &= \|v_{k+1} - x_*\|_{\mathbb{E}[\mathcal{Z}]^\dagger}^2 \\ &= \|\beta v_k + (1 - \beta)y_k - x_* - \gamma \mathcal{Z}_k(y_k - x_*)\|_{\mathbb{E}[\mathcal{Z}]^\dagger}^2 \\ &= \underbrace{\|\beta v_k + (1 - \beta)y_k - x_*\|_{\mathbb{E}[\mathcal{Z}]^\dagger}^2}_I + \underbrace{\gamma^2 \|\mathcal{Z}_k(y_k - x_*)\|_{\mathbb{E}[\mathcal{Z}]^\dagger}^2}_{II} \\ &\quad - 2\gamma \underbrace{\left\langle \beta(v_k - x_*) + (1 - \beta)(y_k - x_*), \mathbb{E}[\mathcal{Z}]^\dagger \mathcal{Z}_k(y_k - x_*) \right\rangle}_{III} \\ &= I + \gamma^2 II - 2\gamma III. \end{aligned} \quad (\text{J.5})$$

The first term can be upper bounded as follows

$$\begin{aligned}
I &= \|\beta(v_k - x_*) + (1 - \beta)(y_k - x_*)\|_{\mathbb{E}[\mathcal{Z}]^\dagger}^2 \\
&= \beta^2 \|v_k - x_*\|_{\mathbb{E}[\mathcal{Z}]^\dagger}^2 + (1 - \beta)^2 \|y_k - x_*\|_{\mathbb{E}[\mathcal{Z}]^\dagger}^2 + 2\beta(1 - \beta) \langle v_k - x_*, y_k - x_* \rangle_{\mathbb{E}[\mathcal{Z}]^\dagger} \\
&\stackrel{(J.7)}{=} \beta \|v_k - x_*\|_{\mathbb{E}[\mathcal{Z}]^\dagger}^2 + (1 - \beta) \|y_k - x_*\|_{\mathbb{E}[\mathcal{Z}]^\dagger}^2 - \beta(1 - \beta) \|v_k - y_k\|_{\mathbb{E}[\mathcal{Z}]^\dagger}^2 \\
&\leq \beta r_k^2 + (1 - \beta) \|y_k - x_*\|_{\mathbb{E}[\mathcal{Z}]^\dagger}^2,
\end{aligned} \tag{J.6}$$

where in the third equality we used a form of the parallelogram identity

$$2 \langle u, v \rangle = \|u\|^2 + \|v\|^2 - \|u - v\|^2, \tag{J.7}$$

with $u = v_k - x_*$ and $v = y_k - x_*$.

Taking expectation with to \mathcal{S}_k in the third term in (J.5) gives

$$\begin{aligned}
\mathbb{E}[III \mid y_k, v_k, x_k] &= \left\langle \beta v_k + (1 - \beta)y_k - x_*, \mathbb{E}[\mathcal{Z}]^\dagger \mathbb{E}[\mathcal{Z}] (y_k - x_*) \right\rangle \\
&= \langle \beta v_k + (1 - \beta)y_k - x_*, y_k - x_* \rangle \\
&= \left\langle \beta \left[\frac{1}{\eta} y_k - \frac{1 - \eta}{\eta} x_k \right] + (1 - \beta)y_k - x_*, y_k - x_* \right\rangle \\
&= \left\langle y_k - x_* + \beta \frac{1 - \eta}{\eta} (y_k - x_k), y_k - x_* \right\rangle \\
&= \|y_k - x_*\|^2 + \beta \frac{1 - \eta}{\eta} \langle y_k - x_k, y_k - x_* \rangle \\
&= \|y_k - x_*\|^2 \\
&\quad - \beta \frac{1 - \eta}{2\eta} (\|x_k - x_*\|^2 - \|y_k - x_k\|^2 - \|y_k - x_*\|^2)
\end{aligned} \tag{J.8}$$

where in the second equality (J.8) we used that $y_k - x_* \in \mathbf{Range}(\mathcal{A}^*) \stackrel{(10.12)}{=} \mathbf{Range}(\mathbb{E}[\mathcal{Z}])$ together with a defining property of pseudoinverse operators $\mathbb{E}[\mathcal{Z}]^\dagger \mathbb{E}[\mathcal{Z}] w = w$ for all $w \in \mathbf{Range}(\mathbb{E}[\mathcal{Z}])$. In the last equality (J.9) we used yet again the identity (J.7) with $u = y_k - x_k$ and $v = y_k - x_*$.

Plugging (J.6) and (J.9) into (J.5) and taking conditional expectation gives

$$\begin{aligned}
\mathbb{E} \left[r_{k+1}^2 \mid y_k, v_k, x_k \right] &= I + \gamma^2 \mathbb{E} [II \mid y_k] - 2\gamma \mathbb{E} [III \mid y_k, v_k, x_k] \\
&\stackrel{(J.6)+(J.9)+(J.3)}{=} \beta r_k^2 + (1 - \beta) \|y_k - x_*\|_{\mathbb{E}[Z]^\dagger}^2 + \gamma^2 \nu \|y_k - x_*\|_{\mathbb{E}[Z]}^2 \\
&\quad - 2\gamma \|y_k - x_*\|^2 \\
&\quad + \gamma \beta \frac{1 - \eta}{\eta} (\|x_k - x_*\|^2 - \|y_k - x_k\|^2 - \|y_k - x_*\|^2) \\
&\stackrel{(J.4)+(10.14)}{\leq} \beta r_k^2 + \frac{1 - \beta}{\theta} \|y_k - x_*\|^2 + \gamma^2 \nu \|y_k - x_*\|^2 \\
&\quad - \gamma^2 \nu \mathbb{E} [\|x_{k+1} - x_*\|^2 \mid y_k] - 2\gamma \|y_k - x_*\|^2 \\
&\quad + \left(\beta \frac{1 - \eta}{2\eta} (\|x_k - x_*\|^2 - \|y_k - x_*\|^2) \right). \quad (J.10)
\end{aligned}$$

Therefore we have that

$$\begin{aligned}
&\mathbb{E} \left[r_{k+1}^2 + \gamma^2 \nu \|x_{k+1} - x_*\|^2 \mid y_k, v_k, x_k \right] \quad (J.11) \\
&\leq \beta \left(r_k^2 + \underbrace{\gamma \frac{1 - \eta}{\eta}}_{P_1} \|x_k - x_*\|^2 \right) \\
&\quad + \left(\underbrace{\frac{1 - \beta}{\theta} - 2\gamma + \gamma^2 \nu - \beta \gamma \frac{1 - \eta}{\eta}}_{P_2} \|y_k - x_*\|^2 \right).
\end{aligned}$$

To establish a recurrence, we need to choose the free parameters γ, η and β so that $P_1 = \gamma^2 \nu$ and $P_2 = 0$. Furthermore we should try to set β as small as possible so as to have a fast rate of convergence. Choosing $\beta = 1 - \sqrt{\frac{\theta}{\nu}}$, $\gamma = \sqrt{\frac{1}{\theta \nu}}$, $\eta = \frac{1}{1 + \gamma \nu}$ gives $P_2 = 0$, $\gamma^2 \nu = 1/\theta$ and

$$\mathbb{E} \left[r_{k+1}^2 + \frac{1}{\theta} \|x_{k+1} - x_*\|^2 \mid y_k, v_k, x_k \right] \leq \left(1 - \sqrt{\frac{\theta}{\nu}} \right) \left(r_k^2 + \frac{1}{\theta} \|x_k - x_*\|^2 \right).$$

Taking expectation and using the tower rules gives the result. \square

J.1.4 Changing norm

Given an invertible positive self-adjoint $\mathcal{B} \in L(\mathcal{X})$, suppose we want to find the least norm solution of (10.7) under the norm defined by $\|x\|_{\mathcal{B}} \stackrel{\text{def}}{=} \sqrt{\langle \mathcal{B}x, x \rangle}$ as the metric in \mathcal{X} . That is, we want to solve

$$x^* \stackrel{\text{def}}{=} \arg \min_{x \in \mathcal{X}} \frac{1}{2} \|x - x_0\|_{\mathcal{B}}^2, \quad \text{subject to } \mathcal{A}x = b. \quad (J.12)$$

By changing variables $x = \mathcal{B}^{-1/2}z$ we have that the above is equivalent to solving

$$z^* \stackrel{\text{def}}{=} \arg \min_{z \in \mathcal{X}} \frac{1}{2} \|z - z_0\|^2, \quad \text{subject to } \mathcal{A}\mathcal{B}^{-1/2}z = b, \quad (\text{J.13})$$

with $x^* = \mathcal{B}^{-1/2}z^*$, and $\mathcal{B}^{1/2}$ is the unique symmetric square root of \mathcal{B} (see Lemma J.8.4). We can now apply Algorithm 22 to solve (J.13) where $\mathcal{A}\mathcal{B}^{-1/2}$ is the system matrix. Let x_k and v_k be the resulting iterates of applying Algorithm 22. To make explicit this change in the system matrix we define the matrix

$$\mathcal{Z}_\mathcal{B} \stackrel{\text{def}}{=} \mathcal{B}^{-1/2} \mathcal{A}^* \mathcal{S}_k^* (\mathcal{S}_k \mathcal{A} \mathcal{B}^{-1} \mathcal{A}^* \mathcal{S}_k^*)^\dagger \mathcal{S}_k \mathcal{A} \mathcal{B}^{-1/2},$$

and the constants

$$\theta_B \stackrel{\text{def}}{=} \inf_{x \in \text{Range}(\mathcal{B}^{-1/2} \mathcal{A}^*)} \frac{\langle \mathbb{E}[\mathcal{Z}_\mathcal{B}] x, x \rangle}{\langle x, x \rangle} \quad (\text{J.14})$$

and

$$\nu_B \stackrel{\text{def}}{=} \sup_{x \in \text{Range}(\mathcal{B}^{-1/2} \mathcal{A}^*)} \frac{\langle \mathbb{E}[\mathcal{Z}_\mathcal{B} \mathbb{E}[\mathcal{Z}_\mathcal{B}]^\dagger \mathcal{Z}_\mathcal{B}] x, x \rangle}{\langle \mathbb{E}[\mathcal{Z}_\mathcal{B}] x, x \rangle}. \quad (\text{J.15})$$

Theorem 10.3.3 then guarantees that

$$\mathbb{E} \left[\|v_{k+1} - z_*\|_{\mathbb{E}[\mathcal{Z}_\mathcal{B}]^\dagger}^2 + \frac{1}{\theta_B} \|x_{k+1} - z_*\|^2 \right] \leq \left(1 - \sqrt{\frac{\theta_B}{\nu_B}} \right) \mathbb{E} \left[\|v_k - z_*\|_{\mathbb{E}[\mathcal{Z}_\mathcal{B}]^\dagger}^2 + \frac{1}{\theta_B} \|x_k - z_*\|^2 \right].$$

Reversing our change of variables $\bar{x}_k = \mathcal{B}^{-1/2}x_k$ and $\bar{v}_k = \mathcal{B}^{-1/2}v_k$ in the above displayed equation gives

$$\begin{aligned} & \mathbb{E} \left[\|\bar{v}_{k+1} - x_*\|_{\mathcal{B}^{1/2} \mathbb{E}[\mathcal{Z}_\mathcal{B}]^\dagger \mathcal{B}^{1/2}}^2 + \frac{1}{\theta_B} \|\bar{x}_{k+1} - x_*\|_{\mathcal{B}}^2 \right] \\ & \leq \left(1 - \sqrt{\frac{\theta_B}{\nu_B}} \right) \mathbb{E} \left[\|\bar{v}_k - x_*\|_{\mathcal{B}^{1/2} \mathbb{E}[\mathcal{Z}_\mathcal{B}]^\dagger \mathcal{B}^{1/2}}^2 + \frac{1}{\theta_B} \|\bar{x}_k - x_*\|_{\mathcal{B}}^2 \right]. \quad (\text{J.16}) \end{aligned}$$

Thus we recover the same exact from the main theorem in [178], but in a much more general setting.

J.2 Proof of Corollary 10.3.4

Clearly, $\mathbf{Z} = \frac{1}{\mathbf{A}_{i,i}} \mathbf{A}^{\frac{1}{2}} \mathbf{S} \mathbf{S}^\top \mathbf{A}^{\frac{1}{2}}$, and hence $\mathbb{E}[\mathbf{Z}] = \frac{\mathbf{A}}{\text{Tr}(\mathbf{A})}$ and $\theta^P = \frac{\lambda_{\min}(\mathbf{A})}{\text{Tr}(\mathbf{A})}$. After simple algebraic manipulations we get

$$\mathbb{E} \left[\mathbb{E}[\mathbf{Z}]^{-\frac{1}{2}} \mathbf{Z} \mathbb{E}[\mathbf{Z}]^{-1} \mathbf{Z} \mathbb{E}[\mathbf{Z}]^{-\frac{1}{2}} \right] = \text{Tr}(\mathbf{A})^2 \mathbb{E} \left[\frac{1}{\mathbf{A}_{i,i}^2} \mathbf{S} \mathbf{S}^\top \mathbf{S} \mathbf{S}^\top \right] = \text{Tr}(\mathbf{A}) \text{Diag } \mathbf{A}_{i,i}^{-1},$$

and therefore $\nu^P = \lambda_{\max} \mathbb{E} \left[\mathbb{E}[\mathbf{Z}]^{-\frac{1}{2}} \mathbf{Z} \mathbb{E}[\mathbf{Z}]^{-1} \mathbf{Z} \mathbb{E}[\mathbf{Z}]^{-\frac{1}{2}} \right] = \frac{\text{Tr}(\mathbf{A})}{\min_i \mathbf{A}_{i,i}}.$

J.3 Adding a stepsize

In this section we enrich Algorithm 22 with several *additional* parameters and study their effect on convergence of the resulting method.

First, we consider an extension of Algorithm 22 to a variant which uses a *stepsize parameter* $0 < \omega < 2$. That is, instead of performing the update

$$x_{k+1} = y_k - g_k, \quad (\text{J.17})$$

we perform the update

$$x_{k+1} = y_k - \omega g_k. \quad (\text{J.18})$$

Parameters η, β, γ are adjusted accordingly. The resulting method enjoys the rate

$$\mathcal{O} \left(\left(1 - \sqrt{\frac{\nu}{\theta} \omega (2 - \omega)} \right)^k \right),$$

recovering the rate from Theorem 10.3.3 as a special case for $\omega = 1$. The formal statement follows.

Theorem J.3.1. *Let $0 < \omega < 2$ be an arbitrary stepsize and define*

$$\alpha \stackrel{\text{def}}{=} 2\omega - \omega^2 \geq 0. \quad (\text{J.19})$$

Consider a modification of Algorithm 22 where instead of (J.17) we perform the update (J.18). If we use the parameters

$$\eta = \frac{1}{1 + \gamma\nu} \quad \beta = 1 - \sqrt{\frac{\theta\alpha}{\nu}} \quad \gamma = \sqrt{\frac{\alpha}{\theta\nu}}, \quad (\text{J.20})$$

then the iterates $\{v_k, x_k\}_{k \geq 0}$ of Algorithm 22 satisfy

$$\mathbb{E} \left[\|v_k - x_*\|_{\mathbb{E}[\mathcal{Z}]^\dagger}^2 + \frac{1}{\theta} \|x_k - x_*\|^2 \right] \leq \left(1 - \sqrt{\frac{\theta\alpha}{\nu}} \right)^k \mathbb{E} \left[\|v_0 - x_*\|_{\mathbb{E}[\mathcal{Z}]^\dagger}^2 + \frac{1}{\theta} \|x_0 - x_*\|^2 \right].$$

Proof. See Appendix J.5. □

J.4 Allowing for different η

In this section we study how the choice of the key parameter η affects the convergence rate.

This parameter determines how much the sequence $y_k = \eta v_k + (1 - \eta)x_k$ resembles the sequence given by x_k or by v_k . For instance, when $\eta = 0$, $y_k \equiv x_k$, i.e., we recover the steps of the non-accelerated method, and thus one would expect to obtain the same convergence rate as the non-accelerated method. Similar considerations hold in the other

extreme, when $\eta \rightarrow 1$. We investigate this hypothesis, and especially discuss how β and γ must be chosen as a function of η to ensure convergence.

The following statement is a generalization of Theorem 10.3.3. For simplicity, we assume that the optional stepsize that was introduced in Theorem J.3.1 is set to one again, $\omega \equiv 1$.

Theorem J.4.1. *Let $0 < \eta < 1$ be fixed. Then the iterates $\{v_k, x_k\}_{k \geq 0}$ of Algorithm 22 with parameters*

$$\beta(s) = \frac{1 + s - s\sqrt{\frac{\nu + 4\theta s - 2\nu s + \nu s^2}{\nu s^2}}}{2s}, \quad \gamma(s) = \frac{1}{(1 - s\beta(s))\nu}. \quad (\text{J.21})$$

where $\tau \stackrel{\text{def}}{=} \frac{1-\eta}{\eta}$ and $s \stackrel{\text{def}}{=} \frac{\tau}{\beta\gamma}$, satisfy

$$\mathbb{E} \left[\|v_k - x_*\|_{\mathbb{E}[\mathcal{Z}]^\dagger}^2 + \gamma\tau \|x_k - x_*\|^2 \right] \leq \rho^k \mathbb{E} \left[\|v_0 - x_*\|_{\mathbb{E}[\mathcal{Z}]^\dagger}^2 + \gamma\tau \|x_0 - x_*\|^2 \right].$$

(or put differently):

$$\mathbb{E} \left[\|v_k - x_*\|_{\mathbb{E}[\mathcal{Z}]^\dagger}^2 + (1 - \eta)\gamma \|x_k - x_*\|^2 \right] \leq \rho^k \mathbb{E} \left[\|v_0 - x_*\|_{\mathbb{E}[\mathcal{Z}]^\dagger}^2 + (1 - \eta)\gamma \|x_0 - x_*\|^2 \right].$$

where $\rho = \max\{\beta(s), s\beta(s)\} \leq 1$.

We can now exemplify a few special parameter settings.

Example 16. For $\eta = 1$, i.e., if $s \rightarrow 0$, we get the rate $\rho = 1 - \frac{\theta}{\nu}$ with $\beta = 1 - \frac{\theta}{\nu}$, $\gamma = \frac{1}{\nu}$.

Example 17. For $\eta \rightarrow 0$, i.e., in the limit $s \rightarrow \infty$, we get the rate $\rho = 1 - \frac{\theta}{\nu}$.

Example 18. The rate ρ is minimized for $s = 1$, i.e., $\beta = 1 - \sqrt{\frac{\nu}{\theta}}$ and $\gamma = \sqrt{\frac{1}{\theta\nu}}$; recovering Theorem 10.3.3.

The best case, in terms of convergence rate for both non-unit stepsize and a variable parameter choice happened to be the default parameter setup. The non-optimal parameter choice was studied in order to have theoretical guarantees for a wider class of parameters, as in practice one might be forced to rely on sub-optimal / inexact parameter choices.

J.5 Proof of Theorem J.3.1

The proof follows by slight modifications of the proof of Theorem 10.3.3.

First we adapt Lemma J.1.3. As we have $x_{k+1} - x_* = (\mathcal{I} - \omega \mathcal{Z}_k)(y_k - x_*)$ the following statement follows by the same arguments as in the proof of Lemma J.1.3.

Lemma J.5.1 (Lemma J.1.3').

$$\alpha \|y_k - x_*\|_{\mathbb{E}[\mathcal{Z}]}^2 = \|y_k - x_*\|^2 - \mathbb{E} [\|x_{k+1} - x_*\|^2 \mid y_k] \quad (\text{J.22})$$

Proof.

$$\begin{aligned}
\mathbb{E} [\|x_{k+1} - x_*\|^2 \mid y_k] &= \mathbb{E} [\|(\mathcal{I} - \mathcal{Z}_k)(y_k - x_*)\|^2 \mid y_k] \\
&= \mathbb{E} [\langle (\mathcal{I} - \omega \mathcal{Z}_k)(y_k - x_*), (\mathcal{I} - \omega \mathcal{Z}_k)y_k - x_* \rangle] \\
&= \|y_k - x_*\|^2 - \alpha \|y_k - x_*\|_{\mathbb{E}[\mathcal{Z}]}^2.
\end{aligned}$$

□

We now follow the same steps as in proof of Theorem 10.3.3 in Section J.1.3. We observe, that the first time Lemma J.1.3 is applied is in equation (J.10). Using Lemma J.5.1 instead, gives

$$\mathbb{E} [r_{k+1}^2 \mid y_k, v_k, x_k] \quad (\text{J.23})$$

$$\begin{aligned}
&\leq \beta r_k^2 + \frac{1-\beta}{\theta} \|y_k - x_*\|^2 + \frac{\gamma^2 \nu}{\alpha} (\|y_k - x_*\|^2 - \mathbb{E} [\|x_{k+1} - x_*\|^2 \mid y_k]) \\
&\quad + 2\gamma \left(-\|y_k - x_*\|^2 + \beta \frac{1-\eta}{2\eta} (\|x_k - x_*\|^2 - \|y_k - x_*\|^2) \right). \quad (\text{J.24})
\end{aligned}$$

Therefore we have that

$$\begin{aligned}
&\mathbb{E} [r_{k+1}^2 + \gamma^2 \nu \|x_{k+1} - x_*\|^2 \mid y_k, v_k, x_k] \quad (\text{J.25}) \\
&\leq \beta \left(r_k^2 + \underbrace{\gamma \frac{1-\eta}{\eta}}_{P'_1} \|x_k - x_*\|^2 \right) \\
&\quad + \left(\underbrace{\frac{1-\beta}{\theta} - 2\gamma + \frac{\gamma^2 \nu}{\alpha} - \beta \gamma \frac{1-\eta}{\eta}}_{P'_2} \right) \|y_k - x_*\|^2.
\end{aligned}$$

Noting that $\frac{1-\eta}{\eta} = \gamma \nu$ and $\frac{\gamma^2 \nu}{\alpha} = \frac{\gamma(1-\eta)}{\alpha \eta} = \frac{1}{\theta}$, we observe $P'_2 = 0$ and deduce the statement of Theorem J.3.1.

J.6 Proof of Theorem J.4.1

It suffices to study equation (J.10). We observe that for convergence the big bracket, P_2 , should be negative,

$$(1-\beta) \frac{1}{\theta} + \gamma^2 \nu - 2\gamma - \gamma \beta \frac{1-\eta}{\eta} \leq 0 \quad (\text{J.26})$$

The convergence rate is then

$$\rho \stackrel{\text{def}}{=} \max \left\{ \beta, \frac{(1-\eta)\beta}{\eta\gamma\nu} \right\}. \quad (\text{J.27})$$

or in the notation of Theorem J.4.1, $\rho = \max\{\beta, s\beta\}$.

This means, that in order to obtain the best convergence rate, we should therefore choose parameters β and γ such that β is as small as possible. This observation is true regardless of the value of s (which itself depends on γ).

With the notation $\tau = s\gamma\beta$, we reformulate (J.26) to obtain

$$\frac{1}{\theta} + \gamma^2\nu - 2\gamma \leq \beta \left(\frac{1}{\theta} + s\gamma^2\nu \right) \quad (\text{J.28})$$

Thus we see, that β cannot be chosen smaller than

$$\beta^*(s, \gamma) = \frac{1 + \theta\gamma^2\nu - 2\theta\gamma}{1 + s\theta\gamma^2\nu} \quad (\text{J.29})$$

Minimizing this expression in γ gives

$$\beta^*(s) = \frac{1 + s - s\sqrt{\frac{\nu + 4\theta s - 2\nu s + \nu s^2}{\nu s^2}}}{2s} \quad (\text{J.30})$$

with $\gamma^*(s) = \frac{1}{(1 - s\beta^*(s))\nu}$.

We further observe that this parameter setting indeed guarantees convergence, i.e. $\rho \leq 1$. From (J.30) we observe ($\nu > 0$, $s \geq 0$, $\theta \geq 0$):

$$\beta^*(s) \leq \frac{1 + s - \sqrt{\frac{\nu - 2\nu s + \nu s^2}{\nu}}}{2s} = \frac{1 + s - (s - 1)}{2s} = \frac{1}{s} \quad (\text{J.31})$$

Hence $s\beta^*(s) \leq 1$. On the other hand, $(1 - s) \leq \sqrt{(1 - s)^2 + \frac{4\theta s}{\nu}}$ and hence $(1 + s) - \sqrt{(1 - s)^2 + \frac{4\theta s}{\nu}} \leq 2s$, which shows $\beta^*(s) \leq 1$.

J.7 Proofs and further comments on Section 10.4

J.7.1 Proof of Theorem 10.4.1

We perform a change of coordinates since it is easier to work with the standard Frobenius norm as opposed to the weighted Frobenius norm. Let $\hat{\mathbf{X}} = \mathbf{A}^{1/2}\mathbf{X}\mathbf{A}^{1/2}$ so that (10.18) and (10.20) become

$$\hat{\mathbf{X}}_* \stackrel{\text{def}}{=} \arg \min \left\| \hat{\mathbf{X}} \right\|_F^2 \quad \text{subject to} \quad \hat{\mathbf{X}} = \mathbf{I}, \quad \hat{\mathbf{X}} = \hat{\mathbf{X}}^\top, \quad (\text{J.32})$$

and

$$\hat{\mathbf{X}}_{k+1} = \mathbf{P} + (\mathbf{I} - \mathbf{P}) \hat{\mathbf{X}}_k (\mathbf{I} - \mathbf{P}), \quad (\text{J.33})$$

respectively, where $\mathbf{P} = \mathbf{A}^{1/2} \mathbf{S} (\mathbf{S}^\top \mathbf{A} \mathbf{S})^{-1} \mathbf{S}^\top \mathbf{A}^{1/2}$. The linear operator that encodes the constant in (10.19) is given by $\hat{\mathcal{A}}(\mathbf{X}) = (\mathbf{X}, \mathbf{X} - \mathbf{X}^\top)$ the adjoint of which is given by $\hat{\mathcal{A}}^*(\mathbf{Y}_1, \mathbf{Y}_2) = \mathbf{Y}_1 + \mathbf{Y}_2 - \mathbf{Y}_2^\top$. Since $\hat{\mathcal{A}}^*$ is clearly surjective, it follows that $\text{Range}(\hat{\mathcal{A}}^*) = \mathbb{R}^{d \times d}$.

Subtracting the identity matrix from both sides of (J.33) and using that \mathbf{P} is a projection matrix, we have that

$$\hat{\mathbf{X}}_{k+1} - \mathbf{I} = (\mathbf{I} - \mathbf{P}) (\hat{\mathbf{X}}_k - \mathbf{I}) (\mathbf{I} - \mathbf{P}). \quad (\text{J.34})$$

To determine the \mathbf{Z} operator (10.9), from (10.11) and (J.34) we know that

$$(\mathbf{I} - \mathbf{P}) (\hat{\mathbf{X}}_k - \mathbf{I}) (\mathbf{I} - \mathbf{P}) = (\mathbf{I} - \mathbf{Z}) (\hat{\mathbf{X}}_k - \mathbf{I}).$$

Thus for every matrix $\mathbf{X} \in \mathbb{R}^{d \times d}$ we have that

$$\mathbf{Z}(\mathbf{X}) = \mathbf{X} - (\mathbf{I} - \mathbf{P}) \mathbf{X} (\mathbf{I} - \mathbf{P}) = \mathbf{X} \mathbf{P} + \mathbf{P} \mathbf{X} (\mathbf{I} - \mathbf{P}). \quad (\text{J.35})$$

Denote column-wise vectorization of \mathbf{X} as x : $x \stackrel{\text{def}}{=} \text{Vec}(\mathbf{X})$. To calculate a useful lower bound on θ , note that

$$\begin{aligned} \text{Tr}(\mathbf{X}^\top \mathbf{Z}(\mathbf{X})) &= \text{Tr}(\mathbf{X}^\top \mathbf{X} \mathbf{P}) + \text{Tr}(\mathbf{X}^\top \mathbf{P} \mathbf{X} (\mathbf{I} - \mathbf{P})) \\ &= x^\top \text{Vec}(\mathbf{X} \mathbf{P}) + x^\top \text{Vec}(\mathbf{P} \mathbf{X} (\mathbf{I} - \mathbf{P})) \\ &= x^\top (\mathbf{P} \otimes \mathbf{I}) x + x^\top ((\mathbf{I} - \mathbf{P}) \otimes \mathbf{P}) x \\ &\stackrel{(10.23)}{=} x^\top \mathbf{Z}' x, \end{aligned} \quad (\text{J.36})$$

where we used that $\text{Tr}(\mathbf{A}^\top \mathbf{B}) = \text{Vec}(\mathbf{A})^\top \text{Vec}(\mathbf{B})$ and $\text{Vec}(\mathbf{A} \mathbf{X} \mathbf{B}) = (\mathbf{B}^\top \otimes \mathbf{A}) \text{Vec}(x)$ holds for any $\mathbf{A}, \mathbf{B}, \mathbf{X}$.

Consequently, θ is equal to

$$\theta \stackrel{(10.22)}{=} \inf_{\mathbf{X} \in \mathbb{R}^{n \times n}} \frac{\langle \mathbb{E}[\mathbf{Z}] \mathbf{X}, \mathbf{X} \rangle_F}{\|\mathbf{X}\|_F^2} \stackrel{(\text{J.36})}{=} \inf_{x \in \mathbb{R}^{n^2 \times n^2}} \frac{x^\top \mathbb{E}[\mathbf{Z}'] x}{x^\top x} = \lambda_{\min}(\mathbb{E}[\mathbf{Z}']).$$

Notice that we have $2\lambda_{\min}(\mathbb{E}[\mathbf{P}]) \geq \lambda_{\min}(\mathbb{E}[\mathbf{Z}']) \geq \lambda_{\min}(\mathbb{E}[\mathbf{P}])$ since $(\mathbf{P} \otimes \mathbf{I}) + (\mathbf{I} \otimes \mathbf{P}) \geq \mathbf{Z}' \geq (\mathbf{P} \otimes \mathbf{I})$.

In light of Algorithm 22, the iterates of the accelerated version of (J.33) are given by

$$\begin{aligned} \hat{\mathbf{Y}}_k &= \eta \hat{\mathbf{V}}_k + (1 - \eta) \hat{\mathbf{X}}_k \\ \hat{\mathbf{G}}_k &= \mathbf{Z}_k(\hat{\mathbf{Y}}_k - \mathbf{I}) \\ \hat{\mathbf{X}}_{k+1} &= \hat{\mathbf{Y}}_k - \hat{\mathbf{G}}_k \\ \hat{\mathbf{V}}_{k+1} &= \beta \hat{\mathbf{V}}_k + (1 - \beta) \hat{\mathbf{Y}}_k - \gamma \hat{\mathbf{G}}_k \end{aligned} \quad (\text{J.37})$$

where $\hat{\mathbf{Y}}_k, \hat{\mathbf{V}}_k, \hat{\mathbf{G}} \in \mathbb{R}^{n \times n}$. From Theorem 10.3.3 we have that $\hat{\mathbf{V}}_k$ and $\hat{\mathbf{X}}_k$ converge to

the identity matrix according to

$$\mathbb{E} \left[\left\| \hat{\mathbf{V}}_{k+1} - \mathbf{I} \right\|_{\mathbb{E}[\mathbf{Z}]^\dagger}^2 + \frac{1}{\theta} \left\| \hat{\mathbf{X}}_{k+1} - \mathbf{I} \right\|_F^2 \right] \leq \left(1 - \sqrt{\frac{\theta}{\nu}} \right) \mathbb{E} \left[\left\| \hat{\mathbf{V}}_k - \mathbf{I} \right\|_{\mathbb{E}[\mathbf{Z}]^\dagger}^2 + \frac{1}{\theta} \left\| \hat{\mathbf{X}}_k - \mathbf{I} \right\|_F^2 \right], \quad (\text{J.38})$$

where $\|\mathbf{X}\|_{\mathbb{E}[\mathbf{Z}]^\dagger}^2 = \left\langle \mathbb{E}[\mathbf{Z}]^\dagger \mathbf{X}, \mathbf{X} \right\rangle_F$. Changing coordinates back to $\hat{\mathbf{X}}_k = \mathbf{A}^{1/2} \mathbf{X}_k \mathbf{A}^{1/2}$ and defining $\mathbf{Y}_k \stackrel{\text{def}}{=} \mathbf{A}^{-1/2} \hat{\mathbf{Y}}_k \mathbf{A}^{-1/2}$, $\mathbf{V}_k \stackrel{\text{def}}{=} \mathbf{A}^{-1/2} \hat{\mathbf{V}}_k \mathbf{A}^{-1/2}$ and $\mathbf{G}_k \stackrel{\text{def}}{=} \mathbf{A}^{-1/2} \hat{\mathbf{G}}_k \mathbf{A}^{-1/2}$, we have that (J.38) gives (10.21). Furthermore, using the same coordinate change applied to the iterates (J.37) gives Algorithm 23.

J.7.2 Matrix inversion as linear system

Denote $x = \text{Vec}(\mathbf{X})$, i.e. x is d^2 dimensional vector such that $x_{(n(i-1)+1):ni} = \mathbf{X}_{:,i}$. Similarly, denote $e = \text{Vec}(\mathbf{I})$. System (10.6) can be thus rewritten as

$$(\mathbf{I} \otimes \mathbf{A})x = e. \quad (\text{J.39})$$

Notice that all linear sketches of the original system $\mathbf{AX} = \mathbf{I}$ can be written as

$$\mathbf{S}_0^\top (\mathbf{I} \otimes \mathbf{A})x = \mathbf{S}_0^\top e \quad (\text{J.40})$$

for a suitable $d^2 \times d^2$ matrix \mathbf{S}_0 , therefore the setting is fairly general.

Alternative proof of Theorem 10.4.1

Let us now, for a purpose of this proof, consider sketch matrix \mathbf{S}_0 to capture only sketching the original matrix system $\mathbf{AX} = \mathbf{I}$ by left multiplying by \mathbf{S} , i.e. $\mathbf{S}_0 = (\mathbf{I} \otimes \mathbf{S})$, as those are the considered sketches in the setting of Section 10.4.

As we have

$$\text{Tr}(\mathbf{BX}^\top \mathbf{BX}) = \text{Vec}(\mathbf{BXB})^\top x = x^\top (\mathbf{B} \otimes \mathbf{B})x,$$

weighted Frobenius norm of matrices is equivalent to a special weighted euclidean norm of vectors. Define also \mathbf{C} to be a matrix such that $\mathbf{C}x = 0$ if and only if $\mathbf{X} = \mathbf{X}^\top$. Therefore, (10.19) is equivalent to

$$x_{k+1} = \arg \min \|x - x_k\|_{\mathbf{A} \otimes \mathbf{A}}^2 \quad \text{subject to} \quad (\mathbf{I} \otimes \mathbf{S}^\top)(\mathbf{I} \otimes \mathbf{A})x = (\mathbf{I} \otimes \mathbf{S}^\top)e, \quad \mathbf{C}x = 0, \quad (\text{J.41})$$

which is a sketch-and-project method applied on the linear system, with update as per (10.20):

$$x^{k+1} = x^k - (\mathbf{H} \otimes \mathbf{I})((\mathbf{I} \otimes \mathbf{A})x - e) - (\mathbf{I} \otimes \mathbf{H})((\mathbf{I} \otimes \mathbf{A})x - e) + (\mathbf{HA} \otimes \mathbf{H})((\mathbf{I} \otimes \mathbf{A})x - e)$$

for $\mathbf{H} \stackrel{\text{def}}{=} \mathbf{S}(\mathbf{S}^\top \mathbf{AS})^{-1} \mathbf{S}^\top$. Using substitution $\hat{x} = (\mathbf{A}^{\frac{1}{2}} \otimes \mathbf{A}^{\frac{1}{2}})x$; $\hat{\mathbf{S}} = \mathbf{A}^{\frac{1}{2}} \mathbf{S}$ and comparing

to (10.11), we get

$$\mathbf{Z} = \mathbf{I} \otimes \mathbf{I} - (\mathbf{I} - \mathbf{P}) \otimes (\mathbf{I} - \mathbf{P})$$

for \mathbf{P} as defined inside the statement of Theorem 10.4.1. Therefore, we have all necessary information to apply the results from [178], recovering Theorem 10.4.1.

J.8 Linear operators in Euclidean spaces

Here we provide some technical lemmas and results for linear operators in Euclidean space, that we used in the main body of the chapter. Most of these results can be found in standard textbooks of analysis, such as [162]. We give them here for completion.

Let $\mathcal{X}, \mathcal{Y}, \mathcal{Z}$ be Euclidean spaces, equipped with inner products. Formally, we should use a notation that distinguishes the inner product in each space. But instead we use $\langle \cdot, \cdot \rangle$ to denote the inner product on all spaces, as it will be easy to determine from which space the elements are in. That is, for $x_1, x_2 \in \mathcal{X}$, we denote by $\langle x_1, x_2 \rangle$ the inner product between x_1 and x_2 in \mathcal{X} .

Let

$$\|T\| \stackrel{\text{def}}{=} \sup_{\|x\| \leq 1} \|Tx\|,$$

denote the operator norm of T . Let $0 \in L(\mathcal{X}, \mathcal{Y})$ denote the zero operator and $I \in L(\mathcal{X}, \mathcal{Y})$ the identity map.

The adjoint. Let $T^* \in L(\mathcal{Y}, \mathcal{X})$ denote the unique operator that satisfies

$$\langle Tx, y \rangle = \langle x, T^*y \rangle,$$

for all $x \in \mathcal{X}$ and $y \in \mathcal{Y}$. We say that T^* is the *adjoint* of T . We say T is *self-adjoint* if $T = T^*$. Since for all $x \in \mathcal{X}$ and $s \in \mathcal{S}$,

$$\langle x, (ST)^*s \rangle = \langle STx, s \rangle_{\mathcal{S}} = \langle Tx, S^*s \rangle_{\mathcal{Y}} = \langle x, T^*S^*s \rangle,$$

we have

$$(ST)^* = T^*S^*.$$

Lemma J.8.1. For $T \in L(\mathcal{X}, \mathcal{Y})$ we have that $\text{Range}(T^*)^\perp = \text{Null}(T)$. Thus

$$\mathcal{X} = \text{Range}(T^*) \oplus \text{Null}(T) \tag{J.42}$$

$$\mathcal{Y} = \text{Range}(T) \oplus \text{Null}(T^*) \tag{J.43}$$

Proof. See 3.2.6 in [162]. □

J.8.1 Positive operators

We say that $G \in L(\mathcal{X})$ is positive if it is self-adjoint and if $\langle x, Gx \rangle \geq 0$ for all $x \in \mathcal{X}$. Let $(e_j)_{j=1}^\infty \in \mathcal{X}$ be an orthonormal basis. The trace of G is defined as

$$\mathbf{Tr}(G) \stackrel{\text{def}}{=} \sum_{j=1}^{\infty} \langle Ge_j, e_j \rangle. \quad (\text{J.44})$$

The definition of trace is independent of the choice of basis due to the following lemma.

Lemma J.8.2. *If U is unitary and $G \geq 0$ then $\mathbf{Tr}(UGU^*) = \mathbf{Tr}(G)$.*

Proof. See 3.4.3 and 3.4.4 in [162]. □

Lemma J.8.3. *If $P \in L(\mathcal{X})$ is a projection matrix then $\mathbf{Tr}(P) = \dim(\mathbf{Range}(P)) = \mathbf{Rank}(P)$.*

Proof. Let $d = \dim(\mathbf{Range}(P))$ which is possibly infinite. Given that P is a projection we have that $\mathbf{Range}(P)$ is a closed subspace and thus there exists orthonormal basis $(e_j)_{j=1}^d$ of $\mathbf{Range}(P)$. Consequently, $\mathbf{Tr}(P) \stackrel{(\text{J.44})}{=} \sum_{j=1}^d 1 = d = \dim(\mathbf{Range}(P))$. □

A square root of an operator $G \in L(\mathcal{X})$ is an operator $R \in L(\mathcal{X})$ such that $R^2 = G$.

Lemma J.8.4. *If $G : \mathcal{X} \rightarrow \mathcal{X}$ is positive, then there exists a unique positive square root of G which we denote by $G^{1/2}$.*

Proof. See 3.2.11 in [162]. □

Lemma J.8.5. *For any $T \in L(\mathcal{X}, \mathcal{Y})$ and any $G \in L(\mathcal{Y}, \mathcal{Y})$ that is positive and injective,*

$$\mathbf{Null}(T) = \mathbf{Null}(T^*GT), \quad (\text{J.45})$$

and

$$\overline{\mathbf{Range}(T^*)} = \overline{\mathbf{Range}(T^*GT)}. \quad (\text{J.46})$$

Proof. The inclusion $\mathbf{Null}(T) \subset \mathbf{Null}(T^*GT)$ is immediate. For the opposite inclusion, let $x \in \mathbf{Null}(T^*GT)$. Since G is positive we have by Lemma J.8.4 that there exists a square root with $G^{1/2}G^{1/2} = G$. Therefore, $\langle x, T^*GTx \rangle = \langle G^{1/2}Tx, G^{1/2}Tx \rangle = 0$, which implies that $G^{1/2}Tx = 0$. Since G is injective, it follows that $G^{1/2}$ is injective and thus $x \in \mathbf{Null}(T)$. Finally (J.46) follows by taking the orthogonal complements of (J.45) and observing Lemma J.8.1. □

As an immediate consequence of (J.45) and (J.46) we have the following lemma.

Corollary J.8.6. *For $G : \mathcal{X} \rightarrow \mathcal{X}$ positive we have that*

$$\mathbf{Null}(G^{1/2}) = \mathbf{Null}(G) \quad (\text{J.47})$$

$$\overline{\mathbf{Range}(G^{1/2})} = \overline{\mathbf{Range}(G)} \quad (\text{J.48})$$

J.8.2 Pseudoinverse

For a bounded linear operator T define the pseudoinverse of T as follows.

Definition J.8.7. Let $T \in L(\mathcal{X}, \mathcal{Y})$ such that $\mathbf{Range}(T)$ is closed. $T^\dagger : \mathcal{Y} \rightarrow \mathcal{X}$ is said to be the pseudoinverse if

1. $T^\dagger T x = x$ for all $x \in \mathbf{Range}(T^*)$.
2. $T^\dagger x = 0$ for all $x \in \mathbf{Null}(T^*)$.
3. If $x \in \mathbf{Null}(T)$ and $y \in \mathbf{Range}(T^*)$ then $T^\dagger(x + y) = T^\dagger x + T^\dagger y$.

It follows directly from the definition (see [39] for details) that T^\dagger is a unique bounded linear operator. The following properties of pseudoinverse will be important.

Lemma J.8.8 (Properties of pseudoinverse). Let $T \in L(\mathcal{X}, \mathcal{Y})$ such that $\mathbf{Range}(T)$ is closed. It follows that

1. $TT^\dagger T = T$
2. $\mathbf{Range}(T^\dagger) = \mathbf{Range}(T^*)$ and $\mathbf{Null}(T^\dagger) = \mathbf{Null}(T^*)$
3. $(T^*)^\dagger = (T^\dagger)^*$
4. If T is self-adjoint and positive then T^\dagger is self-adjoint and positive.
5. $T^\dagger TT^* = T^*$, that is, $T^\dagger T$ projects orthogonally onto $\mathbf{Range}(T^*)$ and along $\mathbf{Null}(T)$.
6. Consider the linear system $Tx = d$ where $d \in \mathbf{Range}(T)$. It follows that

$$T^\dagger d = \arg \min_{x \in \mathcal{X}} \frac{1}{2} \|x\|^2 \quad \text{subject to } Tx = d. \quad (\text{J.49})$$

7. $T^\dagger = T^*(TT^*)^\dagger$

Proof. The proof of first five items can be found in [39]. The proof of (6) is alternative characterization of the pseudoinverse and it can be established by using that $d \in \mathbf{Range}(T)$ together with item 1 thus $TT^\dagger d = d$. The proof then follows by using the orthogonal decomposition $\mathbf{Range}(T^*) \oplus \mathbf{Null}(T)$ to show that $T^\dagger d$ is indeed the minimum of (J.49). Finally item (7) is a direct consequence of the previous items. \square

Appendix K

Accepted Papers

- [77] Filip Hanzely, Konstantin Mishchenko, and Peter Richtárik. SEGA: Variance reduction via gradient sketching. In *Advances in Neural Information Processing Systems*, pages 2083–2094, 2018.
- [78] Filip Hanzely and Peter Richtárik. Accelerated coordinate descent with arbitrary sampling and best rates for minibatches. In *Proceedings of Machine Learning Research*, pages 304–312. PMLR, 16–18 Apr 2019.
- [58] Robert M Gower, Filip Hanzely, Peter Richtárik, and Sebastian U Stich. Accelerated stochastic matrix inversion: general theory and speeding up bfgs rules for faster second-order optimization. In *Advances in Neural Information Processing Systems*, pages 1619–1629, 2018.
- [45] Aritra Dutta, Filip Hanzely, and Peter Richtárik. A nonconvex projection method for robust PCA. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 1468–1476, 2019.
- [55] Eduard Gorbunov, Filip Hanzely, and Peter Richtárik. A unified theory of SGD: Variance reduction, sampling, quantization and coordinate descent. In *The 23rd International Conference on Artificial Intelligence and Statistics*, 2020.
- [137] Konstantin Mishchenko, Filip Hanzely, and Peter Richtárik. 99% of worker-master communication in distributed optimization is not needed. In *36th Conference on Uncertainty in Artificial Intelligence, (UAI 2020)*. AUAI, 2020.
- [44] Aritra Dutta, Filip Hanzely, Jingwei Liang, and Peter Richtárik. Best pair formulation & accelerated scheme for non-convex principal component pursuit. *IEEE Transactions on Signal Processing*, 2020.
- [76] Filip Hanzely, Dmitry Kovalev, and Peter Richtárik. Variance reduced coordinate descent with acceleration: New method with a surprising application to finite-sum problems. In *International Conference on Machine Learning*, 2020.
- [74] Filip Hanzely, Nikita Doikov, Peter Richtárik, and Yurii Nesterov. Stochastic subspace cubic Newton method. In *International Conference on Machine Learning*, 2020.

Appendix L

Submitted Papers

[81] Filip Hanzely, Peter Richtárik, and Lin Xiao. Accelerated Bregman proximal gradient methods for relatively smooth convex optimization. *arXiv preprint arXiv:1808.03045*, 2018.

[79] Filip Hanzely and Peter Richtárik. One method to rule them all: Variance reduction for data, parameters and many new methods. *arXiv preprint arXiv:1905.11266*, 2019.

[80] Filip Hanzely and Peter Richtárik. Federated learning of a mixture of global and local models. *arXiv preprint arXiv:2002.05516*, 2020.

[75] Filip Hanzely, Slavomír Hanzely, Samuel Horváth, and Peter Richtárik. Lower bounds and optimal algorithms for personalized federated learning. Technical Report, 2020.