A Sparsity Preserving Stochastic Gradient Method for Composite Optimization

Qihang Lin* Xi Chen† Javier Peña‡

April 23, 2011

Abstract

We propose new stochastic gradient algorithms for solving convex composite optimization problems. In each iteration, our algorithms utilize a stochastic oracle of the gradient of the smooth component in the objective function. Our algorithms are based on a stochastic version of the *estimate sequence technique* introduced by Nesterov (*Introductory Lectures on Convex Optimization: A Basic Course, Kluwer, 2003*). We establish convergence results for the expectation and variance as well as large deviation properties of the objective value of the iterates generated by our algorithm. When applied to sparse regression problems, our algorithms have the advantage of readily enforcing sparsity at all iterations. We present some numerical experiments on simulated data sets.

1 Introduction

In the past decade, convex composite optimization has been widely applied in a variety of areas including statistical machine learning, data mining and compressed sensing. A *convex composite optimization problem* is a problem of the form

$$\min_{x\in\mathbf{X}} \phi(x) \equiv f(x) + h(x). \tag{1}$$

Here, the set of feasible solutions $\mathbf{X}$ is a convex set in $\mathbb{R}^n$. The function $h(x)$ is convex but non-smooth. The function $f(x)$ is convex and smooth. Its gradient $\nabla f(x)$ is Lipschitz continuous with Lipschitz constant L, that is,

$$\|\nabla f(x) - \nabla f(y)\| \le L\|x - y\|, \quad \forall x, y \in \mathbb{R}^n,$$

*Tepper School of Business, Carnegie Mellon University, PA 15213
†Department of Machine Learning, Carnegie Mellon University, PA 15213
‡Tepper School of Business, Carnegie Mellon University, PA 15213

or equivalently,

$$f(x) \leq f(y) + \langle x - y, \nabla f(y) \rangle + \frac{L}{2}\|x - y\|^2, \quad \forall x, y \in \mathbb{R}^n. \tag{2}$$

In this paper, the norm $\|\cdot\|$ denotes the Euclidean norm or l_2-norm in $\mathbb{R}^n$ and $\langle \cdot, \cdot \rangle$ denotes the inner product of vectors in $\mathbb{R}^n$.

For solving problem (1), *accelerated gradient algorithms*, for example the ones in [1, 8, 9, 10, 18], have attracted great interest in the last few years. In addition to ease of implementation, these algorithms are capable of solving challenging problems of very large size thanks to their low memory requirement in each iteration. An accelerated gradient algorithm with favorable convergence rate usually requires an exact oracle of the gradient $\nabla f(x)$ for problem (1). However, there are situations where only a random approximation of the gradient is available. Such an approximation is often a *stochastic gradient*, that is, $\nabla f(x)$ plus some random noise with zero expected value. Therefore, it is interesting and important to develop stochastic versions of accelerated gradient methods to solve composite optimization problems of the form (1) using noisy first-order information.

A typical situation to consider stochastic gradients is in stochastic optimization problems [2, 4, 6] where the objective function $f(x)$ is given as an expectation of a random value function $F(x, \xi)$, i.e. $f(x) = \mathbb{E}F(x, \xi)$, where ξ is a random variable and the expectation is taken over ξ. In this situation, a multidimensional numerical integral is required to compute the exact gradient $\nabla f(x) = \mathbb{E}\nabla F(x, \xi)$. That would be too time consuming especially when the dimension of ξ is high. It could be even worse if the distribution of ξ is unknown, as in this case there is no way to obtain the exact gradient $\nabla f(x)$. A way to handle this difficulty is to draw a random sample $\{\xi_1, \ldots, \xi_m\}$ from the distribution of ξ and use as stochastic gradient the sample mean $\frac{1}{m}\sum_{i=1}^{m} \nabla F(x, \xi_i)$.

Research related to stochastic gradient methods dates back to Robbins and Monro's stochastic approximation algorithm [14] in 1951, which was further developed by Polyak and Juditsky [12]. Using a subgradient averaging technique, a mirror-descent stochastic approximation method is proposed by Nemirovski et al [6] for minimizing general non-smooth convex functions on a compact set. This method exhibits an optimal $O(\frac{1}{\sqrt{N}})$ expected convergence rate and triggered a series of works [2, 3, 4, 20] on stochastic gradient algorithms within the framework of complexity theory of continuous optimization [7, 9]. For example, Lan [4] proposed the first unified stochastic gradient method for smooth, nonsmooth and composite optimization. This approach is further generalized in [2], which achieves an optimal $O(\frac{1}{N})$ expected convergence rate for strongly convex composite optimizations.

For the algorithms solving (1), a large family of applications comes from *sparse regression* problems, e.g., lasso [15], group lasso [21], fused lasso [16], etc. In a typical sparse regression setting,

$$f(x) \equiv \frac{1}{K}\sum_{i=1}^{K} \ell(a_i^T x; b_i),$$

where $\{a_i\}_{i=1}^K$ are the K input data points, $\{b_i\}_{i=1}^K$ are the corresponding responses and $\ell(a_i^T x; b_i)$ is the smooth loss function (e.g., l_2-loss where $\ell(a_i^T x; b_i) = (a_i^T x - b_i)^2$). The non-smooth function $h(x)$ is a sparsity-inducing regularizer (e.g., l_1-norm, $h(x) = \|x\|_1$). It enforces the sparsity in the optimal solution x. A sparse optimal solution x indicates the important variables which correspond to non-zeros values of x. Although the sparsity of the optimal solution is due to the optimization model itself and theoretically independent of the algorithm used, different optimization algorithms will recover sparsity in different ways. In the following paragraphs, we discuss the issue of sparsity preserving in both deterministic and stochastic gradient methods for sparse regression.

In most accelerated gradient methods to solve (1), a *composite gradient mapping*, which itself can be a composite optimization problem, has to be solved in each iteration. At iteration k, this projection mapping is usually of the form

$$x_k = \arg\max_{x \in \mathbf{X}} (\|x - \widehat{x}_k\|_2^2 + \lambda_k h(x)) \tag{3}$$

where λ_k is a constant, $\widehat{x}_k$ is a vector computed based on the previous iterates and $h(x)$ is the sparsity enforcing term such as l_1-norm [15] or l_1/l_2-norm [5]. This subproblem can often be solved in closed form using a shrinkage operator [1] (or so-called soft thresholding). This automatically guarantees a sparse structure in the solution x_k.

Some accelerated gradient algorithms, for example, the ones in [1, 9] and Algorithm 2 in [18], directly use x_k above as the final output. This means the convergence of these algorithms is given by showing that $\phi(x_k) - \phi(x^*)$ converges to zero, where x^* is an optimal solution to (1). We call this class of algorithms as *sparsity-preserving* algorithms. They have a wide range of applications, e.g.,, [1, 5, 13, 17] due to the sparse or low-rank (sparse in spectrum) properties their output has.

However, there also exist some accelerated gradient algorithms such as the one in [10] and Algorithms 1 and 3 in [18], whose outputs are not x_k from (3). Instead, after solving x_k from (3), these algorithms perform some additional computation with x_k to generate their output sequence. Usually, this additional computation will sacrifice the sparsity of x_k. For instance, one version of Algorithm 1 in [18] computes

$$z_k = (1 - \theta_k) x_k + \theta_k z_{k-1} \tag{4}$$

with $\theta_k \in (0, 1)$ and uses z_k as the output. Since the sparsity pattern of x_k may change with k, especially when k is not large, their linear combination z_k may not be sparse.

The same issue comes up when we solve sparse regression problems with stochastic gradient algorithms. The algorithms proposed in [4, 6] are not sparsity-preserving because they treat $f(x)$ and $h(x)$ in (1) as a whole and utilize stochastic subgradients of the non-smooth objective function $\phi(x)$. Therefore, the sub-problem (3) in [4, 6] do not contain the sparsity enforcing term $h(x)$ and thus its solution x_k is not sparse.

The accelerated stochastic approximation (AC-SA) algorithm proposed by Lan in [2] does in-

clude $h(x)$ in its projection mapping (3) and solves a sparse x_k from that. However, this algorithm can be viewed as a stochastic extension of the Algorithm 1 in [18]. Similar to its deterministic prototype, Lan's algorithm uses z_k given by an expression like (4) as the output which is not sparse as we discussed above.

Xiao develops a regularized dual average (RDA) algorithm [20], which is a stochastic extension of the dual average method of Nesterov [11]. RDA also needs to solve (3) and obtains a sparse solution x_k. However, Xiao establishes the convergence of RDA by showing that $\bar{x}_k$ converges to optimality, where $\bar{x}_k = (\sum_{i=1}^{k} x_i)/k$ which is not sparse. Although x_k demonstrates a good numerical performance in [20], there is no theoretical guarantee on the convergence of x_k.

Given the theoretical convergence of z_k in AC-SA and $\bar{x}_k$ in RDA, some of their components should be nearly zero when the iterates are close enough to the optimality. However, due to the low speed of convergence, stochastic gradient algorithms are usually stopped well before they reach optimality. Therefore, many of these components may not be small enough to be safely identified as zeros. Performing a rounding procedure on the solutions is a possible alternative, but it is not clear how to choose the rounding threshold. Moreover, if we are just interested in knowing the most relevant variables, we can stop a sparsity-preserving algorithm at a relatively early stage and check the non-zeros components. By contrast, we would have to run a non-sparsity-preserving algorithm to the very end in order to ensure a significant gap in magnitude between the zero and non-zero components.

In this paper, we propose a stochastic gradient algorithm (Algorithm SSG) for the general composite optimization problem (1). Algorithm SSG is a stochastic extension of the sparsity-preserving accelerated gradient algorithm proposed by Nesterov [9, Section 2.2] which, although not readily apparent, generalizes Algorithm FISTA in [1] and Algorithm 2 in [18]. In each iteration of our algorithm, we need to solve a sub-problem of the form (3), whose solution x_k is sparse when $h(x)$ is a sparsity enforcing term. Unlike Algorithms AC-SA or RDA, we can directly use x_k as the output. In other words, we prove that x_k converges to optimality, instead of z_k or $\bar{x}_k$. This makes our algorithm a sparsity-preserving stochastic gradient algorithm.

There is a connection between our algorithm and the one proposed by Hu et al. [3], which also generates sparse solutions. However, the algorithm in [3] is different from ours. As a consequence Hu et al.'s algorithm does not guarantee the convergence unless the iterates are assumed to stay in a bounded set. By contrast, our different updating scheme yields convergence without this assumption. Furthermore, we present analyses on the variance and the probability of a large error that were not discussed in [3].

We will rely on the following definition of a stochastic gradient in the sequel.

Definition 1. *Let f be a continuously differentiable function and ξ be a random variable taking value in an abstract measurable space (Ξ, Σ), where Σ is a σ-algebra over Ξ. If there is a vector-value*

function $G(x,\xi)$ *on* $\mathbb{R}^n \times \Xi$ *such that*

$$\mathbb{E}G(x,\xi) = \nabla f(x) \quad \textit{for } \forall x \in \mathbb{R}^n, \tag{5}$$

and

$$\mathbb{E}\|G(x,\xi) - \nabla f(x)\|^2 \leq \sigma^2 \quad \textit{for } \forall x \in \mathbb{R}^n, \tag{6}$$

where the expectation is taken over ξ*, we say that* $f(x)$ *is equipped with a stochastic first-order* σ*-oracle and* $G(x,\xi)$ *is called a stochastic gradient of* $f(x)$ *at point* x*.*

In this paper, we assume that there is a stochastic first-order σ-oracle available for the smooth component $f(x)$ in (1).

In a stochastic gradient algorithm, a sequence of iterates $\{x_k\}_{k\geq 0}$ is generated using the random first-order information given by $G(x,\xi)$. Hence, the sequence $\{x_k\}_{k\geq 0}$ is a random sequence in the feasible set $\mathbf{X}$. In each run of the algorithm, a different random sequence will be generated even if the same starting point is used. Therefore, it is more appropriate to characterize the convergence of a stochastic gradient algorithm by the convergence of the expected error, i.e. $\mathbb{E}(\phi(x_{k+1}) - \phi(x^*))$.

We show that our algorithm converges with $\mathbb{E}(\phi(x_{N+1}) - \phi(x^*)) \leq O(\frac{1}{\sqrt{N}})$, where N is the total number of iterations (see Theorem 1 and Corollary 1 in Section 3). This convergence rate can be improved to $O(\frac{1}{N})$ if the smooth component $f(x)$ in (1) is strongly convex with a *strong convexity parameter* $\mu > 0$, that is,

$$f(x) \geq f(y) + \langle x - y, \nabla f(y)\rangle + \frac{\mu}{2}\|x - y\|^2, \quad \forall x, y \in \mathbb{R}^n. \tag{7}$$

These convergence rates are asymptotically the same as those obtained by the algorithms in ([2, 4, 6]) for problems with convex and strongly convex $f(x)$ respectively, which are known to be asymptotically optimal (see [7]) in terms of N. The convergence rates are proved via a *stochastic estimate sequence*, which is the stochastic extension of the *estimate sequence* technique introduced by Nesterov to show the convergence rate of his accelerated gradient algorithm in [9].

Due to the randomness of the solution found by a stochastic gradient algorithm, knowing the convergence of $\mathbb{E}(\phi(x_{N+1}) - \phi(x^*))$ does not guarantee the quality of the solution of a single run even though $\phi(x_{N+1}) - \phi(x^*) \geq 0$. Indeed, it is easy to construct a sequence of nonnegative random variables whose means converge to zero but variances go to infinity. Therefore, it is interesting to study how $\mathbb{V}(\phi(x_{k+1}) - \phi(x^*))$, the variance of the error, evolve throughout the algorithm. We show that the iterates generated by our algorithm satisfy $\mathbb{V}(\phi(x_{N+1}) - \phi(x^*)) \leq O(\frac{1}{N})$ provided the fourth moment of $G(x,\xi) - \nabla f(x)$ is bounded, i.e.,

$$\mathbb{E}\|G(x,\xi) - \nabla f(x)\|^4 \leq \sigma^4 \quad \text{for } \forall x \in \mathbb{R}^n. \tag{8}$$

(See Theorem 2 in Section 4.) It is easy to see that (8) implies (6).

These results establish the reduction on the randomness of the iterates as the Algorithm progresses. Knowing the convergence of the expected error in addition to its low variance increases our confidence on the quality of the solution from a single run of our algorithm, especially for large N. To the best of the authors' knowledge, our work presents the first analysis of variance of solutions generated by a stochastic gradient algorithm.

Besides the variance of the errors, another measure of the quality of the solution is the probability of large error. To study the probability of large error of a stochastic gradient algorithm, we look for a function $\epsilon(N,\delta)$ on N and $\delta \in (0,1)$ such that $\text{Prob}\left[\phi(x_{N+1}) - \phi(x^*) \geq \epsilon(N,\delta)\right] \leq \delta$. It is easy to get $\epsilon(N,\delta) = O(\frac{1}{\delta\sqrt{N}})$ (and $\epsilon(N,\delta) = O(\frac{1}{\delta N})$ when $f(x)$ is strongly convex) using the convergence of the expected error. Assuming (8), we obtain $\epsilon(N,\delta) = O(\frac{1}{\sqrt{\delta N}})$ from the convergence of the variance of error. Under an assumption stronger than (8), i.e.,

$$\mathbb{E}\left[\exp\left(\|G(x,\xi) - \nabla f(x)\|^2/\sigma^2\right)\right] \leq \exp(1) \quad \text{for } \forall x \in \mathbb{R}^n, \tag{9}$$

Lan [4] and Ghadimi and Lan [2] show that $\epsilon(N,\delta) = O\left(\frac{\ln(1/\delta)}{\sqrt{N}}\right)$ for their AC-SA algorithm. By further strengthening (9) to $\|G(x,\xi) - \nabla f(x)\| \leq \sigma$ for all $x \in \mathbb{R}^n$, we prove that $\epsilon(N,\delta) = O\left(\sqrt{\frac{\ln(1/\delta)}{N}}\right)$ for our algorithm (see Theorem 3 in Section 5). This bound is also achieved by Xiao's RDA algorithm [20] under the same assumption.

The rest of this paper is organized as follows. Section 2 presents our stochastic gradient descent algorithm. Sections 3 and 4 present the main results on the expectation and variance of the error of the solutions generated by our algorithm. Section 5 presents some results on the probability of large errors. Section 6 presents some applications and numerical results of our algorithms.

2 Stochastic gradient algorithm

In this section, we present our sparsity-preserving stochastic gradient algorithm (Algorithm SSG). The main ideas for this algorithm are based on the optimal first-order method proposed by Nesterov in [9]. As we mentioned in Section 1, our algorithm, just as its deterministic prototype, enforces sparsity of intermediate solutions if applied to sparse regression problems.

We use the notion of *estimate sequence*, as defined by Nesterov in [9], to facilitate the derivation of our algorithms.

Definition 2. *A pair of sequences* $(\{V_k(x)\}_{k\geq 0}, \{\lambda_k > 0\}_{k\geq 0})$ *is called an estimate sequence of* $\phi(x)$ *if* $\lambda_k \to 0$ *and for any* $x \in \mathbb{R}^n$ *and all* $k \geq 0$*, we have*

$$V_k(x) \leq (1-\lambda_k)\phi(x) + \lambda_k V_0(x). \tag{10}$$

If $\{V_k(x)\}_{k\geq 0}$ *is a sequence of random functions, we call the pair* $(\{V_k(x)\}_{k\geq 0}, \{\lambda_k \geq 0\}_{k\geq 0})$ *a stochastic estimate sequence of* $\phi(x)$ *if* (10) *holds almost surely.*

The next lemma shows how an estimate sequence is potentially useful to guarantee the convergence of a sequence of iterates.

Lemma 1. *Suppose* $\{V_k(x)\}_{k\geq 0}$ *and* $\{\lambda_k\}_{k\geq 0}$ *is an stochastic estimate sequence for function* $\phi(x)$ *and* x^* *is an optimal solution to* (1)*. If there exist sequences of random variables* $\{x_k\}_{k\geq 0} \subset \mathbf{X}$*,* $\{B_k \geq 0\}_{k\geq 0} \subset \mathbb{R}$*,* $\{\epsilon_k\}_{k\geq 0} \subset \mathbb{R}^n$ *and* $\{\delta_k\} \subset \mathbb{R}$ *such that*

$$\phi(x_k) - B_k \leq V_k^* := \min[V_k(x) - \langle \epsilon_k, x\rangle - \delta_k] \tag{11}$$

holds almost surely for all $k \geq 0$*, then*

$$\phi(x_k) - \phi(x^*) \leq \lambda_k \left[V_0(x_0) - \phi(x^*)\right] + B_k - \langle \epsilon_k, x^*\rangle - \delta_k \quad \textit{almost surely.} \tag{12}$$

If $\{\epsilon_k\}_{k\geq 0}$ *and* $\{\delta_k\}_{k\geq 0}$ *further satisfy*

$$\mathbb{E}\epsilon_k = 0, \quad \mathbb{E}\delta_k = 0 \tag{13}$$

for all $k \geq 0$*, then*

$$\mathbb{E}(\phi(x_k) - \phi(x^*)) \leq \lambda_k \mathbb{E}\left[V_0(x_0) - \phi(x^*)\right] + \mathbb{E}B_k. \tag{14}$$

Proof. By (11) and (10), we have

$$\begin{aligned} \phi(x_k) - B_k &\leq \min[V_k(x) - \langle \epsilon_k, x\rangle - \delta_k] \\ &\leq \min[(1-\lambda_k)\phi(x) + \lambda_k V_0(x) - \langle \epsilon_k, x\rangle - \delta_k] \\ &\leq (1-\lambda_k)\phi(x^*) + \lambda_k V_0(x^*) - \langle \epsilon_k, x^*\rangle - \delta_k \end{aligned}$$

almost surely. Then we just take the expectation on (12) to get (14). □

If each x_k is deterministic with B_k, ϵ_k and δ_k deterministically equal to zeros, this lemma reduces to Lemma 2.2.1 in [9]. The latter states that if an estimate sequence satisfies $\phi(x_k) \leq V_k^* := \min V_k(x)$ then $\phi(x_k) - \phi(x^*) \leq \lambda_k \left[V_0(x_0) - \phi(x^*)\right]$. In [9], V_k^* is an upper bound on $\phi(x_k)$. By contrast, now we have to supplement V_k^* with a nonegative term B_k to bound $\phi(x_k)$ from above. Moreover, V_k^* is the minimal value of $V_k(x)$ in [9] while now it is the minimal value of $V_k(x)$ plus a linear perturbation $\langle \epsilon_k, x\rangle + \delta_k$. The equation (13) states this linear perturbation has a zero effect on $V(x)$ on average.

Lemma 1 suggests a way to construct a sequence of random solutions $\{x_k\}_{k\geq 0}$ of (1) which converges to optimality on average. This construction involves an estimate sequence V_k and λ_k as well as sequences of x_k, B_k, ϵ_k and δ_k satisfying (11) and (13). If we can further guarantee that both $\lambda_k \to 0$ and $\mathbb{E}B_k \to 0$, the convergence of x_k will follow.

The following lemma will be used to construct an estimate sequence V_k and λ_k recursively for the objective function $\phi(x)$ in (1). This lemma was first given by Hu et al. [3]. However, they use

this lemma to prove the convergence of their stochastic gradient algorithm in a way different from what we do here. For the sake of completeness, we include its proof in the Appendix.

Lemma 2 (Hu et al. [3])**.** *Suppose $f(x)$ is smooth and μ-strongly convex ($\mu = 0$ if $f(x)$ is convex but not strongly convex) with $\nabla f(x)$ L-Lipschitz continuous, $h(x)$ is convex and non-smooth, and $G(x,\xi)$ is the stochastic gradient of $f(x)$ as in Definition 1. For any $\bar{x} \in \mathbb{R}^n$, we denote*

$$\widehat{x} = \arg\min_{x \in \mathbf{X}} \{ f(\bar{x}) + \langle G(\bar{x},\xi), x - \bar{x} \rangle + \frac{\bar{L}}{2}\|x - \bar{x}\|^2 + h(x) \}.$$

Let $g = \bar{L}(\bar{x} - \widehat{x})$ and $\Delta = G(\bar{x},\xi) - \nabla f(\bar{x})$. Then, for any $x \in \mathbf{X}$, we have

$$\phi(x) \geq \phi(\widehat{x}) + \langle g, x - \bar{x} \rangle + \frac{2\bar{L} - L}{2\bar{L}^2}\|g\|^2 + \frac{\mu}{2}\|x - \bar{x}\|^2 + \langle \Delta, \widehat{x} - x \rangle \tag{15}$$

almost surely.

This lemma is a stochastic version of Theorem 2.2.7 in [9]. The inequality (15) provides a stochastic lower bound on the deterministic objective function $\phi(x)$ at the point x, due to the random nature of ξ, Δ and $\widehat{x}$. Based on this lower bound, the next lemma presents a recursive construction of an estimate sequence V_k and λ_k as well as a sequence of random solution x_k for (1). It can be viewed as a generalization of Lemma 2.2.2 in [9] and they share similar proofs.

Lemma 3. *Suppose $f(x)$ is smooth and μ-strongly convex ($\mu = 0$ if $f(x)$ is convex but not strongly convex) with $\nabla f(x)$ L-Lipschitz continuous and $h(x)$ is convex and non-smooth, and $G(x,\xi)$ is a stochastic gradient of $f(x)$ as in Definition 1.*

Let $V_0(x)$ be an arbitrary function, $\{y_k\}_{k\geq 0}$ an arbitrary sequence in $\mathbb{R}^n$, $\{\alpha_k\}_{k\geq 0}$ a sequence in $(0,1)$ with $\sum_{k=0}^{\infty} \alpha_k = \infty$, and $\{L_k\}_{k\geq 0}$ a sequence in (L,∞).

Define the sequences $\{x_k\}_{k\geq 1}, \{\lambda_k\}_{k\geq 0}$, and $\{V_k(x)\}_{k\geq 0}$ as follows. Let

$$x_{k+1} = \arg\min_{x \in \mathbf{X}} \left\{ f(y_k) + \langle G(y_k,\xi_k), x - y_k \rangle + \frac{L_k}{2}\|x - y_k\|^2 + h(x) \right\}, \tag{16}$$

where $\xi_1, \xi_2, \ldots$ are i.i.d. random variables with the same distribution as ξ. Let

$$\lambda_k = \begin{cases} 1 & \text{if } k = 0 \\ \prod_{i=1}^{k-1}(1 - \alpha_i) & \text{if } k \geq 1. \end{cases} \tag{17}$$

Finally, let

$$\begin{aligned} V_{k+1}(x) =& (1 - \alpha_k) V_k(x) \\ &+ \alpha_k \left[\phi(x_{k+1}) + \langle g_k, x - y_k \rangle + \frac{2L_k - L}{2L_k^2}\|g_k\|^2 + \frac{\mu}{2}\|x - y_k\|^2 + \langle \Delta_k, x_{k+1} - x \rangle \right], \end{aligned} \tag{18}$$

where $\Delta_k = G(y_k, \xi_k) - \nabla f(y_k)$ *and* $g_k = L_k(y_k - x_{k+1})$.

Then $(\{V_k(x)\}, \{\lambda_k\})$ *is a stochastic estimate sequence of* $\phi(x) = f(x) + h(x)$.

Proof. The equality (10) holds for $k = 0$ since $\lambda_0 = 1$. Suppose (10) holds for $k \geq 0$. Given any $x \in \mathbf{X}$, we have

$$\begin{aligned} V_{k+1}(x) &\leq (1-\alpha_k)V_k(x) + \alpha_k\phi(x) \quad \text{(by (18) and Lemma 2)} \\ &= (1-(1-a_k)\lambda_k)\phi(x) + (1-\alpha_k)(V_k(x) - (1-\lambda_k)\phi(x)) \\ &\leq (1-(1-a_k)\lambda_k)\phi(x) + (1-\alpha_k)\lambda_k V_0(x) \quad \text{(by (10))} \\ &= (1-\lambda_{k+1})\phi(x) + \lambda_{k+1}V_0(x) \quad \text{(by (17))}. \end{aligned}$$

Therefore, (10) will hold for $k+1$. It is easy to show that $\lambda_k \to 0$ using the properties of α_k. Then $\{V_k(x)\}$ and $\{\lambda_k\}$ is an estimated sequence of $\phi(x)$. □

By Lemma 3, we are able to construct a stochastic estimated sequence $\{V_k(x)\}$ and $\{\lambda_k\}$. The randomness of functions $\{V_k(x)\}$ are due to $\{\xi_k\}$. In order to apply Lemma 1, we still need to construct the sequences $\{x_k\}$, $\{B_k\}$, $\{\epsilon_k\}$ and $\{\delta_k\}$ satisfying (11) and (13). To do so, we have to investigate the particular structure of $V_k(x)$ in more detail. According to (18), if we choose $V_0(x)$ to be a quadratic function as suggested in [9], each of the resulting $V_k(x)$ with $k \geq 1$ will be quadratic as well. The following lemma describes the quadratic formulation of $V_k(x)$. We its proof to the appendix as it contains only technical algebraic details.

Lemma 4. *Suppose we choose* $V_0(x) = V_0^* + \frac{\gamma_0}{2}\|x - v_0\|^2 + \langle \epsilon_0, x\rangle + \delta_0$ *with* $V_0^* = \phi(x_0)$, $\epsilon_0 = 0$ *and* $\delta_0 = 0$ *for arbitrary* $x_0 \in \mathbf{X}$, $v_0 \in \mathbf{X}$ *and* $\gamma_0 > 0$ *in Lemma 3. Then the sequence* $\{V_k(x)\}$ *defined by* (18) *can be written as*

$$V_k(x) = V_k^* + \frac{\gamma_k}{2}\|x - v_k\|^2 + \langle \epsilon_k, x\rangle + \delta_k, \tag{19}$$

where the sequences $\{V_k^*\}$, $\{v_k\}$, $\{\gamma_k\}$, $\{\epsilon_k\}$ *and* $\{\delta_k\}$ *are recursively defined as*

$$\gamma_{k+1} = (1-\alpha_k)\gamma_k + \alpha_k\mu \tag{20}$$

$$v_{k+1} = \frac{(1-\alpha_k)\gamma_k v_k + \alpha_k\mu y_k - \alpha_k g_k}{\gamma_{k+1}} \tag{21}$$

$$\delta_{k+1} = (1-\alpha_k)[\delta_k - \langle \Delta_k, x_k\rangle] + \langle \Delta_k, y_k\rangle \tag{22}$$

$$\epsilon_{k+1} = (1-\alpha_k)\epsilon_k - \alpha_k\Delta_k \tag{23}$$

$$\begin{aligned} V_{k+1}^* = {} & (1-\alpha_k)V_k^* + \alpha_k\phi(x_{k+1}) + \left(\alpha_k\frac{2L_k - L}{2L_k^2} - \frac{\alpha_k^2}{2\gamma_{k+1}}\right)\|g_k\|^2 \\ & + \frac{\alpha_k(1-\alpha_k)\gamma_k}{\gamma_{k+1}}\left(\frac{\mu}{2}\|y_k - v_k\|^2 + \langle g_k, v_k - y_k\rangle\right) + \alpha_k\langle \Delta_k, x_{k+1}\rangle \\ & - \alpha_k\langle \Delta_k, y_k\rangle - (1-\alpha_k)\langle \Delta_k, y_k - x_k\rangle, \end{aligned} \tag{24}$$

where $\Delta_k = G(y_k, \xi_k) - \nabla f(y_k)$ *and* $g_k = L_k(y_k - x_{k+1})$.

Proof. See the Appendix. □

This lemma is a generalization of Lemma 2.2.3 in [9] which guarantees $V_k(x) = V_k^* + \frac{\gamma_k}{2}\|x - v_k\|^2$. In Lemma 4, an additional linear perturbation $\langle \epsilon_k, x\rangle + \delta_k$ is added due to the use of stochastic gradient $G(x, \xi)$. By Lemma 4, $V_k^* = \min[V_k(x) - \langle \epsilon_k, x\rangle - \delta_k]$. To obtain (12), we need a sequence $\{B_k\}$ such that (11) is satisfied.

We observe that the conclusions of Lemma 3 and Lemma 4 hold for any arbitrary sequence $\{y_k\}$. Although the sequences $\{x_k\}$ and $\{v_k\}$ are determined by $\{y_k\}$, we still have complete freedom to choose $\{y_k\}$. In the following proposition, we show that the choice of $\{y_k\}$ given by Nesterov [9] for the deterministic gradient algorithm can be inherited by our stochastic gradient algorithm in order to guarantee (11). For ease of exposition, we defer the technical proof of this proposition to the appendix.

Proposition 1. *Under the assumptions of Lemmas 3 and 4, if we further choose* $\alpha_k \in (0, 1)$ *with*

$$2L_k\alpha_k^2 \le (1 - \alpha_k)\gamma_k + \alpha_k\mu \tag{25}$$

and

$$y_k = \frac{\alpha_k\gamma_k v_k + \gamma_{k+1}x_k}{\gamma_k + \alpha_k\mu} \tag{26}$$

then (11) *holds with* x_k, ϵ_k *and* δ_k *defined as in Lemma 3 and Lemma 4, and* B_k *defined by*

$$\begin{cases} B_0 = 0 \\ B_{k+1} = \frac{\|\Delta_k\|^2}{4L_k^2 A_k} + (1 - \alpha_k)B_k \end{cases} \tag{27}$$

where

$$A_k = \frac{2L_k - L}{2L_k^2} - \frac{\alpha_k^2}{2\gamma_{k+1}}. \tag{28}$$

Proof. See the Appendix. □

We summarize what we have done so far. Lemma 1 suggests that, in order to construct a sequence $\{x_k\}$ converging to an optimal solution of (1), we can construct an estimate sequence and some auxiliary sequences ϵ_k, δ_k and B_k such that (11) is satisfied. Lemma 3 shows a way to construct an estimate sequence with arbitrary $V_0(x)$ and $\{y_k\}$. Lemma 4 suggests a choice of $V_0(x)$ such that the induced $V_k(x)$ is simple a quadratic function plus a stochastic linear perturbation involving ϵ_k and δ_k. Finally, Proposition 1 tells that, if y_k and B_k are given by (26) and (27), the condition (11) is satisfied. The whole procedure is a modification of Nesterov's derivation of his deterministic gradient algorithm [9]. The difference is that we use stochastic gradient $G(x, \xi)$ in the construction instead of $\nabla f(x)$.

Although we use $V_k(x)$, B_k, δ_k and ϵ_k in the construction of x_k, we do not have to compute any one of them explicitly in order to compute x_k. To give an algorithm to generate x_k, we only need (16), (20), (21), and (26). Our stochastic gradient algorithm is given as follows.

Algorithm 1 Stochastic Gradient Algorithm (SSG)

Input: The total number of iterations N, the Lipschitz constant L and the strongly convexity parameter μ of $f(x)$. Let $\mu = 0$ if $f(x)$ is not strongly convex.
Initialization: Choose $\gamma_0 > 0$ and $x_0 = v_0$.
Iterate for $k = 0, 1, 2, \ldots N$:

1. Choose $L_k > L$ and $\alpha_k \in (0, 1)$ with $2L_k\alpha_k^2 \leq (1 - \alpha_k)\gamma_k + \alpha_k\mu$
2. Set $\gamma_{k+1} = (1 - \alpha_k)\gamma_k + \alpha_k\mu$
3. Set $y_k = \frac{\alpha_k\gamma_k v_k + \gamma_{k+1}x_k}{\gamma_k + \alpha_k\mu}$
4. Draw an independent value ξ_k from the distribution of ξ
5. Set $x_{k+1} = \arg\min_{x\in\mathbf{X}}\{f(y_k) + \langle G(y_k, \xi_k), x - y_k\rangle + \frac{L_k}{2}\|x - y_k\|^2 + h(x)\}$
6. Set $g_k = L_k(y_k - x_{k+1})$
7. Set $v_{k+1} = \frac{(1-\alpha_k)\gamma_k v_k + \alpha_k\mu y_k - \alpha_k g_k}{\gamma_{k+1}}$

Output: x_{N+1}

Some of the main differences between Algorithm 1 and Algorithm 2.2.6 in [9, page 76] are the choices of the parameters L_k and α_k. Moreover, Algorithm 2.2.6 in [9] uses the exact gradient $\nabla f(y_k)$ while Algorithm 1 uses the stochastic gradient $G(y_k, \xi_k)$. If we set $L_k = L$ for all $k \geq 0$, compute α_k by solving $L\alpha_k^2 = (1 - \alpha_k)\gamma_k + \alpha_k\mu$ in Step 1 and replace $G(y_k, \xi_k)$ by $\nabla f(y_k)$ in Step 5, we recover Algorithm 2.2.6 in [9].

3 Convergence of the expectation of the error

In this section, we study the convergence of the expected error, $\mathbb{E}[\phi(x_k) - \phi(x^*)]$ for Algorithm 1. In Algorithm 1, after choosing γ_0, α_k, L_k and the initial tuple (x_0, y_0, v_0), we generate three sequences of vectors $\{(x_k, y_k, v_k)\}_{k\geq 0}$ utilizing a sequence of stochastic gradient $\{G(y_k, \xi_k)\}_{k\geq 0}$. It is easy to see from Algorithm 1 that x_k, y_k and v_k completely determined by the the values of random variable $\xi_0, \xi_1, \ldots, \xi_{k-1}$ for $k \geq 1$. For simplicity, we just denote the sequence $\xi_0, \xi_1, \ldots, \xi_k$ by $\xi_{[k]}$ (and let $\xi_{[-1]} = \emptyset$ for convenience) in the rest of the paper.

Before we bring up the convergence result of x_k, we made a few comments about the induced error of a stochastic gradient, i.e. $\Delta_k = G(y_k, \xi_k) - \nabla f(y_k)$. Because y_k is essentially a function of $\xi_{[k-1]}$, Δ_k is a function of $\xi_{[k]}$. According to (5), (6), the independence among ξ_k, we have the

following conditional expectations

$$\mathbb{E}_{\xi_k}(\Delta_k|\xi_{[k-1]}) = \mathbb{E}_{\xi_k}(G(y_k,\xi_k) - \nabla f(y_k)|\xi_{[k-1]}) = 0, \quad \text{for } \forall k \geq 1, \tag{29}$$

$$\mathbb{E}_{\xi_k}(\|\Delta_k\|^2|\xi_{[k-1]}) = \mathbb{E}_{\xi_k}(\|G(y_k,\xi_k) - \nabla f(y_k)\|^2|\xi_{[k-1]}) \leq \sigma^2, \quad \text{for } \forall k \geq 1. \tag{30}$$

These further imply

$$\mathbb{E}\Delta_k = \mathbb{E}_{\xi_{[k-1]}}\left[\mathbb{E}_{\xi_k}(\Delta_k|\xi_{[k-1]})\right] = \mathbb{E}_{\xi_{[k-1]}}0 = 0, \quad \text{for } \forall k \geq 1 \tag{31}$$

$$\mathbb{E}\|\Delta_k\|^2 = \mathbb{E}_{\xi_{[k-1]}}\left[\mathbb{E}_{\xi_k}(\|\Delta_k\|^2|\xi_{[k-1]})\right] \leq \mathbb{E}_{\xi_{[k-1]}}\sigma^2 = \sigma^2, \quad \text{for } \forall k \geq 1, \tag{32}$$

where the outer expectation is taken over $\xi_{[k-1]}$ while the inner expectation is taken over ξ_k. As the following lemma formalizes it, these properties of Δ_k imply (13).

Lemma 5. *The sequences $\{\epsilon_k\}_{k\geq 0}$ and $\{\delta_k\}_{k\geq 0}$ defined by* (23) *and* (22) *in Lemma 4 satisfy* (13)*.*

Proof. We prove (13) by induction. By Lemma 4, $\epsilon_0 = 0$ and $\delta_0 = 0$ so that (13) holds for $k = 0$. Suppose (13) holds for k. By (23) and (31), we have $\mathbb{E}\epsilon_{k+1} = (1-\alpha_k)\mathbb{E}\epsilon_k - \alpha_k\mathbb{E}\Delta_k = 0$. Because x_k is completely determined by $\xi_{[k-1]}$, $\mathbb{E}\left\langle \Delta_k, x_k\right\rangle = \mathbb{E}_{\xi_{[k-1]}}[\mathbb{E}_{\xi_k}(\left\langle \Delta_k, x_k\right\rangle |\xi_{[k-1]})] = 0$ by (29). Similarly, $\mathbb{E}\left\langle \Delta_k, y_k\right\rangle = 0$. Therefore, we have $\mathbb{E}\delta_{k+1} = (1-\alpha_k)[\mathbb{E}\delta_k - \mathbb{E}\left\langle \Delta_k, x_k\right\rangle] + \mathbb{E}\left\langle \Delta_k, y_k\right\rangle = 0$. Then (13) holds for $k+1$. □

We now present our main theorem about the convergence of $\mathbb{E}(\phi(x_k) - \phi(x^*))$.

Theorem 1. *Let $\{x_k, y_k, v_k\}_{k\geq 0}$ be the sequence of vectors generated in Algorithm 1 and x^* be an optimal solution to the problem* (1)*. Then*

$$\begin{aligned}\phi(x_k) - \phi(x^*) \quad \leq \quad & \lambda_k\left[\phi(x_0) - \phi(x^*) + \frac{\gamma_0}{2}\|x_0 - x^*\|^2\right] + \lambda_k\left[\sum_{i=0}^{k-1}\frac{\|\Delta_i\|^2}{2(L_i - L)\lambda_{i+1}}\right] \\ & + \lambda_k\left[\sum_{i=0}^{k-1}\frac{\left\langle \Delta_i, \alpha_i x^* + (1-\alpha_i)x_i - y_i\right\rangle}{\lambda_{i+1}}\right],\end{aligned} \tag{33}$$

and

$$\mathbb{E}(\phi(x_k) - \phi(x^*)) \leq \lambda_k\left[\phi(x_0) - \phi(x^*) + \frac{\gamma_0}{2}\|x_0 - x^*\|^2\right] + \lambda_k\left[\sum_{i=0}^{k-1}\frac{\sigma^2}{2(L_i - L)\lambda_{i+1}}\right], \tag{34}$$

where $\{\lambda_k\}_{k\geq 1}$ is defined as in (17)*.*

Proof. Let δ_k, ϵ_k, B_k be as in Lemma 4 and Proposition 1. By Proposition 1, it follows that x_k, δ_k, ϵ_k, B_k satisfy the conditions of Lemma 1 with the stochastic estimate sequence defined by (18) and (17). Then (12) and (14) hold as consequence of Lemma 1 and Lemma 5.

According to Algorithm 1, we have $2L_k\alpha_k^2 \leq (1-\alpha_k)\gamma_k + \alpha_k\mu = \gamma_{k+1}$, which implies $A_k \geq \frac{2L_k - L}{2L_k^2} - \frac{1}{2L_k} = \frac{L_k - L}{2L_k^2}$ from (28). ($A_k > 0$ since $L_k > L$ for all $k \geq 0$.) Applying this inequality to the recursive definition (27) of B_k, we get $B_{k+1} \leq \frac{\|\Delta_k\|^2}{2(L_k - L)} + (1-\alpha_k)B_k$. Further dividing this inequality by λ_{k+1}, we obtain

$$\frac{B_{k+1}}{\lambda_{k+1}} \leq \frac{\|\Delta_k\|^2}{2(L_k - L)\lambda_{k+1}} + \frac{B_k}{\lambda_k}. \tag{35}$$

Similarly, from (22) and (23) it follows that

$$\frac{\delta_{k+1}}{\lambda_{k+1}} = \frac{\langle \Delta_k, y_k - (1-\alpha_k)x_k\rangle}{\lambda_{k+1}} + \frac{\delta_k}{\lambda_k}, \quad \text{and} \quad \frac{\epsilon_{k+1}}{\lambda_{k+1}} = -\frac{\alpha_k \Delta_k}{\lambda_{k+1}} + \frac{\epsilon_k}{\lambda_k}. \tag{36}$$

Applying (35) and (36) recursively, and using that $B_0 = 0,\ \delta_0 = 0,\ \epsilon_0 = 0,\ V_0(x) = \phi(x_0) + \frac{\gamma_0}{2}\|x - v_0\|^2$ defined in Lemma 4 and (27), inequality (33) follows from (12). Furthermore, taking expectations in (35) and applying (6) and (32), we obtain (34). □

Note that (34) holds for any choices of γ_0, α_k and L_k as long as $L_k > L$, $\alpha_k \in (0,1)$ and $2L_k\alpha_k^2 \leq (1-\alpha_k)\gamma_k + \alpha_k\mu$. In the following two corollaries, we specify values for γ_0, α_k and L_k, depending on whether $\mu = 0$, and derive the desired convergence rates for the expectation of $\phi(x_k) - \phi(x^*)$.

Corollary 1. *Suppose $\mu = 0$ in Algorithm 1 and we choose*

$$L_k = (k+3)^{\frac{3}{2}} + L, \quad \alpha_k = \frac{2}{k+3}, \quad \gamma_0 = 4L + 4(N+3)^{\frac{3}{2}}, \tag{37}$$

where N is the total number of iterations, then we have

$$\mathbb{E}\phi(x_{N+1}) - \phi(x^*) \leq \frac{2(\phi(x_0) - \phi(x^*)) + 4L\|x_0 - x^*\|^2}{(N+2)(N+3)} + \frac{8\|x_0 - x^*\|^2 + \sigma^2}{\sqrt{N+2}}.$$

Proof. Since $\alpha_k = \frac{2}{k+3}$ and $\gamma_{k+1} = (1-\alpha_k)\gamma_k$ in Algorithm 1, we can easily show that $\gamma_{k+1} = \frac{2\gamma_0}{(k+2)(k+3)}$ and $\lambda_{k+1} = \frac{2}{(k+3)(k+2)}$. These guarantee $2L_k\alpha_k^2 = ((k+3)^{\frac{3}{2}} + L)\frac{8}{(k+3)^2} \leq \frac{2\gamma_0}{(k+2)(k+3)} = (1-\alpha_k)\gamma_k$ in each iteration. Therefore, (34) is guaranteed by Theorem 1.

To finish the proof, we first need to bound $\sum_{i=0}^k \frac{\sigma^2}{2(L_i - L)\lambda_{i+1}}$ from above as

$$\sum_{i=0}^k \frac{\sigma^2}{2(L_i - L)\lambda_{i+1}} = \sum_{i=0}^k \frac{\sigma^2}{2(i+3)^{\frac{3}{2}}\frac{2}{(i+3)(i+2)}} \leq \sum_{i=0}^k \frac{\sigma^2(i+3)^{\frac{1}{2}}}{4} \leq \frac{\sigma^2}{4}\int_0^{4+k} \sqrt{x}dx = \frac{\sigma^2}{6}(k+4)^{\frac{3}{2}}. \tag{38}$$

Plugging this inequality into (34) we obtain

$$\begin{aligned}
\mathbb{E}\phi(x_{N+1}) - \phi(x^*) &\leq \frac{2\phi(x_0) - 2\phi(x^*) + \gamma_0\|x_0 - x^*\|^2}{(N+2)(N+3)} + \frac{\sigma^2(N+4)^{\frac{3}{2}}}{3(N+3)(N+2)} \\
&\leq \frac{2\phi(x_0) - 2\phi(x^*) + \gamma_0\|x_0 - x^*\|^2}{(N+2)(N+3)} + \frac{\sigma^2}{\sqrt{N+2}} \\
&= \frac{2(\phi(x_0) - \phi(x^*))}{(N+2)(N+3)} + \frac{(4L + 4(N+3)^{\frac{3}{2}})\|x_0 - x^*\|^2}{(N+2)(N+3)} + \frac{\sigma^2}{\sqrt{N+2}} \\
&\leq \frac{2(\phi(x_0) - \phi(x^*)) + 4L\|x_0 - x^*\|^2}{(N+2)(N+3)} + \frac{8\|x_0 - x^*\|^2}{\sqrt{N+2}} + \frac{\sigma^2}{\sqrt{N+2}} \\
&\leq \frac{2(\phi(x_0) - \phi(x^*)) + 4L\|x_0 - x^*\|^2}{(N+2)(N+3)} + \frac{8\|x_0 - x^*\|^2 + \sigma^2}{\sqrt{N+2}}.
\end{aligned}$$

□

If the smooth component $f(x)$ is strongly convex, our algorithm can attain a better convergence rate, as stated in the following corollary.

Corollary 2. *Suppose $\mu > 0$ in Algorithm 1 and we choose*

$$L_k = \frac{\mu(k+3)^2}{8} + L, \quad \alpha_k = \frac{2}{k+3}, \quad \gamma_0 = 4L + \mu, \tag{39}$$

where N is the total number of iterations, then we have

$$\mathbb{E}\phi(x_{N+1}) - \phi(x^*) \leq \frac{2(\phi(x_0) - \phi(x^*)) + 5L\|x_0 - x^*\|^2}{(N+2)(N+3)} + \frac{4\sigma^2}{\mu(N+2)}.$$

Proof. Since $\alpha_k = \frac{2}{k+3}$ and $\gamma_{k+1} = (1-\alpha_k)\gamma_k + \alpha_k\mu$ in Algorithm 1, we can easily show that $\gamma_{k+1} = \frac{8L}{(k+2)(k+3)} + \mu$ and $\lambda_{k+1} = \frac{2}{(k+3)(k+2)}$. These guarantee $2L_k\alpha_k^2 = \mu + \frac{8L}{(k+3)^2} \leq \gamma_{k+1}$ in each iteration. Therefore, (34) is guaranteed by Theorem 1.

We bound $\sum_{i=0}^{k} \frac{\sigma^2}{2(L_i - L)\lambda_{i+1}}$ from above as

$$\sum_{i=0}^{k} \frac{\sigma^2}{2(L_i - L)\lambda_{i+1}} = \sum_{i=0}^{k} \frac{\sigma^2}{2\frac{\mu(i+3)^2}{8}\frac{2}{(i+3)(i+2)}} \leq \sum_{i=0}^{k} \frac{2\sigma^2}{\mu} = \frac{2(k+1)\sigma^2}{\mu}. \tag{40}$$

Plugging this inequality into (34) we obtain

$$\begin{aligned}
\mathbb{E}\phi(x_{N+1}) - \phi(x^*) &\leq \frac{2\phi(x_0) - 2\phi(x^*) + \gamma_0\|x_0 - x^*\|^2}{(N+2)(N+3)} + \frac{4\sigma^2(N+1)}{\mu(N+3)(N+2)} \\
&\leq \frac{2\phi(x_0) - 2\phi(x^*) + 5L\|x_0 - x^*\|^2}{(N+2)(N+3)} + \frac{4\sigma^2}{\mu(N+2)}.
\end{aligned}$$

To get the second inequality above, we use a straightforward result that $\mu \leq L$, and thus, $\gamma_0 \leq 5L$. □

4 Convergence of the variance of error

Due to the nature of the SSG algorithm, the solution generated is a discrete Markov random sequence in $\mathbf{X}$. Every time we run the algorithm, we could generate a different trajectory of the solutions. Theorem 1, Corollary 1 and Corollary 2 show that, if we run the algorithm multiple times, the error of the solution, i.e. $\phi(x_k) - \phi(x^*)$ converges to zero on average. However, it does not guarantee the accuracy of the solution from each single run of the algorithm. In this section, we show that the variance of the error of the solution, i.e. $\mathbb{V}\left[\phi(x_k) - \phi(x^*)\right]$ also converges to zero. It means the uncertainty of the error will be reduced as the number of the iterations increases. Therefore, the error of the solution will eventually follow a distribution that concentrates close to zero, which makes us trust more the solution from a single run of the algorithm, not just the average.

Theorem 2. *Suppose* (8) *holds and there exists a constant* D *such that* $\|x^* - x_k\| \leq D$ *and* $\|x^* - v_k\| \leq D$ *for* $k \geq 0$. *Then we have*

$$\begin{aligned}\mathbb{V}(\phi(x_{k+1}) - \phi(x^*)) \leq & 3\lambda_{k+1}^2 \left[\phi(x_0) - \phi(x^*) + \frac{\gamma_0}{2}\|x_0 - x^*\|^2\right]^2 \\ & + 3\lambda_{k+1}^2 \left(\sum_{i=0}^{k} \frac{\sigma^2}{2(L_i - L)\lambda_{i+1}}\right)^2 + 3\lambda_{k+1}^2 \sum_{i=0}^{k} \frac{\alpha_i^2 \sigma^2 D^2}{\lambda_{i+1}^2}. \end{aligned} \tag{41}$$

Proof. We will rely on the following elementary consequence of the convexity of the function $t \mapsto t^2$: For $a_1, \ldots, a_n \in \mathbb{R}$ we have

$$(a_1 + \cdots + a_n)^2 \leq n(a_1^2 + \cdots + a_n^2). \tag{42}$$

Since $\mathbb{V}(\phi(x_{k+1}) - \phi(x^*)) \leq \mathbb{E}(\phi(x_{k+1}) - \phi(x^*))^2$, we can derive an upper bound of $\mathbb{V}(\phi(x_{k+1}) - \phi(x^*))$ by bounding $\mathbb{E}(\phi(x_{k+1}) - \phi(x^*))^2$ from above. According to (12) assured by Theorem 1, $\phi(x_{k+1}) - \phi(x^*) \geq 0$ and applying (42) for $n = 3$, we have

$$(\phi(x_{k+1}) - \phi(x^*))^2 \leq 3\lambda_{k+1}^2 \left[\phi(x_0) - \phi(x^*) + \frac{\gamma_0}{2}\|x - v_0\|^2\right]^2 + 3B_{k+1}^2 + 3(\langle \epsilon_{k+1}, x^* \rangle + \delta_{k+1})^2. \tag{43}$$

Applying (35) in the proof of Theorem 1 recursively with $\lambda_0 = 1$ and $B_0 = 0$, we obtain

$$\frac{B_{k+1}}{\lambda_{k+1}} \leq \sum_{i=0}^{k} \frac{\|\Delta_i\|^2}{2(L_i - L)\lambda_{i+1}}$$

Since B_k is nonnegative for all k, if we square and take the expectation of the two sides of this inequality, we get

$$\begin{aligned}\frac{\mathbb{E}B_{k+1}^2}{\lambda_{k+1}^2} &\leq \mathbb{E}\left(\sum_{i=0}^{k}\frac{\|\Delta_i\|^2}{2(L_i-L)\lambda_{i+1}}\right)^2 = \left(\sum_{i=0}^{k}\sum_{j=0}^{k}\frac{\mathbb{E}\left(\|\Delta_i\|^2\|\Delta_j\|^2\right)}{4(L_i-L)(L_j-L)\lambda_{i+1}\lambda_{j+1}}\right) \\ &\leq \left(\sum_{i=0}^{k}\sum_{j=0}^{k}\frac{\sigma^4}{4(L_i-L)(L_j-L)\lambda_{i+1}\lambda_{j+1}}\right) = \left(\sum_{i=0}^{k}\frac{\sigma^2}{2(L_i-L)\lambda_{i+1}}\right)^2, \qquad (44)\end{aligned}$$

where the second inequality is from $\mathbb{E}\left(\|\Delta_i\|^2\|\Delta_j\|^2\right) \leq \sqrt{\mathbb{E}\|\Delta_i\|^4}\sqrt{\mathbb{E}\|\Delta_j\|^4}$ and (8).

According to the recurrence equations (23) and (22), we have

$$\begin{aligned}\frac{\langle\epsilon_{k+1},x^*\rangle+\delta_{k+1}}{\lambda_{k+1}} &= \frac{\langle\epsilon_k,x^*\rangle+\delta_k}{\lambda_k} - \frac{\alpha_k\langle\Delta_k,x^*\rangle - \langle\Delta_k,y_k-(1-\alpha_k)x_k\rangle}{\lambda_{k+1}} \\ &= \frac{\langle\epsilon_k,x^*\rangle+\delta_k}{\lambda_k} - \frac{\alpha_k\langle\Delta_k,x^*-\bar{x}_k\rangle}{\lambda_{k+1}}\end{aligned}$$

for $\bar{x}_k = y_k-(1-\alpha_k)x_k = \frac{\alpha_k\mu x_k+\gamma_k v_k}{\alpha_k\mu+\gamma_k}$; where the last step follows from the updating equations for y_k and γ_k in Algorithm 1. Applying this equation recursively with $\epsilon_0 = 0$ and $\delta_0 = 0$, we get

$$\frac{\langle\epsilon_{k+1},x^*\rangle+\delta_{k+1}}{\lambda_{k+1}} = -\sum_{i=0}^{k}\frac{\alpha_i\langle\Delta_i,x^*-\bar{x}_i\rangle}{\lambda_{i+1}}.$$

Therefore, the expected value $\mathbb{E}\left(\frac{\langle\epsilon_{k+1},x^*\rangle+\delta_{k+1}}{\lambda_{k+1}}\right)^2$ can be reformulated as a sum of $\mathbb{E}\left(\frac{\alpha_i\langle\Delta_i,x^*-\bar{x}_i\rangle}{\lambda_{i+1}}\right)^2$ and $\mathbb{E}\left(\frac{\alpha_i\langle\Delta_i,x^*-\bar{x}_i\rangle}{\lambda_{i+1}}\frac{\alpha_j\langle\Delta_j,x^*-\bar{x}_j\rangle}{\lambda_{j+1}}\right)$ with $i<j$.

Since $\bar{x}_i$ is a convex combination of x_i and v_i, we have $\|x^*-\bar{x}_i\|^2 \leq \frac{\alpha_k\mu}{\alpha_k\mu+\gamma_k}\|x^*-x_i\|^2 + \frac{\gamma_k}{\alpha_k\mu+\gamma_k}\|x^*-v_i\|^2 \leq D$ using the assumptions on x_i and v_i and the convexity of $\|\cdot\|^2$. By Cauchy-Schwarz inequality, it is easy to show that

$$\mathbb{E}\left(\frac{\alpha_i\langle\Delta_i,x^*-\bar{x}_i\rangle}{\lambda_{i+1}}\right)^2 \leq \frac{\alpha_i^2\mathbb{E}(\|\Delta_i\|^2\|x^*-\bar{x}_i\|^2)}{\lambda_{i+1}^2} \leq \frac{\alpha_i^2\sigma^2D^2}{\lambda_{i+1}^2}.$$

Because $\langle\Delta_i,x^*-\bar{x}_i\rangle$ only depends on $\xi_{[i]}$ and $\bar{x}_j$ only depends on $\xi_{[j-1]}$ $(i<j)$, the cross-term

$$\mathbb{E}\left[\langle\Delta_i,x^*-\bar{x}_i\rangle\langle\Delta_j,x^*-\bar{x}_j\rangle\right] = \mathbb{E}_{\xi_{[j-1]}}\left[\mathbb{E}_{\xi_j}(\langle\Delta_i,x^*-\bar{x}_i\rangle\langle\Delta_j,x^*-\bar{x}_j\rangle\,|\xi_{[j-1]})\right] = 0$$

by (29). Therefore, we obtain

$$\mathbb{E}\left(\frac{\langle\epsilon_{k+1},x^*\rangle+\delta_{k+1}}{\lambda_{k+1}}\right)^2 \leq \sum_{i=0}^{k}\frac{\alpha_i^2\sigma^2D^2}{\lambda_{i+1}^2}. \qquad (45)$$

Taking the expectation of (43) and applying (44) and (45), we get

$$\begin{aligned}\mathbb{V}(\phi(x_{k+1})-\phi(x^*)) \leq & \mathbb{E}(\phi(x_{k+1})-\phi(x^*))^2 \\ & \leq 3\lambda_{k+1}^2\left[\phi(x_0)-\phi(x^*)+\frac{\gamma_0}{2}\|x-v_0\|^2\right]^2+3\lambda_{k+1}^2\left(\sum_{i=0}^{k}\frac{\sigma^2}{2(L_i-L)\lambda_{i+1}}\right)^2 \\ & \quad +3\lambda_{k+1}^2\sum_{i=0}^{k}\frac{\alpha_i^2\sigma^2D^2}{\lambda_{i+1}^2}.\end{aligned}$$

□

Similar to the convergence results for $\mathbb{E}(\phi(x_{k+1})-\phi(x^*))$, by choosing different parameters α_k, L_k and γ_0 according to the convexity parameter μ, we can obtain different convergence rates for $\mathbb{V}(\phi(x_{k+1})-\phi(x^*))$. These properties are shown in the following two corollaries of Theorem 2.

Corollary 3. *Under the assumption of Theorem 2, if the strongly convexity parameter $\mu=0$ and we choose L_k, α_k and γ_0 according to* (37) *in Algorithm 1, we have*

$$\mathbb{V}(\phi(x_{N+1})-\phi(x^*)) \leq \frac{36\left[\phi(x_0)-\phi(x^*)\right]^2}{(N+2)^2(N+3)^2}+\frac{144L^2\|x_0-x^*\|^4}{(N+2)^2(N+3)^2}+\frac{288\|x_0-x^*\|^4}{N+2}+\frac{3\sigma^4}{N+3}+\frac{4\sigma^2D^2}{N+2}.$$

Proof. From (38) in the proof of Corollary 1, we have

$$\left(\sum_{i=0}^{k}\frac{\sigma^2}{2(L_i-L)\lambda_{i+1}}\right)^2 \leq \frac{\sigma^4(k+4)^3}{36}. \tag{46}$$

Since $\lambda_{k+1}=\frac{2}{(k+2)(k+3)}$ and $\alpha_k=\frac{2}{k+3}$, it follows that

$$\sum_{i=0}^{k}\frac{\alpha_i^2\sigma^2D^2}{\lambda_{i+1}^2}=\sum_{i=0}^{k}\sigma^2D^2(i+2)^2 \leq \frac{\sigma^2D^2(k+2)(k+3)(2k+5)}{6}. \tag{47}$$

Plugging (46) and (47) into (41) and applying (42) with $n=3$, we get

$$\begin{aligned}&\mathbb{V}(\phi(x_{N+1})-\phi(x^*)) \\ &\leq \frac{12\left[\phi(x_0)-\phi(x^*)+\frac{\gamma_0}{2}\|x_0-x^*\|^2\right]^2}{(N+2)^2(N+3)^2}+\frac{\sigma^4(N+4)^3}{3(N+2)^2(N+3)^2}+\frac{2\sigma^2D^2(N+2)(N+3)(2N+5)}{(N+2)^2(N+3)^2} \\ &\leq \frac{36\left[\phi(x_0)-\phi(x^*)\right]^2}{(N+2)^2(N+3)^2}+\frac{144L^2\|x_0-x^*\|^4}{(N+2)^2(N+3)^2}+\frac{288\|x_0-x^*\|^4}{N+2}+\frac{3\sigma^4}{N+3}+\frac{4\sigma^2D^2}{N+2}.\end{aligned}$$

□

Similar to Corollary 2, if the function $f(x)$ is strongly convex, we have a better convergence

rate for the variance of the error, as stated in the next corollary.

Corollary 4. *Under the assumption of Theorem 2, if the strongly convexity parameter $\mu > 0$ and we choose L_k, α_k and γ_0 according to* (39) *in Algorithm 1, we have*

$$\mathbb{V}(\phi(x_{N+1}) - \phi(x^*)) \leq \frac{24\left[\phi(x_0) - \phi(x^*)\right]^2}{(N+2)^2(N+3)^2} + \frac{150L^2\|x - v_0\|^4}{(N+2)^2(N+3)^2} + \frac{48\sigma^4}{\mu^2(N+3)^2} + \frac{4\sigma^2 D^2}{N+2}.$$

Proof. According to (40) in the proof of Corollary 2, we have

$$\left(\sum_{i=0}^{k} \frac{\sigma^2}{2(L_i - L)\lambda_{i+1}}\right)^2 \leq \frac{4\sigma^4(k+1)^2}{\mu^2}. \tag{48}$$

Recall that, to obtain (47) in Corollary 3, we only use the fact that $\alpha_k = \frac{2}{k+3}$ which is still true in this case. Therefore, (47) remains valid here. Plugging (48) and (47) into (41) applying (42) with $n = 2$, we get

$$\begin{aligned}
&\mathbb{V}(\phi(x_{N+1}) - \phi(x^*)) \\
&\quad \leq \frac{12\left[\phi(x_0) - \phi(x^*) + \frac{\gamma_0}{2}\|x - v_0\|^2\right]^2}{(N+2)^2(N+3)^2} + \frac{48\sigma^4(N+1)^2}{\mu^2(N+2)^2(N+3)^2} + \frac{2\sigma^2 D^2(N+2)(N+3)(2N+5)}{(N+2)^2(N+3)^2} \\
&\quad \leq \frac{24\left[\phi(x_0) - \phi(x^*)\right]^2}{(N+2)^2(N+3)^2} + \frac{150L^2\|x - v_0\|^4}{(N+2)^2(N+3)^2} + \frac{48\sigma^4}{\mu^2(N+3)^2} + \frac{4\sigma^2 D^2}{N+2}.
\end{aligned}$$

□

Unlike the situation with the expected error, the convergence rates for the variances of the error are asymptotically $O(\frac{1}{N})$ regardless of the strong convexity of $f(x)$.

5 Probability of a large error

Besides the convergences of the expectation and variance of the solution, another way to evaluate a stochastic gradient algorithm is the confidence level of its solution. Regarding this, in this section, we analyze the probability of a large error for our SSG algorithm by showing the result as

$$\text{Prob}\left[\phi(x_{N+1}) - \phi(x^*) \geq \epsilon(N, \delta)\right] \leq \delta, \tag{49}$$

where $\delta \in (0, 1)$ and $\epsilon(N, \delta)$ is decreasing with both N and δ.

If we apply Markov's inequality to the conclusion of Corollary 1 and Corollary 2, we conclude that (49) holds with

$$\epsilon(N, \delta) = \frac{2(\phi(x_0) - \phi(x^*)) + 4L\|x_0 - x^*\|^2}{\delta(N+2)(N+3)} + \frac{8\|x_0 - x^*\|^2 + \sigma^2}{\delta\sqrt{N+2}} \tag{50}$$

and

$$\epsilon(N,\delta) = \frac{2(\phi(x_0)-\phi(x^*))+5L\|x_0-x^*\|^2}{\delta(N+2)(N+3)} + \frac{4\sigma^2}{\delta\mu(N+2)} \tag{51}$$

under different choices of parameters in the algorithm respectively.

According to the one-sided version of the Chebyshev's inequality (also called Cantelli's inequality), we have

$$\text{Prob}\left[\phi(x_{N+1})-\phi(x^*) \geq \mathbb{E}[\phi(x_{N+1})-\phi(x^*)] + \sqrt{\mathbb{V}[\phi(x_{N+1})-\phi(x^*)]/\delta}\right] \leq \delta. \tag{52}$$

Therefore, assuming (8), we can obtain (49) with $\epsilon(N,\delta) = O\left(\frac{1}{\sqrt{\delta N}}\right)$ for our algorithm by substituting $\mathbb{E}[\phi(x_{N+1})-\phi(x^*)]$ and $\mathbb{V}[\phi(x_{N+1})-\phi(x^*)]$ in (52) with the upper bounds given in Corollary 1 and Corollary 3 (or Corollary 2 and Corollary 4, since strongly convexity of $f(x)$ does not improve the bound on the variance anyway). So far we have only described $\epsilon(N,\delta)$ in asymptotic rate for ease of notation.

By strengthening (8) to (9), an $\epsilon(N,\delta) = O\left(\frac{\ln(1/\delta)}{\sqrt{N}}\right)$ is obtained in [2, 4] regardless of whether $f(x)$ is strongly convex. We note that the same rate of $\epsilon(N,\delta)$ can be obtained for our algorithm using a proof similar to [2, 4].

Next we show that, an $\epsilon(N,\delta) = O\left(\sqrt{\frac{\ln(1/\delta)}{N}}\right)$ is available for our algorithm under an assumption somewhat stronger than (9) (and thus stronger than (6) and (8)). This result is given by the following theorem and its corollaries. Part of techniques used in the proof are modifications of the techniques used in [20] for proving similar results.

Theorem 3. *If the stochastic gradient $G(x,\xi)$ satisfies $\|G(x,\xi)-\nabla f(x)\| \leq \sigma$ for all $(x,\xi) \in \mathbb{R}^n \times \Xi$ and there exists a constant D such that $\|x^*-x_k\| \leq D$ and $\|x^*-v_k\| \leq D$ for $k \geq 0$, then for any $\delta \in (0,1)$, (49) holds with*

$$\begin{aligned}\epsilon(N,\delta) &= \lambda_{N+1}\left[\phi(x_0)-\phi(x^*)+\frac{\gamma_0}{2}\|x_0-v_0\|^2\right] \\ &\quad +\lambda_{N+1}\left[\sum_{i=0}^{N}\frac{\sigma^2}{2(L_i-L)\lambda_{i+1}}\right] + 3\sigma D\lambda_{N+1}\sqrt{2\sum_{i=0}^{N}\frac{\alpha_i^2}{\lambda_{i+1}^2}\ln\frac{1}{\delta}}.\end{aligned} \tag{53}$$

Proof. By the assumption, $\|\Delta_i\| \leq \sigma$. We claim the sequence

$$m_k := \sum_{i=0}^{k-1}\frac{\langle \Delta_i, \alpha_i x^* + (1-\alpha_i)x_i - y_i\rangle}{\lambda_{i+1}} \quad k \geq 1 \tag{54}$$

appearing in (33) is a discrete martingale relative to the filtration $\mathcal{F}_k := \sigma(\{\xi_i : i = 0,\ldots,k-1\})$. To see this, first observe that $\mathbb{E}[m_{k+1}|\mathcal{F}_k] = m_k + \frac{\mathbb{E}[\langle \Delta_k, \alpha_k x^* + (1-\alpha_k)x_k - y_k\rangle|\mathcal{F}_k]}{\lambda_{k+1}}$. Hence it suffices to

show that the second term is zero. Indeed,

$$\begin{aligned}\mathbb{E}\left[\langle \Delta_k, \alpha_k x^* + (1-\alpha_k)x_k - y_k\rangle \,|\mathcal{F}_k\right] &= \langle \mathbb{E}[\Delta_k|\mathcal{F}_k], \alpha_k x^* + (1-\alpha_k)x_k - y_k\rangle \\ &= \langle 0, \alpha_k x^* + (1-\alpha_k)x_k - y_k\rangle \\ &= 0.\end{aligned}$$

The first equality holds because x_k and y_k are measurable with respect to $\mathcal{F}_k$. The second equality holds because ξ_k is independent of $\mathcal{F}_k$ so that $\mathbb{E}[\Delta_k|\mathcal{F}_k] = \mathbb{E}[G(y_k,\xi_k) - \nabla f(y_k)|\mathcal{F}_k] = 0$ by (5). Therefore, $\mathbb{E}\left[m_{k+1}|\mathcal{F}_k\right] = m_k$ and thus (54) is a discrete martingale. Furthermore, by the above assumptions in statement of the theorem and the updating equation of y_k, we have

$$\|\alpha_k x^* + (1-\alpha_k)x_k - y_k\| \le \alpha_k\|x^* - x_k\| + \frac{\alpha_k\gamma_k\|x_k - v_k\|}{\gamma_k + \alpha_k\mu} \le \alpha_k D + \alpha_k\|x_k - v_k\| \le 3\alpha_k D$$

so that the absolute martingale difference can be bounded as

$$|m_{k+1} - m_k| \le \left|\frac{\langle \Delta_k, \alpha_k x^* + (1-\alpha_k)x_k - y_k\rangle}{\lambda_{k+1}}\right| \le \frac{3\alpha_k\sigma D}{\lambda_{k+1}}.$$

Therefore, by the Azuma-Hoeffding inequality [19], we have

$$\text{Prob}(\lambda_k m_k \ge c) \le \exp\left(\frac{-c^2}{18(\sigma D)^2\lambda_k^2\sum_{i=0}^{k-1}\alpha_i^2/\lambda_{i+1}^2}\right). \tag{55}$$

Using $\|\Delta_i\| \le \sigma$, (33) and (55), it follows that

$$\begin{aligned}\text{Prob}\left(\phi(x_k) - \phi(x^*) \ge \lambda_k\left[\phi(x_0) - \phi(x^*) + \frac{\gamma_0}{2}\|x_0 - v_0\|^2\right] + \lambda_k\left[\sum_{i=0}^{k-1}\frac{\sigma^2}{2(L_i - L)\lambda_{i+1}}\right] + c\right) \\ \le \exp\left(\frac{-c^2}{18(\sigma D\lambda_k)^2\sum_{i=0}^{k-1}\alpha_i^2/\lambda_{i+1}^2}\right).\end{aligned}$$

The conclusion of the theorem is obtained by choosing $c = 3\sigma D\lambda_k\sqrt{2\sum_{i=0}^{k-1}\alpha_i^2/\lambda_{i+1}^2\ln\frac{1}{\delta}}$. □

The following corollaries are derived by choosing specific values for L_k, α_k and γ_0.

Corollary 5. *Under the assumption of Theorem 3, if the strongly convexity parameter $\mu = 0$ and we choose L_k, α_k and γ_0 according to* (37) *in Algorithm 1, then* (49) *holds with*

$$\epsilon(N,\delta) = \frac{2(\phi(x_0) - \phi(x^*)) + 4L\|x_0 - x^*\|^2}{(N+2)(N+3)} + \frac{8\|x_0 - x^*\|^2 + \sigma^2}{\sqrt{N+2}} + \frac{6\sigma D\sqrt{\ln(1/\delta)}}{\sqrt{N+2}}.$$

Proof. We can upper bound the first two term in (53) as we did in Corollary 1. The third term in

(53) is upper bounded by $\frac{6\sigma D\sqrt{\ln(1/\delta)}}{\sqrt{N+2}}$ because $\lambda_i = \frac{2}{(i+1)(i+2)}$, $\alpha_i = \frac{2}{i+3}$ and

$$\lambda_{N+1}\sqrt{2\sum_{i=0}^{N}\alpha_i^2/\lambda_{i+1}^2} = \frac{2}{(N+2)(N+3)}\sqrt{2\sum_{i=0}^{N}(i+2)^2} \leq \frac{2}{\sqrt{N+2}}. \tag{56}$$

Then we just apply Theorem 3. □

Corollary 6. *Under the assumption of Theorem 3, if the strongly convexity parameter $\mu > 0$ and we choose L_k, α_k and γ_0 according to (39) in Algorithm 1, then (49) holds with*

$$\epsilon(N,\delta) \quad = \quad \frac{2\phi(x_0) - 2\phi(x^*) + 5L\|x_0 - x^*\|^2}{(N+2)(N+3)} + \frac{4\sigma^2}{\mu(N+2)} + \frac{6\sigma D\sqrt{\ln(1/\delta)}}{\sqrt{N+2}}.$$

Proof. We can upper bound the first two term in (53) as we did in Corollary 2 and the third term in (53) is still upper bounded by $\frac{6\sigma D\sqrt{\ln(1/\delta)}}{\sqrt{N+2}}$ as Corollary 5. Then we just apply Theorem 3. □

According to Corollary 5 and Corollary 6, $\epsilon(N,\delta) = O\left(\sqrt{\frac{\ln(1/\delta)}{N}}\right)$ asymptotically regardless of whether $f(x)$ is strongly convex. This rate is asymptotically better than the rate i.e, $O(\frac{1}{\delta\sqrt{N}})$ in (50), and the rate $O(\frac{1}{\delta N})$ in (51). The same rate applies to the RDA algorithm. (See [20] for details.)

6 Numerical experiments

In this section, we present the results of our computational experiments. We compare our SSG algorithm with two other stochastic gradient approaches, AC-SA[4, 2] and RDA[20], as well as a batch learning approach. We test these algorithms on the following problem

$$\min_{x\in\mathbb{R}^p} \frac{\mathbb{E}_{a,b}[(a^Tx - b)^2]}{2} + \lambda\rho\|x\|_1 + \frac{(1-\rho)\|x\|_2^2}{2}, \tag{57}$$

where p is the dimension of the problem and the parameters $\lambda > 0$ and $0 \leq \rho \leq 1$ are chosen by users. When $\rho = 1$, (57) is the l_1 regularized linear regression problem (i.e., lasso [15]). When $0 < \rho < 1$, (57) becomes the elastic net regression problem [22]. Both of these two problems are of the form (1). We notice that the smooth component $f(x)$ in this problem is $\frac{\mathbb{E}_{a,b}[(a^Tx-b)^2]}{2} + \frac{(1-\rho)\|x\|_2^2}{2}$, which is strongly convex with a strong convexity parameter $\mu = (1-\rho)$ if $0 \leq \rho < 1$. We implement our algorithms using MATLAB R2010a on a machine with 2.67 GHz CPU and 12Gb RAM.

In our experiments, we set a to be uniformly distributed in the hypercube $[0,1]^p$ and choose $b = a^T\bar{x} + e$. Here, $\bar{x}$ is the true coefficients defining the linear relationship between a and b. We choose $\bar{x}_i = 10$ for $1 \leq i \leq p/2$ and $\bar{x}_i = 0$ for $p/2 + 1 \leq i \leq p$. The variable e is a random noise

which is independent of a and follows a normal distribution $N(0, \sigma_e^2)$. In the following experiments, the dimension p is chosen to be either 20 or 100.

Suppose in each iteration, there are m data points $\xi = \{(a_i, b_i)\}_{1 \leq i \leq m}$ generated from the distribution of (a, b) given above. The stochastic gradient $G(x, \xi)$ at point x can be chosen to be

$$G(x, \xi) = \frac{1}{m} \sum_{i=1}^{m} (a_i^T x - b_i) a_i + (1 - \rho) x, \tag{58}$$

which is an empirical estimator of the true gradient $\nabla f(x) = \mathbb{E}_{a,b}[(a^T x - b)a] + (1-\rho)x$ for (57). As m increases, $G(x, \xi)$ converges to $\nabla f(x)$. The accuracy of this estimator is determined by m and σ_e together. We choose different values for m (10, 100, 1000) and σ_e^2 (1, 100) in our experiments in order to test the performance of the algorithms when the first-order information has different levels of accuracy.

We also compare our method with the batch learning approach. To do this, we generate another set of K data points $\{(\widehat{a}_i, \widehat{b}_i)\}_{1 \leq i \leq K}$ from the distribution of (a, b). Then we apply the deterministic accelerated gradient algorithm, FISTA [1], to solve the batch learning version of (57), which is formulated as

$$\min_{x \in \mathbb{R}^p} \frac{1}{2K} \sum_{i=1}^{K} (\widehat{a}_i^T x - \widehat{b}_i)^2 + \lambda \rho \|x\|_1 + \frac{(1-\rho)\|x\|_2^2}{2}, \tag{59}$$

where the first term in the objective function is just an empirical estimator of $\frac{\mathbb{E}_{a,b}[(a^T x - b)^2]}{2}$. In order to make the comparisons fair enough, all approaches should utilize the same amount of data points. When the stochastic gradient methods run for N iterations and use m data points in each iterations, they actually access to Nm different data points. Therefore, we choose K to be Nm which will change if either m or N changes.

For all of the tested algorithms, we choose an all-zero vector as the initial iterate x_0 and each algorithm will be stopped after $N = 2000$ iterations. Then we compare the objective values achieved, the CPU time and the sparsity of the solutions. The expectation $\mathbb{E}_{a,b}[(a^T x - b)^2]$ is usually hard to compute explicitly for real problems, which prevent us from evaluating the objective values returned by the algorithms. Fortunately, under the specific setting in our example, we can derive a closed form expression for $\mathbb{E}_{a,b}[(a^T x - b)^2]$, which is $(x - \bar{x})^T Q (x - \bar{x}) + \sigma_e^2$. Here, Q is a p by p matrix with $Q_{ii} = \frac{1}{3}$ and $Q_{ij} = \frac{1}{4}$ if $i \neq j$. We use this formula to evaluate the objective values of (57) in all of our experiments. To compare the sparse patterns of the solutions, we compute the exact density ($d_x := \frac{|\{i | x_i \neq 0\}|}{p}$) and truncated density ($td_x := \frac{|\{i | |x_i| > \epsilon_r\}|}{p}$), where ϵ_r is the truncating threshold chosen to be 10^{-5}.

Due to the stochastic nature of these algorithms, it is more reasonable to compare the performances on average of several runs. Hence, we run each algorithm twenty times with the same

initialization. We report the results of our experiments in Table 1 to Table 3. The choice of parameters p, σ_e^2, ρ, λ, N and m are indicated in the tables. We note that the density is very sensitive to the choice of λ. Therefore, we carefully tuned λ to avoid the extreme cases (e.g. all zero or all non-zero solutions) and achieve different levels of density. From the first to the forth column in the table are the minimum, maximum, mean and variance of the final objective values in the twenty repeated experiments. The fifth column is the average CPU time used in seconds. The sixth and seventh columns are the mean of d_x and td_x respectively.

There are some details about the implementations of these algorithms which we also want to indicate here. Both SSG and AC-SA assume we know the Lipschitz constant L of $f(x)$. SSG also assumes we know the strong convexity parameter μ of $f(x)$. For problem (57), L is the largest eigenvalue of Q plus $1-\rho$ while μ is simply $1-\rho$.

According to Proposition 4 in [2], users have to specified a parameter γ when AC-SA is applied to a general convex problem. In order for AC-SA to achieve the optimal rate of convergence, the parameter γ has to be equal to $\gamma_N^* = \max\left\{2L, \left[\frac{2\sigma^2 N(N+1)(N+2)}{3\|x^*\|^2}\right]^{\frac{1}{2}}\right\}$ according to [2]. Hence, we first apply the batch learning approach on (57). Then we approximate x^* using the solution found by the batch learning approach. To evaluate σ^2, we generate ten stochastic gradients on the point x^* using the formula (58). Then we approximate σ^2 by taking the average of squares of the differences between these ten stochastic gradients and $\nabla f(x^*) = Q(x^* - \bar{x})$. After these, we will have a good estimation for γ_N^*.

RDA computes a value $\beta_k = \gamma\sqrt{k}$ in each iteration when it is applied to a general convex problem. According to [20], the best γ can be set as G/D, where G is an upper bound on the norm of the gradients and D is roughly the norm of the optimal x^*. We still approximate x^* with the solution of the batch learning approach and use its norm as D. After that, we randomly generate ten points uniformly in the hypercube $[0, 10]^p$ and use the largest norm of (58) on these ten points as the estimation of G. When RDA is applied to a strongly convex problem, [20] suggests more than one way to choose β_k. The way we follow here is setting $\beta_k = 1 - \rho$.

As we indicate in Section 1, RDA generates a sparse vector x_k by solving the composite gradient mapping. But it is not proved to converge to the optimality in theory. What converges is the average of $x_1, x_2, \ldots, x_k$, denoted by $\bar{x}_k$. According to the experiments in [20] and our own testings, x_k is better than $\bar{x}_k$ in recovering the sparse pattern in the solutions. However, we only focus on comparing the algorithms which are guaranteed to converge. Hence, the objective values, d_x and td_x for RDA algorithm are evaluated on $\bar{x}_k$ instead of x_k.

The phenomena revealed by the results in these numerical experiments can be summarized as follows:

1. The objective values found by the batch learning approach are generally smaller than those found by the stochastic methods. This is not surprising since the batch learning approach uses the whole data in each iteration and it is solved by FISTA which has a better convergence

Table 1: Numerical results on l_1 regularized linear regression problem

$\sigma_e^2 = 1$		p=20, N=2000, m=10						
		min	max	mean	var	time(s)	d_x	td_x
$\rho = 1$ $\lambda = 20$	SSG	1226.59	1227.60	1227.01	0.061382	0.12	0.97	0.95
	ACSA	1227.86	1228.69	1228.32	0.039345	0.13	1.00	1.00
	RDA	1230.78	1231.93	1231.27	0.100096	0.14	1.00	1.00
	Batch	1226.51	1227.01	1226.72	0.018617	0.62	0.49	0.49
$\rho = 1$ $\lambda = 25$	SSG	1290.85	1291.04	1290.91	0.003811	0.12	0.93	0.90
	ACSA	1291.12	1291.57	1291.27	0.010468	0.13	1.00	1.00
	RDA	1290.84	1291.00	1290.88	0.001613	0.14	0.94	0.88
	Batch	1290.86	1291.10	1290.95	0.004877	0.63	0.22	0.22
$\sigma_e^2 = 1$		p=20, N=2000, m=100						
		min	max	mean	var	time(s)	d_x	td_x
$\rho = 1$ $\lambda = 20$	SSG	1226.40	1226.69	1226.57	0.004910	0.20	0.93	0.82
	ACSA	1227.85	1228.00	1227.93	0.001814	0.19	1.00	1.00
	RDA	1230.89	1231.65	1231.38	0.038025	0.20	1.00	1.00
	Batch	1226.31	1226.39	1226.35	0.000394	14.6	0.50	0.50
$\rho = 1$ $\lambda = 25$	SSG	1290.82	1290.84	1290.82	3.58E-05	0.20	0.74	0.54
	ACSA	1290.84	1290.87	1290.85	5.28E-05	0.20	1.00	1.00
	RDA	1290.82	1290.85	1290.84	5.01E-05	0.20	0.94	0.75
	Batch	1290.82	1290.87	1290.85	0.000135	14.6	0.41	0.41
$\sigma_e^2 = 1$		p=20, N=2000, m=1000						
		min	max	mean	var	time(s)	d_x	td_x
$\rho = 1$ $\lambda = 20$	SSG	1226.42	1226.53	1226.47	0.000819	1.31	0.81	0.77
	ACSA	1227.87	1227.91	1227.89	0.000137	1.28	1.00	1.00
	RDA	1230.91	1231.60	1231.33	0.031766	1.28	1.00	1.00
	Batch	1226.30	1226.31	1226.31	2.74E-06	160	0.50	0.50
$\rho = 1$ $\lambda = 25$	SSG	1290.82	1290.83	1290.82	6.99E-06	1.25	0.53	0.50
	ACSA	1290.82	1290.83	1290.82	5.69E-06	1.22	1.00	0.52
	RDA	1290.83	1290.84	1290.84	1.54E-05	1.18	0.88	0.55
	Batch	1290.82	1290.83	1290.82	5.07E-06	161	0.49	0.49

rate than any stochastic methods. However, the differences are tiny in general, especially when the problem (57) is strongly convex ($\rho = 0.5$) for which the convergence rates of the stochastic first-methods are further improved by Corollary 1 and 2.

2. Under the same conditions, the solutions found by SSG are more sparse (having a smaller d_x and td_x) than the ones found by AC-SA and RDA. This is because SSG is a sparsity-preserving algorithm while AC-SA and RDA are not as we mentioned in Section 1. As m increases and thus σ^2 is reduced, the level of sparsity found by SSG are improved. This is because the convergence rates of SSG are better if σ^2 is smaller (Corollary 1 and 2) so that the returned solutions are closer to the optimal ones. In most of the instances with $m = 1000$,

Table 2: Numerical results on elastic net regression problem

$\sigma_e^2 = 1$		p=20, N=2000, m=10						
		min	max	mean	var	time(s)	d_x	td_x
$\rho = 0.5$ $\lambda = 50$	SSG	1292.04	1292.05	1292.04	5.62E-06	0.09	0.99	0.92
	ACSA	1292.04	1292.05	1292.04	3.57E-06	0.10	1.00	1.00
	RDA	1292.04	1292.05	1292.04	1.29E-05	0.10	0.98	0.89
	Batch	1292.04	1292.04	1292.04	1.37E-06	0.61	0.63	0.63
$\rho = 0.5$ $\lambda = 51$	SSG	1292.14	1292.15	1292.15	4.27E-06	0.10	0.96	0.62
	ACSA	1292.15	1292.15	1292.15	9.13E-07	0.10	1.00	1.00
	RDA	1292.14	1292.15	1292.15	4.28E-06	0.11	0.93	0.77
	Batch	1292.14	1292.15	1292.14	2.53E-06	0.62	0.49	0.49
$\sigma_e^2 = 1$		p=20, N=2000, m=100						
		min	max	mean	var	time(s)	d_x	td_x
$\rho = 0.5$ $\lambda = 50$	SSG	1292.04	1292.04	1292.04	1.68E-08	0.18	0.99	0.72
	ACSA	1292.04	1292.04	1292.04	1.29E-08	0.18	1.00	1.00
	RDA	1292.04	1292.04	1292.04	8.37E-08	0.18	0.95	0.79
	Batch	1292.04	1292.04	1292.04	1.48E-08	14.6	0.50	0.50
$\rho = 0.5$ $\lambda = 51$	SSG	1292.14	1292.14	1292.14	2.69E-08	0.18	0.84	0.50
	ACSA	1292.14	1292.14	1292.14	3.25E-08	0.19	1.00	0.89
	RDA	1292.14	1292.14	1292.14	7.26E-08	0.18	0.81	0.54
	Batch	1292.14	1292.14	1292.14	1.10E-08	14.6	0.50	0.50
$\sigma_e^2 = 1$		p=20, N=2000, m=1000						
		min	max	mean	var	time(s)	d_x	td_x
$\rho = 0.5$ $\lambda = 50$	SSG	1292.04	1292.04	1292.04	1.44E-10	1.08	0.94	0.50
	ACSA	1292.04	1292.04	1292.04	2.37E-10	1.06	1.00	0.65
	RDA	1292.04	1292.04	1292.04	8.17E-10	1.07	0.91	0.54
	FISTA	1292.04	1292.04	1292.04	1.77E-10	159	0.50	0.50
$\rho = 0.5$ $\lambda = 51$	SSG	1292.14	1292.14	1292.14	1.30E-10	1.24	0.60	0.50
	ACSA	1292.14	1292.14	1292.14	2.09E-10	1.25	1.00	0.50
	RDA	1292.14	1292.14	1292.14	8.75E-10	1.24	0.58	0.50
	Batch	1292.14	1292.14	1292.14	2.16E-10	160	0.50	0.50

SSG recovers the same level of sparsity as $\bar{x}$, i.e, 50%.

3. In general, the batch learning approach finds sparser solutions than the three stochastic methods. With an appropriate choice of λ, it can almost recover the 50% sparsity level of $\bar{x}$ for all choices of m. There are two mains reasons for this. First, the batch learning approach utilizes FISTA which is a sparsity-preserving algorithm. Second, the batch learning approach uses deterministic gradients so that the solutions it finds are almost idetical to the solutions of (59) which are sparse. In contrast, the stochastic methods utilize the stochastic gradients such that the solutions they find will be, more or less, perturbed away from the desired sparse solutions. The perturbations are relatively small and converge to zero eventually, which have

Table 3: Numerical results on l_1 regularized linear regression problem

$\sigma_e^2 = 100$		p=20, N=2000, m=10						
		min	max	mean	var	time(s)	d_x	td_x
$\rho = 1$ $\lambda = 20$	SSG	1276.17	1277.20	1276.53	0.067564	0.13	0.97	0.95
	ACSA	1277.44	1278.25	1277.90	0.039824	0.13	1.00	1.00
	RDA	1280.29	1281.60	1280.85	0.118222	0.14	1.00	1.00
	Batch	1276.02	1276.44	1276.20	0.021279	0.60	0.49	0.49
$\rho = 1$ $\lambda = 25$	SSG	1340.35	1340.56	1340.41	0.004323	0.12	0.90	0.82
	ACSA	1340.65	1341.09	1340.81	0.015765	0.13	1.00	1.00
	RDA	1340.34	1340.45	1340.38	0.001179	0.14	0.95	0.88
	Batch	1340.37	1340.58	1340.46	0.004171	0.60	0.22	0.22
$\sigma_e^2 = 100$		p=20, N=2000, m=100						
		min	max	mean	var	time(s)	d_x	td_x
$\rho = 1$ $\lambda = 20$	SSG	1275.95	1276.30	1276.10	0.005990	0.21	0.92	0.85
	ACSA	1277.27	1277.54	1277.42	0.003864	0.20	1.00	1.00
	RDA	1280.22	1281.13	1280.78	0.060141	0.20	1.00	1.00
	Batch	1275.82	1275.85	1275.83	9.40E-05	14.5	0.50	0.50
$\rho = 1$ $\lambda = 25$	SSG	1340.32	1340.35	1340.33	7.43E-05	0.19	0.76	0.54
	ACSA	1340.35	1340.37	1340.36	3.98E-05	0.19	1.00	1.00
	RDA	1340.33	1340.36	1340.34	0.000152	0.20	0.92	0.75
	Batch	1340.34	1340.38	1340.35	0.000122	14.6	0.40	0.40
$\sigma_e^2 = 100$		p=20, N=2000, m=1000						
		min	max	mean	var	time(s)	d_x	td_x
$\rho = 1$ $\lambda = 20$	SSG	1275.94	1276.00	1275.97	0.000326	1.14	0.86	0.77
	ACSA	1277.36	1277.42	1277.39	0.000249	1.13	1.00	1.00
	RDA	1280.50	1281.15	1280.87	0.029098	1.14	1.00	1.00
	Batch	1275.80	1275.81	1275.81	2.20E-06	160	0.50	0.50
$\rho = 1$ $\lambda = 25$	SSG	1340.32	1340.33	1340.32	4.46E-06	1.14	0.53	0.50
	ACSA	1340.32	1340.34	1340.32	2.28E-05	1.13	1.00	0.69
	RDA	1340.33	1340.35	1340.34	1.56E-05	1.12	0.87	0.54
	Batch	1340.32	1340.33	1340.32	5.68E-06	160	0.49	0.49

been reflected by the small gaps between the objective values found by the batch learing approach and the stochastic methods. However, even a tiny pertubation can still rule out some zeros components in the returned solutions without changing the objetive value too much.

4. As m increases with other parameters fixed, the final objective values decrease in all of the tested algorithms, although not significantly. This happens for different reasons in the batch learning approach and the stochastic methods. In stochastic methods, a larger m corresponds to a smaller σ^2, which further gives a better convergence rate according to Corollary 1 and 2. In the deterministic methods, we are actually solving (59) whose solutions converges to the

Table 4: Numerical results on elastic net regression problem

$\sigma_e^2 = 100$		p=20, N=2000, m=10						
		min	max	mean	var	time(s)	d_x	td_x
$\rho = 0.5$ $\lambda = 50$	SSG	1341.54	1341.56	1341.54	2.45E-05	0.11	1.00	0.94
	ACSA	1341.54	1341.55	1341.54	5.09E-06	0.12	1.00	1.00
	RDA	1341.54	1341.55	1341.54	9.07E-06	0.11	0.98	0.93
	Batch	1341.54	1341.55	1341.54	2.98E-06	0.61	0.62	0.61
$\rho = 0.5$ $\lambda = 51$	SSG	1341.64	1341.65	1341.65	2.01E-06	0.10	0.96	0.67
	ACSA	1341.65	1341.66	1341.65	5.43E-06	0.10	1.00	1.00
	RDA	1341.64	1341.65	1341.65	6.59E-06	0.10	0.93	0.79
	Batch	1341.64	1341.65	1341.65	3.57E-06	0.60	0.49	0.49
$\sigma_e^2 = 100$		p=20, N=2000, m=100						
		min	max	mean	var	time(s)	d_x	td_x
$\rho = 0.5$ $\lambda = 50$	SSG	1341.54	1341.54	1341.54	8.72E-08	0.18	0.99	0.71
	ACSA	1341.54	1341.54	1341.54	5.47E-08	0.18	1.00	1.00
	RDA	1341.54	1341.54	1341.54	1.00E-07	0.18	0.97	0.75
	Batch	1341.54	1341.54	1341.54	5.84E-08	14.3	0.51	0.51
$\rho = 0.5$ $\lambda = 51$	SSG	1341.64	1341.64	1341.64	4.89E-08	0.18	0.85	0.50
	ACSA	1341.64	1341.64	1341.64	2.02E-08	0.19	1.00	0.99
	RDA	1341.64	1341.64	1341.64	1.90E-07	0.19	0.83	0.55
	Batch	1341.64	1341.64	1341.64	3.58E-08	14.1	0.50	0.50
$\sigma_e^2 = 100$		p=20, N=2000, m=1000						
		min	max	mean	var	time(s)	d_x	td_x
$\rho = 0.5$ $\lambda = 50$	SSG	1341.54	1341.54	1341.54	2.12E-10	1.17	0.93	0.50
	ACSA	1341.54	1341.54	1341.54	7.19E-10	1.17	1.00	0.79
	RDA	1341.54	1341.54	1341.54	3.39E-10	1.21	0.92	0.53
	Batch	1341.54	1341.54	1341.54	8.14E-10	159	0.50	0.50
$\rho = 0.5$ $\lambda = 51$	SSG	1341.64	1341.64	1341.64	2.21E-10	1.24	0.59	0.50
	ACSA	1341.64	1341.64	1341.64	9.44E-10	1.21	1.00	0.50
	RDA	1341.64	1341.64	1341.64	1.09E-09	1.24	0.59	0.50
	Batch	1341.64	1341.64	1341.64	8.60E-11	159	0.50	0.50

solutions of (57) when m, and thus, $K = Nm$ increases.

5. The CPU times spent by the three stochastic gradient methods are nearly the same because they all have to solve a similar composite gradient mapping and apply some similar updates in each iteration. However, the deterministic batch learning method is much slower than the stochastic ones and the differences become more and more significant as the size of the data set, i.e. $K = Nm$, increases. Given the small gaps in the final objective values, it is not always worth the extra cost of the batch learning approach for huge problems. Moreover, when the size of the data is extremely large, the batch learning approach may not even be applicable due to memory limitations. By contrast, one can still applied the stochastic methods using a

Table 5: Numerical results on l_1 regularized linear regression problem

$\sigma_e^2 = 100$		p=100, N=2000, m=10						
		min	max	mean	var	time(s)	d_x	td_x
$\rho = 1$ $\lambda = 120$	SSG	31444.5	31455.6	31449.1	6.712604	0.31	0.96	0.95
	ACSA	31482.1	31500.8	31488.5	22.53959	0.31	1.00	1.00
	RDA	31446.5	31448.2	31447.2	0.255176	0.33	1.00	1.00
	Batch	31443.2	31451.9	31446.4	3.832635	5.69	0.08	0.08
$\rho = 1$ $\lambda = 125$	SSG	31510.7	31519.2	31514.6	4.801911	0.30	0.93	0.91
	ACSA	31576.0	31597.0	31584.4	30.20217	0.29	1.00	1.00
	RDA	31507.2	31507.6	31507.4	0.010281	0.30	0.97	0.90
	Batch	31507.7	31512.4	31509.0	1.112550	5.79	0.03	0.03
$\sigma_e^2 = 100$		p=100, N=2000, m=100						
		min	max	mean	var	time(s)	d_x	td_x
$\rho = 1$ $\lambda = 120$	SSG	31441.1	31442.1	31441.5	0.068742	0.80	0.95	0.92
	ACSA	31446.7	31447.8	31447.3	0.084048	0.79	1.00	1.00
	RDA	31446.6	31447.3	31447.0	0.031878	0.81	1.00	1.00
	Batch	31441.6	31443.2	31442.2	0.162158	59.8	0.19	0.19
$\rho = 1$ $\lambda = 125$	SSG	31507.1	31508.1	31507.5	0.084862	0.77	0.91	0.86
	ACSA	31515.2	31516.7	31515.8	0.130403	0.77	1.00	1.00
	RDA	31507.0	31507.1	31507.0	0.000944	0.78	0.96	0.84
	Batch	31507.1	31507.4	31507.2	0.013452	60.1	0.05	0.05
$\sigma_e^2 = 100$		p=100, N=2000, m=1000						
		min	max	mean	var	time(s)	d_x	td_x
$\rho = 1$ $\lambda = 120$	SSG	31440.7	31440.9	31440.8	0.002513	5.56	0.80	0.62
	ACSA	31441.6	31441.7	31441.6	0.000796	5.12	1.00	1.00
	RDA	31446.7	31447.2	31446.9	0.013792	5.30	1.00	1.00
	Batch	31440.9	31441.3	31441.1	0.008126	660	0.37	0.37
$\rho = 1$ $\lambda = 125$	SSG	31506.9	31507.0	31506.9	0.000221	5.56	0.82	0.62
	ACSA	31507.6	31507.9	31507.7	0.003671	5.46	1.00	1.00
	RDA	31506.9	31507.0	31506.9	7.53E-05	5.40	0.93	0.66
	Batch	31506.9	31507.0	31507.0	0.000216	661	0.11	0.11

small portion of the data in each iteration.

6. The variances of the final objective value are very small in all of our experiments, especially when the problem is strongly convex ($\rho = 0.5$), and the variances decrease sas m increases. The theoretical results proved in this paper can at least explain this phenomenon for SSG algorithm and we expect there are similar results for RDA and AC-SA. According to Corollary 3 and 4, the variance of the objective values found by SSG will converge to zeros and convergence rate is slightly better when the problem is strongly convex. Moreover, as m increases, σ^2 is reduced also so that the variances converge to zeros even faster.

Table 6: Numerical results on elastic net regression problem

$\sigma_e^2 = 100$		p=100, N=2000, m=10						
		min	max	mean	var	time(s)	d_x	td_x
$\rho = 0.5$ $\lambda = 250$	SSG	31508.2	31508.3	31508.3	0.000627	0.32	0.99	0.96
	ACSA	31508.2	31508.4	31508.3	0.001300	0.32	1.00	1.00
	RDA	31508.2	31508.3	31508.3	0.000551	0.32	0.98	0.94
	Batch	31508.2	31508.3	31508.2	0.000194	5.68	0.65	0.65
$\rho = 0.5$ $\lambda = 251$	SSG	31508.3	31508.4	31508.3	0.000339	0.32	0.99	0.87
	ACSA	31508.4	31508.5	31508.5	0.000634	0.32	1.00	1.00
	RDA	31508.3	31508.4	31508.4	0.000889	0.32	0.97	0.87
	Batch	31508.3	31508.3	31508.3	0.000106	5.79	0.41	0.41
$\sigma_e^2 = 100$		p=100, N=2000, m=100						
		min	max	mean	var	time(s)	d_x	td_x
$\rho = 0.5$ $\lambda = 250$	SSG	31508.2	31508.2	31508.2	5.40E-06	0.79	0.99	0.89
	ACSA	31508.2	31508.2	31508.2	5.28E-06	0.80	1.00	1.00
	RDA	31508.2	31508.2	31508.2	4.78E-06	0.81	0.98	0.87
	Batch	31508.2	31508.2	31508.2	1.57E-06	59.9	0.64	0.63
$\rho = 0.5$ $\lambda = 251$	SSG	31508.3	31508.3	31508.3	2.65E-06	0.75	0.97	0.59
	ACSA	31508.3	31508.3	31508.3	3.00E-06	0.75	1.00	1.00
	RDA	31508.3	31508.3	31508.3	8.01E-06	0.76	0.93	0.70
	Batch	31508.3	31508.3	31508.3	2.36E-06	60.1	0.48	0.47
$\sigma_e^2 = 100$		p=100, N=2000, m=1000						
		min	max	mean	var	time(s)	d_x	td_x
$\rho = 0.5$ $\lambda = 250$	SSG	31508.2	31508.2	31508.2	2.62E-08	5.46	0.98	0.66
	ACSA	31508.2	31508.2	31508.2	3.03E-08	5.63	1.00	1.00
	RDA	31508.2	31508.2	31508.2	6.99E-08	5.69	0.97	0.67
	Batch	31508.2	31508.2	31508.2	1.28E-08	663	0.52	0.51
$\rho = 0.5$ $\lambda = 251$	SSG	31508.3	31508.3	31508.3	1.32E-08	5.75	0.91	0.50
	ACSA	31508.3	31508.3	31508.3	1.40E-08	5.71	1.00	0.50
	RDA	31508.3	31508.3	31508.3	5.64E-08	5.89	0.85	0.51
	Batch	31508.3	31508.3	31508.3	1.35E-08	662	0.50	0.50

The values d_x and td_x in the tables represent the sparsity of the solutions in different algorithm by counting how many components of the solutions are zeros or almost zeros. Moreover, we also want to know if the non-zero components found by these algorithms are the same as the ones in the $\bar{x}$. Hence, we represent the returned solutions by the bars in Figure 1 to Figure 3. The height of the ith bar in each figure equals to $|x_i|$ in the output solution x.

From these figures, we can see that all of the algorithms find a solution whose first half of the components are significantly larger than the second half. But there are still some differences in their performances. The batch learning approach always enforces more exactly the zero components than the stochastic methods. The reasons have been given in the discussions on the results in tables.

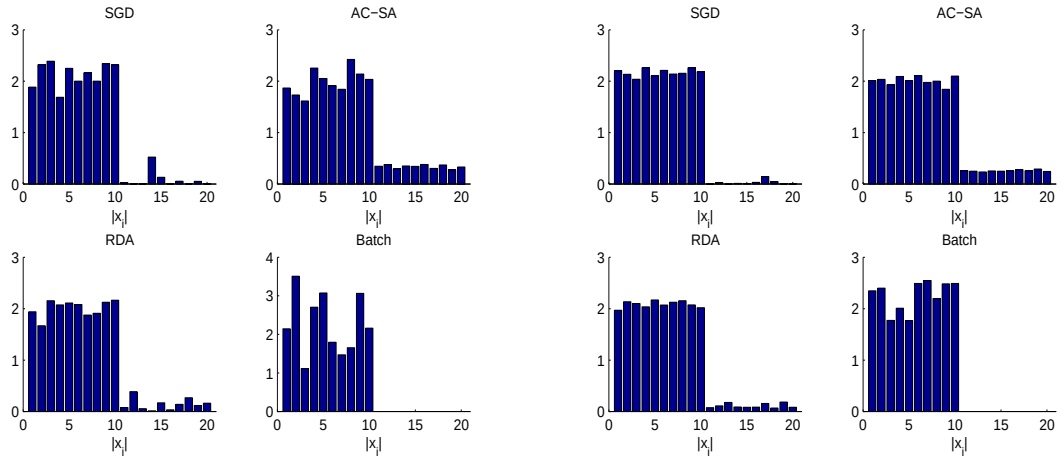

Figure 1: Sparsity of the solutions with parameters $p = 20$, $\rho = 1$, $\lambda = 20$, $\sigma_e^2 = 100$, $N = 2000$ and $m = 10$ in the left figure and $m = 100$ in the right figure.

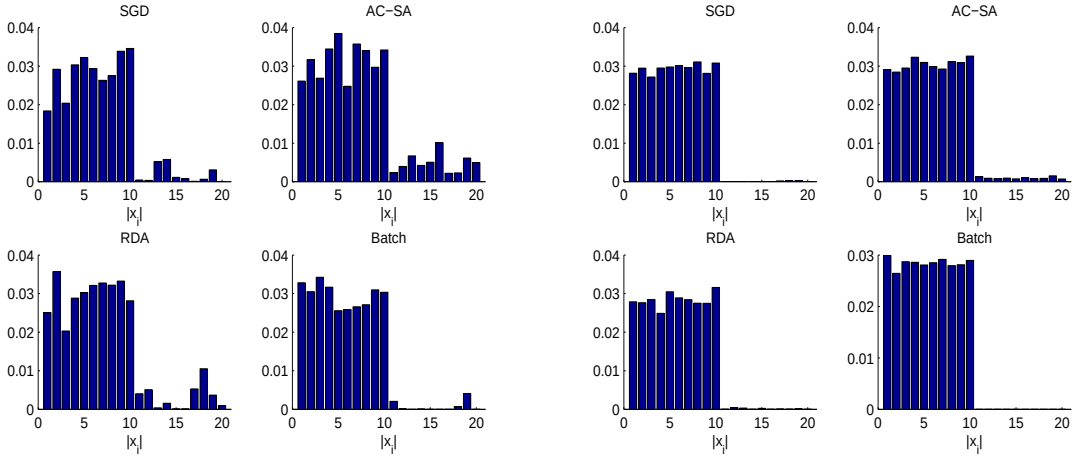

Figure 2: Sparsity of the solutions with parameters $p = 20$, $\rho = 0.5$, $\lambda = 20$, $\sigma_e^2 = 100$, $N = 2000$ and $m = 10$ in the left figure and $m = 100$ in the right figure.

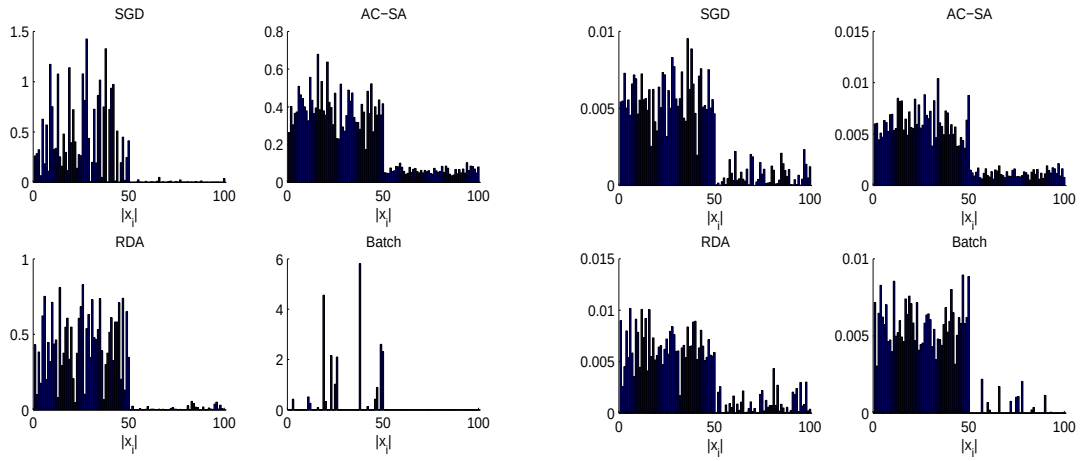

Figure 3: Sparsity of the solutions. Left: $p = 100$, $\rho = 1$, $\lambda = 120$, $\sigma_e^2 = 100$, $N = 2000$ and $m = 100$. Right: $p = 100$, $\rho = 0.5$, $\lambda = 250$, $\sigma_e^2 = 100$, $N = 2000$ and $m = 100$.

Among the stochastic approaches, our SSG algorithm is slightly better than RDA and significantly better than AC-SA in that SSG gives a smaller value or event exactly zeros in the second half components. The reason is just as what we point out in Section 1: SSG is a sparsity-preserving algorithm while AC-SA and RDA are not.

7 Conclusions

We propose a stochastic gradient algorithm (SSG) for composite optimization problems. This algorithm is an extension of Nesterov's algorithm in [9]. The algorithm is derived by implicitly constructing a stochastic estimate sequence. The convergence rates of the expected error of our algorithm are proved. To analyze the quality of the solution from a single run of our algorithm, we analyze the variance of the solution error and the probability of returning a large error. The numerical performance of our algorithm is tested on simulated data.

A Appendix

A.1 Proof of Lemma 2

Proof. According to the definition of $\widehat{x}$, there is a $\eta \in \partial h(\widehat{x})$ such that

$$\left\langle G(\bar{x}) + \bar{L}(\widehat{x} - \bar{x}) + \eta, x - \widehat{x} \right\rangle \geq 0 \text{ for all } x \in \mathbf{X}. \tag{60}$$

By the convexity of $h(x)$, we will have

$$h(x) \geq h(\widehat{x}) + \langle \eta, x - \widehat{x} \rangle . \tag{61}$$

It then follow from (2), (7), (60) and (61) that

$$\begin{aligned}
\phi(x) - \frac{\mu}{2}\|x-\bar{x}\|^2 &= f(x) - \frac{\mu}{2}\|x-\bar{x}\|^2 + h(x) \\
&\geq f(\bar{x}) + \langle \nabla f(\bar{x}), x - \bar{x}\rangle + h(\widehat{x}) + \langle \eta, x - \widehat{x}\rangle \quad \text{(by (7) and (61))} \\
&\geq f(\bar{x}) + \langle \nabla f(\bar{x}), x - \bar{x}\rangle + h(\widehat{x}) - \left\langle G(\bar{x}) + \bar{L}(\widehat{x}-\bar{x}), x - \widehat{x}\right\rangle \quad \text{(by (60))} \\
&\geq f(\widehat{x}) - \frac{L}{2}\|\widehat{x}-\bar{x}\|^2 - \langle \nabla f(\bar{x}), \widehat{x} - \bar{x}\rangle + \langle \nabla f(\bar{x}), x - \bar{x}\rangle + h(\widehat{x}) \quad \text{(by (2))} \\
&\quad - \left\langle G(\bar{x},\xi) + \bar{L}(\widehat{x}-\bar{x}), x - \widehat{x}\right\rangle \\
&= \phi(\widehat{x}) - \frac{L}{2}\|\widehat{x}-\bar{x}\|^2 + \langle \Delta, \widehat{x} - x\rangle - \left\langle \bar{L}(\widehat{x}-\bar{x}), x - \widehat{x}\right\rangle \quad \text{(by the definition of } \Delta) \\
&= \phi(\widehat{x}) - \frac{L}{2\bar{L}^2}\|g\|^2 + \langle \Delta, \widehat{x} - x\rangle + \langle g, \bar{x} - \widehat{x}\rangle + \langle g, x - \bar{x}\rangle \quad \text{(by the definition of } g) \\
&= \phi(\widehat{x}) + \frac{2\bar{L} - L}{2\bar{L}^2}\|g\|^2 + \langle g, x - \bar{x}\rangle + \langle \Delta, \widehat{x} - x\rangle .
\end{aligned}$$

Notice that the inequalities above hold for any realization of stochastic gradient $G(x,\xi)$. Hence, (15) holds almost surely. □

A.2 Proof of Lemma 4

Proof. By the choice of $V_0(x)$, (19) holds for $k = 0$ with with $V_0^* = \phi(x_0)$, $\epsilon_0 = 0$ and $\delta_0 = 0$.

Suppose we have $V_k(x) = V_k^* + \frac{\gamma_k}{2}\|x - v_k\|^2 + \langle \epsilon_k, x\rangle + \delta_k$. According to the updating equation (18), $V_{k+1}(x)$ is also a quadratic function whose coefficient on $\|x\|^2$ is $\frac{(1-\alpha_k)\gamma_k + \alpha_k\mu}{2}$. Therefore, $V_{k+1}(x)$ can definitely be represented as $V_{k+1}(x) = V_{k+1}^* + \frac{\gamma_{k+1}}{2}\|x - v_{k+1}\|^2 + \langle \epsilon_{k+1}, x\rangle + \delta_{k+1}$ with $\gamma_{k+1} = (1-\alpha_k)\gamma_k + \alpha_k\mu$ and some V_{k+1}^*, v_{k+1}, ϵ_{k+1} and δ_{k+1}.

It follows from (18) that

$$\begin{aligned}
V_{k+1}(x) &= (1-\alpha_k)V_k^* + \frac{(1-\alpha_k)\gamma_k}{2}\|x - v_k\|^2 + (1-\alpha_k)\langle \epsilon_k, x\rangle + (1-\alpha_k)\delta_k \\
&\quad + \alpha_k\left[\phi(x_{k+1}) + \langle g_k, x - y_k\rangle + \frac{2L_k - L}{2L_k^2}\|g_k\|^2 + \frac{\mu}{2}\|x - y_k\|^2 + \langle \Delta_k, x_{k+1} - x\rangle\right] \\
&= \underbrace{\frac{(1-\alpha_k)\gamma_k}{2}\|x - v_k\|^2 + \frac{\alpha_k\mu}{2}\|x - y_k\|^2 + \alpha_k\langle g_k, x - y_k\rangle}_{T_1} \\
&\quad + (1-\alpha_k)\langle \epsilon_k, x\rangle - \alpha_k\langle \Delta_k, x\rangle + (1-\alpha_k)V_k^* + (1-\alpha_k)\delta_k \\
&\quad + \alpha_k\left[\phi(x_{k+1}) + \frac{2L_k - L}{2L_k^2}\|g_k\|^2 + \langle \Delta_k, x_{k+1}\rangle\right]
\end{aligned} \tag{62}$$

We take the term $T_1 = \frac{(1-\alpha_k)\gamma_k}{2}\|x - v_k\|^2 + \frac{\alpha_k\mu}{2}\|x - y_k\|^2 + \alpha_k \langle g_k, x - y_k\rangle$ out from $V_{k+1}(x)$ and consider it separately. We define γ_{k+1} and v_{k+1} according to (20) and (21), we get

$$\begin{aligned} T_1 &= \tfrac{(1-\alpha_k)\gamma_k}{2}\|x - v_k\|^2 + \tfrac{\alpha_k\mu}{2}\|x - y_k\|^2 + \alpha_k \langle g_k, x - y_k\rangle \\ &= \tfrac{\gamma_{k+1}}{2}\|x\|^2 + \langle x, (1-\alpha_k)\gamma_k v_k + \alpha_k\mu y_k - \alpha_k g_k\rangle + \tfrac{(1-\alpha_k)\gamma_k}{2}\|v_k\|^2 + \tfrac{\alpha_k\mu}{2}\|y_k\|^2 - \alpha_k \langle g_k, y_k\rangle \\ &= \tfrac{\gamma_{k+1}}{2}\|x - v_{k+1}\|^2 - \tfrac{\gamma_{k+1}}{2}\left\|\tfrac{(1-\alpha_k)\gamma_k v_k + \alpha_k\mu y_k - \alpha_k g_k}{\gamma_{k+1}}\right\|^2 + \tfrac{(1-\alpha_k)\gamma_k}{2}\|v_k\|^2 + \tfrac{\alpha_k\mu}{2}\|y_k\|^2 - \alpha_k \langle g_k, y_k\rangle \\ &= \tfrac{\gamma_{k+1}}{2}\|x - v_{k+1}\|^2 + \tfrac{\alpha_k(1-\alpha_k)\gamma_k\mu}{2\gamma_{k+1}}\|y_k - v_k\|^2 - \tfrac{\alpha_k^2}{2\gamma_{k+1}}\|g_k\|^2 + \tfrac{\alpha_k(1-\alpha_k)\gamma_k}{\gamma_{k+1}} \langle g_k, v_k - y_k\rangle . \end{aligned} \tag{63}$$

(The last step follows from a straightforward, though somewhat tedious algebraic expansion.) Therefore, replacing T_1 in (62) with (63), we can rewrite $V_{k+1}(x)$ as

$$\begin{aligned} V_{k+1}(x) &= \frac{\gamma_{k+1}}{2}\|x - v_{k+1}\|^2 + \underbrace{(1-\alpha_k)\langle \epsilon_k, x\rangle - \alpha_k \langle \Delta_k, x\rangle}_{\langle \epsilon_{k+1}, x\rangle} \\ &\quad + \underbrace{(1-\alpha_k)\delta_k + \alpha_k \langle \Delta_k, y_k\rangle + (1-\alpha_k)\langle \Delta_k, y_k - x_k\rangle}_{\delta_{k+1}} \\ &\quad + (1-\alpha_k)V_k^* + \alpha_k\phi(x_{k+1}) + (\alpha_k\frac{2L_k - L}{2L_k^2} - \frac{\alpha_k^2}{2\gamma_{k+1}})\|g_k\|^2 \\ &\quad + \frac{\alpha_k(1-\alpha_k)\gamma_k}{\gamma_{k+1}}(\frac{\mu}{2}\|y_k - v_k\|^2 + \langle g_k, v_k - y_k\rangle) + \alpha_k \langle \Delta_k, x_{k+1}\rangle \\ &\quad - \alpha_k \langle \Delta_k, y_k\rangle - (1-\alpha_k)\langle \Delta_k, y_k - x_k\rangle \end{aligned}$$

In order to represent $V_{k+1}(x)$ by the formulation (19), it suffices to define δ_{k+1}, ϵ_{k+1} and V_{k+1}^* as in (22), (23) and (24). □

A.3 Proof of Proposition 1

Proof. We prove (11) is true by induction. According to Lemma 4, $V_k(x)$ is given by (19). Hence, we just need to prove $V_k^* \geq \phi(x_k) - B_k$. Since $V_0^* = \phi(x_0)$ and $B_0 = 0$, (11) is true when $k = 0$. Suppose (11) is true for k. We are going to show that it is also true for $k + 1$.

Dropping the non-negative term $\frac{\mu}{2}\|y_k - v_k\|^2$ in (24) and applying the assumption $V_k^* \geq \phi(x_k) - B_k$, we can bound V_{k+1}^* from below as

$$\begin{aligned} V_{k+1}^* &\geq (1-\alpha_k)\phi(x_k) - (1-\alpha_k)B_k + \alpha_k\phi(x_{k+1}) + (\alpha_k\frac{2L_k - L}{2L_k^2} - \frac{\alpha_k^2}{2\gamma_{k+1}})\|g_k\|^2 \\ &\quad + \frac{\alpha_k(1-\alpha_k)\gamma_k}{\gamma_{k+1}} \langle g_k, v_k - y_k\rangle + \alpha_k \langle \Delta_k, x_{k+1}\rangle - \alpha_k \langle \Delta_k, y_k\rangle - (1-\alpha_k)\langle \Delta_k, y_k - x_k\rangle . \end{aligned} \tag{64}$$

According to Lemma 2 (with $\widehat{x} = x_{k+1},\ \bar{x} = y_k$) and (16), we have

$$\begin{aligned}\phi(x_k) &\geq \phi(x_{k+1}) + \langle g_k, x_k - y_k\rangle + \frac{2L_k - L}{2L_k^2}\|g_k\|^2 + \frac{\mu}{2}\|x_k - y_k\|^2 + \langle \Delta_k, x_{k+1} - x_k\rangle \\ &\geq \phi(x_{k+1}) + \langle g_k, x_k - y_k\rangle + \frac{2L_k - L}{2L_k^2}\|g_k\|^2 + \langle \Delta_k, x_{k+1} - x_k\rangle .\end{aligned}$$

Applying this to $\phi(x_k)$ in (64), it follows that

$$\begin{aligned}V_{k+1}^* \geq\ & \phi(x_{k+1}) + \left(\frac{2L_k - L}{2L_k^2} - \frac{\alpha_k^2}{2\gamma_{k+1}}\right)\|g_k\|^2 - (1-\alpha_k)B_k + (1-\alpha_k)\left\langle g_k, \underbrace{\frac{\alpha_k\gamma_k}{\gamma_{k+1}}(v_k - y_k) + x_k - y_k}_{T_2}\right\rangle \\ &+\alpha_k\langle \Delta_k, x_{k+1}\rangle + (1-\alpha_k)\langle \Delta_k, x_{k+1} - x_k\rangle - \alpha_k\langle \Delta_k, y_k\rangle - (1-\alpha_k)\langle \Delta_k, y_k - x_k\rangle .\end{aligned}$$

Steps 2 and 3 in Algorithm 1 imply that the term T_2 above is zero. Hence, V_{k+1}^* can be bounded from below as

$$\begin{aligned}V_{k+1}^* &\geq \phi(x_{k+1}) + \left(\frac{2L_k - L}{2L_k^2} - \frac{\alpha_k^2}{2\gamma_{k+1}}\right)\|g_k\|^2 - (1-\alpha_k)B_k + \langle \Delta_k, x_{k+1} - y_k\rangle \\ &\geq \phi(x_{k+1}) + \left(\frac{2L_k - L}{2L_k^2} - \frac{\alpha_k^2}{2\gamma_{k+1}}\right)\|g_k\|^2 - (1-\alpha_k)B_k - \|\Delta_k\|\|x_{k+1} - y_k\| \quad \text{(Cauchy-Schwarz)} \\ &\geq \phi(x_{k+1}) - \frac{\|\Delta_k\|^2}{4L_k^2 A_k} - (1-\alpha_k)B_k = \phi(x_{k+1}) - B_{k+1}.\end{aligned}$$

Here, the last inequality is from the definition of A_k given by (28) and the following inequality

$$A_k\|g_k\|^2 - \|g_k\|\frac{\|\Delta_k\|}{L_k} \geq -\frac{\|\Delta_k\|^2}{4L_k^2 A_k}.$$

Hence, (11) holds for $k+1$. □

References

[1] Amir Beck and Marc Teboulle. A fast iterative shrikage-threshold algorithm for linear inverse problems. *SIAM Journal of Image Science*, 2(1):183–202, 2009.

[2] S. Ghadimi and G. Lan. Optimal stochastic approximation algorithms for strongly convex stochastic composite optimization. Technical report, University of Florida, 2010.

[3] Chonghai Hu, James T. Kwok, and Weike Pan. Accelerated gradient methods for stochastic optimization and online learning. In *Advances in Neural Information Processing Systems (NIPS)*, 2009.

[4] Guanghui Lan. An optimal method for stochastic composite optimization. Technical report, University of Florida, 2010.

[5] Jun Liu, Shuiwang Ji, and Jieping Ye. Multi-task feature learning via efficient ℓ_1/ℓ_2 norm minimization. In *The Twenty-fifth Conference on Uncertainty in Artificial Intelligence (UAI)*, 2009.

[6] A. Nemirovski, A. Juditsky, G. Lan, and A. Shapiro. Robust stochastic approximation approach to stochastic programming. *SIAM Journal on Optimization*, 19(4):1574–1609, 2009.

[7] A. Nemirovski and D. Yudin. *Problem complexity and method efficiency in optimization*. John Wiley, 1983.

[8] Y. E. Nesterov. Gradient methods for minimizing composite objective function. Technical report, CORE, 2007.

[9] Yurii Nesterov. *Introductory lectures on convex optimization: a basic course*. Kluwer Academic Pub, 2003.

[10] Yurii Nesterov. Smooth minimization of non-smooth functions. *Mathematical Programming*, 103(1):127–152, 2005.

[11] Yurii Nesterov. Primal-dual subgradient methods for convex problems. *Mathematical Programming*, 120:221–259, 2009.

[12] B. T. Polyak and A. B. Juditsky. Acceleration of stochastic approximation by averaging. *SIAM Journal on Control and Optimization*, 30:838–855, 1992.

[13] Ting Kei Pong, Paul Tseng, Shuiwang Ji, , and Jieping Ye. Trace norm regularization: Reformulations, algorithms, and multi-task learning. *SIAM Journal on Optimization (accepted)*, 2010.

[14] Herbert Robbins and Sutton Monro. A stochastic approximation method. *Annals of Mathematical Statistics*, 22(3):400–407, 1951.

[15] Robert Tibshirani. Regression shrinkage and selection via the lasso. *J.R.Statist.Soc.B*, 58:267–288, 1996.

[16] Robert Tibshirani and Michael Saunders. Sparsity and smoothness via the fused lasso. *J.R.Statist.Soc.B*, 67(1):91–108, 2005.

[17] K.-C. Toh and S. Yun. An accelerated proximal gradient algorithm for nuclear norm regularized least squares problems. *Pacific J. Optimization (accepted)*, 2009.

[18] Paul Tseng. On accelerated proximal gradient methods for convex-concave optimization. *SIAM Journal on Optimization (Submitted)*, 2008.

[19] David Williams. *Probability with Martingales*. Cambridge University Textbooks, 1999.

[20] Lin Xiao. Dual averaging methods for regularized stochastic learning and online optimization. *Journal of Machine Learning Research*, 11:2543–2596, 2010.

[21] Ming Yuan and Yi Lin. Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society, Series B, Methodological*, 68:49–67, 2006.

[22] Hui Zou and Trevor Hastie. Regularization and variable selection via the elastic net. *J. R. Statist. Soc. B*, 67(2):301–320, 2005.