Machine Learning manuscript No.
(will be inserted by the editor)

Distributed Block-diagonal Approximation Methods for Regularized Empirical Risk Minimization

Ching-pei Lee · Kai-Wei Chang

received: date / accepted: date

Abstract In recent years, there is a growing need to train machine learning models on a huge volume of data. Therefore, designing efficient distributed optimization algorithms for empirical risk minimization (ERM) has become an active and challenging research topic. In this paper, we propose a flexible framework for distributed ERM training through solving the dual problem, which provides a unified description and comparison of existing methods. Our approach requires only approximate solutions of the sub-problems involved in the optimization process, and is versatile to be applied on many large-scale machine learning problems including classification, regression, and structured prediction. We show that our framework enjoys global linear convergence for a broad class of non-strongly-convex problems, and some specific choices of the sub-problems can even achieve much faster convergence than existing approaches by a refined analysis. This improved convergence rate is also reflected in the superior empirical performance of our method.

Keywords Distributed optimization, large-scale learning, empirical risk minimization, dual method, inexact method

1 Introduction

With the rapid growth of data volume and model complexity, designing scalable learning algorithms has become increasingly important. Distributed opti-

Version of November 17, 2019.

Ching-pei Lee
Department of Mathematics and Institute for Mathematical Sciences, National University of Singapore
E-mail: leechingpei@gmail.com

Kai-Wei Chang
Department of Computer Science ,University of California Los Angeles
E-mail: kw@kwchang.net

mization techniques, which distribute the computational burden across multiple machines, have shown early success on this path. This type of approaches are particularly useful when the optimization problem involves massive computation or when the dataset is stored across multiple computational nodes. However, the communication cost and the asynchronous nature of distributed computation challenge the design of efficient optimization algorithms in the distributed environment.

In this paper, we study distributed optimization algorithms for training machine learning models that can be represented by the regularized empirical risk minimization (ERM) framework. Given a set of training data, $\{X_i\}_{i=1}^{l}, X_i \in \mathbf{R}^{n\times c_i}, c_i \in \mathbf{N}$, where $l, n > 0$ are the number of instances and the dimension of the model respectively, regularized ERM models solve the following optimization problem:

$$\min_{\boldsymbol{w}\in\mathbf{R}^n} \quad f^P(\boldsymbol{w}) \coloneqq g(\boldsymbol{w}) + \sum_{i=1}^{l} \xi_i\left(X_i^T \boldsymbol{w}\right). \tag{1}$$

In the literature, g and ξ_i are called the regularization term and the loss term, respectively. We assume that f^P is a proper and closed convex function that can be extended-valued and the solution set of (1) is nonempty. Besides, we specifically focus on linear models, which have been shown successful in dealing with large-scale data thank to their efficiency and interpretability.[1]

The definition in problem (1) is general and covers a variety of learning problems, including binary classification, multi-class classification, regression, and structured prediction. To unify different learning problems, we encode the true labels (i.e., $y_i \in \mathcal{Y}_i$) in the loss term ξ_i and the input data X_i. For some learning problems, the space of X_i is spanned by a set of variables whose size may vary for different i. Therefore, we represent X_i as an $n \times c_i$ matrix. For example, in the part-of-speech tagging task, where each input sentence consists of a sequence of words, c_i represents the number of words in the i-th sentence. We discuss in details the loss terms for different learning problems in Section 6. Regarding the regularization term, common choices include the squared-ℓ_2 norm, the ℓ_1 norm, and the elastic net that combines both [55].

In many applications, it is preferable to solve the dual problem whose optimization might be easier. The dual problem of (1) is

$$\min_{\boldsymbol{\alpha}\in\Omega} \quad f(\boldsymbol{\alpha}) \coloneqq g^*(X\boldsymbol{\alpha}) + \sum_{i=1}^{l} \xi_i^*(-\boldsymbol{\alpha}_i), \tag{2}$$

where

$$X \coloneqq [X_1, \dots, X_l], \quad \boldsymbol{\alpha} \coloneqq \begin{bmatrix} \boldsymbol{\alpha}_1 \\ \vdots \\ \boldsymbol{\alpha}_l \end{bmatrix},$$

[1] Linear models allow developers to interpret the value of each feature from the learned model parameters.

$\boldsymbol{\alpha}_i \in \mathbf{R}^{c_i}$ is the dual variable vector associated with X_i, for any function $h(\cdot)$, $h^*(\cdot)$ is the convex conjugate of $h(\cdot)$:

$$h^*(\boldsymbol{w}) \coloneqq \max_{\boldsymbol{z} \in \mathrm{dom}(h)} \quad \boldsymbol{z}^T \boldsymbol{w} - h(\boldsymbol{z}), \quad \forall \boldsymbol{w},$$

and as g^* is finite everywhere in our setting (see Assumption 1 and the description that follows), the domain Ω is

$$\Omega \coloneqq \prod_{i=1}^{l} -\mathrm{dom}(\xi_i^*) \subseteq \mathbf{R}^{\sum_{i=1}^{l} c_i}.$$

The goal of solving the dual problem (2) is still getting a solution to the original primal problem (1). It can be shown easily by Slater's condition that when f^P is convex, strong duality between (1) and (2) holds, which means that any pair of primal and dual optimal solutions $(\boldsymbol{w}^*, \boldsymbol{\alpha}^*)$ satisfies

$$f^P\left(\boldsymbol{w}^*\right) = -f\left(\boldsymbol{\alpha}^*\right).$$

Despite that optimization methods for the dual ERM problem (2) on a single machine have been widely studied (see, for example, the survey paper [51]), adapting them to a distributed environment is not straightforward due to the following two reasons. First, most existing single-core algorithms for dual ERM are inherently sequential and hence hard to parallelize. Second, in a distributed environment, inter-machine communication is usually the bottleneck for parallel optimizers and a careful design to reduce the communication cost is essential. For example, we may prefer an algorithm with faster convergence even if it takes longer computation time at each iteration as it induces fewer communication rounds and consequentially reduces overall communication overhead.

In this paper, we propose a distributed learning framework for solving (2). At each iteration, it minimizes a sub-problem consisting of the sum of a second-order approximation of $g^*(X\boldsymbol{\alpha})$ and the original $\boldsymbol{\xi}^*(-\boldsymbol{\alpha})$. We study how to choose the approximation to let the proposed approach enjoy not only fast theoretical and empirical convergence rate but low communication overhead. After solving the sub-problem, we conduct a line search that requires only negligible computational cost and $O(1)$ communication to ensure sufficient function value decrease. With this line search procedure, our algorithm achieves faster empirical performance compared with existing approaches.

By utilizing relaxed conditions, even if the subproblem is solved only approximately, our method is able to achieve global linear convergence for many problems whose dual objective is not strongly convex, including support vectors machines (SVM) [3,45] and structured support vector machines (SSVM) [44,42]. In other words, our algorithm takes only $O(\log(1/\epsilon))$ iterations, or equivalently, rounds of communication, to obtain an ϵ-accurate solution for (2).[2]

[2] Given any optimization problem $\min_{x \in X} f(x)$ whose minimum is attainable and denoted by f^*, we call $x \in X$ an ϵ-accurate solution for this problem if $f(x) - f^* \leq \epsilon$.

Our analysis then shows that this result implies that obtaining an ϵ-accurate solution for the original ERM problem (1) also takes only $O(\log(1/\epsilon))$ iterations. We further show that when the choice of the sub-problem properly extracts information from the Hessian of $g^*(X\boldsymbol{\alpha})$, the convergence can be significantly accelerated to reduce the required number of iterations and therefore the running time. Besides, our flexible framework generalizes existing approaches and hence facilitates the discussion of the differences between the proposed distributed learning algorithms and the existing ones for (2).

Recently, [53] proposed an accelerated method for solving the dual ERM problem in a distributed setting. Their techniques derived from [40] is similar to the Catalyst framework for convex optimization [27]. In essence, at every iteration, their approach adds a term $\kappa\|\boldsymbol{w} - \boldsymbol{z}\|_2^2/2$ to (1) and approximately solves the dual problem of the modified primal problem by an existing distributed optimization algorithm for (2). The solution is then used to generate $\boldsymbol{z}$ for the next iteration. Like the Catalyst framework that can be combined with any convex optimization algorithm, the acceleration technique in [53] can also be incorporated with our distributed learning algorithm. Specifically, we can apply our proposed algorithm to solve the modified dual problem in the procedure mentioned above. Therefore, to simplify the discussion, we focus on comparing methods that solve the original optimization problem (2), and acceleration approach discussed in [53] and other studies not designed for distributed learning (e.g., [27,40]) are not in the scope of this study.

Different from approaches that consider the theoretical communication efficiency only, our goal is to design a practical distributed training algorithm for regularized ERM with strong empirical performance in terms of the overall running time. Therefore, we propose an algorithm that is both computation and communication efficient by designing a second-order method, in which the approximated Hessian can be computed without lengthy rounds of communication. We show that this approach is also extremely communication-efficient in theory by taking the approximated Hessian as a preconditioner that can significantly improve the condition number of the problem.

Special cases of the proposed framework were published earlier as conference and workshop papers [20,18]. In the journal version, we unify the results and extend the previous work to a general setting that covers a much broader class of problems, and provide thorough theoretical and empirical analyses. We show that either when the sub-problem is solved exactly or approximately at every iteration, our approach enjoys fast linear convergence, and the convergence rate behaves benignly with respect to the inexactness. We also provide a novel analysis showing why the selected sub-problem can greatly improve the convergence speed when compared to existing general analyses.

Contributions We propose a general framework for optimizing the dual ERM problem (2) when the data are stored on multiple machines. Our contributions are summarized in the following.

1. Our framework is flexible, allowing different choices of sub-problem formulations, sub-problem solvers, and line search approaches. Furthermore,

approximate sub-problem solutions can be used. As a result, many existing methods can be viewed as special cases of our framework.
2. We provide detailed theoretical analysis, showing that our framework converges linearly on a class of problems broader than the strongly convex ones, even when the sub-problem is solved only approximately. Our analysis not only shows fast convergence of the proposed algorithm, it also provides sharper convergence guarantees for existing methods that can fit into our framework.
3. We further give an analysis through change of norm to show that under specific sub-problem choices, our algorithm can achieve much faster convergence. Our analysis gets around the dependency on the condition number defined by the Euclidean norm and can therefore obtain faster rates than existing approaches.
4. The proposed approach is also empirically communication- and computation-efficient. Our empirical study shows that it outperforms existing methods on real-world large-scale datasets.

Notations We use the following notations.

$$\boldsymbol{\xi}(X^T\boldsymbol{w}) := \sum_{i=1}^{l} \xi_i(X_i^T\boldsymbol{w}), \quad \boldsymbol{\xi}^*(-\boldsymbol{\alpha}) := \sum_{i=1}^{l} \xi_i^*(-\boldsymbol{\alpha}_i), \quad G^*(\boldsymbol{\alpha}) := g^*(X\boldsymbol{\alpha}).$$

For any positive integer m, any vector $\boldsymbol{v} \in \mathbf{R}^m$, and any $I \subseteq \{1,\dots,m\}$, $\boldsymbol{v}_I$ denotes the sub-vector in $\mathbf{R}^{|I|}$ that contains the coordinates of $\boldsymbol{v}$ indexed by I. We use $\|\cdot\|$ to denote the Euclidean norm, and when given a symmetric positive semidefinite matrix A, we denote the seminorm induced by it as

$$\|x\|_A := \sqrt{x^TAx}.$$

Assumptions We consider the following setting in this paper. First, we assume the training instances are distributed across K machines, where the instances on machine k are $\{X_i\}_{i\in J_k}$. In our setting, J_k are disjoint index sets such that

$$\bigcup_{k=1}^{K} J_k = \{1,\dots,l\}, \quad J_i \cap J_k = \phi, \quad \forall i \neq k.$$

Without loss of generality, we assume that there is a sequence of non-decreasing non-negative integers

$$0 = j_0 \leq j_1 \leq \dots \leq j_K = l$$

such that

$$J_k = \{j_{k-1}+1,\dots,j_k\}, \quad k = 1,\dots,K.$$

We do not make any further assumption on how those instances are distributed across machines. That is, the data distributions on different machines can be different. Second, the problem is assumed to have the following properties.

Assumption 1 *The loss function ξ_i is convex for all i and there exists $\sigma > 0$ such that the regularizer g in the primal problem (1) is σ-strongly convex. Namely,*

$$g(\alpha \boldsymbol{w}_1 + (1-\alpha)\boldsymbol{w}_2) \leq \alpha g(\boldsymbol{w}_1) + (1-\alpha) g(\boldsymbol{w}_2) - \frac{\sigma\alpha(1-\alpha)}{2}\|\boldsymbol{w}_1 - \boldsymbol{w}_2\|^2,$$
$$\forall \boldsymbol{w}_1, \boldsymbol{w}_2 \in \mathbf{R}^n, \quad \forall \alpha \in [0,1]. \tag{3}$$

Moreover, the convex function $f^P(\boldsymbol{w})$ is proper and closed, and (1) has a non-empty solution set.

Since g is σ-strongly convex, g^* is differentiable and has $(1/\sigma)$-Lipschitz continuous gradient [11, Part E, Theorem 4.2.2]. Therefore, G^* has $(\|X^T X\|/\sigma)$-Lipschitz continuous gradient. This also indicates that even if g is extended-valued, g^* is still finite everywhere, hence the only constraint on the feasible region is from the domain of $\boldsymbol{\xi}^*$, namely $\boldsymbol{\alpha} \in \Omega$.

Organization The paper is organized as follows. We give an overview of the proposed framework in Section 2. Implementation details and convergence analysis are respectively discussed in Sections 3 and 4. We summarize related studies for distributed ERM optimization in Section 5 and discuss applications of the proposed approach in Section 6. We then demonstrate the empirical performance of the proposed algorithms in Section 7 and discuss possible extensions and limitations of this work in Section 8. The conclusions and final remarks are in Section 9.

The code for reproducing the experimental results in this paper is available at `http://github.com/leepei/blockERM`.

2 A Block-diagonal Approximation Framework

As g^* is differentiable from Assumption 1, the KKT optimality conditions imply that for any pair of primal and dual optimal solutions $(\boldsymbol{w}^*, \boldsymbol{\alpha}^*)$,

$$\boldsymbol{w}^* = \nabla g^*(X\boldsymbol{\alpha}^*). \tag{4}$$

Although (4) holds only at the optima, we take the same formulation to define $\boldsymbol{w}(\boldsymbol{\alpha})$ as the primal iterate associated with any $\boldsymbol{\alpha}$ for the dual problem (2):

$$\boldsymbol{w}(\boldsymbol{\alpha}) \coloneqq \nabla g^*(X\boldsymbol{\alpha}). \tag{5}$$

Our framework is an iterative descent method for solving (2). Starting with an arbitrary feasible $\boldsymbol{\alpha}^0$, it generates a sequence of feasible iterates $\{\boldsymbol{\alpha}^1, \boldsymbol{\alpha}^2, \dots\} \subset \Omega$ with the property that $f(\boldsymbol{\alpha}^i) \leq f(\boldsymbol{\alpha}^j)$ if $i > j$. Each iterate is updated by a direction $\boldsymbol{\Delta\alpha}^t$ and a step size $\eta_t \geq 0$.

$$\boldsymbol{\alpha}^{t+1} = \boldsymbol{\alpha}^t + \eta_t \boldsymbol{\Delta\alpha}^t, \quad \forall t \geq 0. \tag{6}$$

The term $\boldsymbol{\xi}^*$ in (2) is separable in $\boldsymbol{\alpha}$ and hence can be optimized directly in a coordinate-wise manner. However, the term G^* is often complex and difficult to optimize. Therefore, we approximate it using a quadratic surrogate based on the fact that G^* is Lipschitz-continuously differentiable.

Putting them together, given the current iterate $\boldsymbol{\alpha}^t$, we solve

$$\begin{aligned}\boldsymbol{\Delta\alpha}^t &\approx \arg\min_{\boldsymbol{\Delta\alpha}} Q^{\boldsymbol{\alpha}^t}_{B_t}(\boldsymbol{\Delta\alpha}),\\ Q^{\boldsymbol{\alpha}^t}_{B_t}(\boldsymbol{\Delta\alpha}) &\coloneqq \nabla G^*(\boldsymbol{\alpha}^t)^T\boldsymbol{\Delta\alpha} + \frac{1}{2}(\boldsymbol{\Delta\alpha})^T B_t \boldsymbol{\Delta\alpha} + \boldsymbol{\xi}^*(-\boldsymbol{\alpha}^t - \boldsymbol{\Delta\alpha})\end{aligned} \tag{7}$$

to obtain the update direction $\boldsymbol{\Delta\alpha}^t$, where B_t for each t is some symmetric matrix selected to approximate $\nabla^2 G^*(\boldsymbol{\alpha}^t)$ (note that since ∇G^* is Lipschitz continuous, G^* is twice-differentiable almost everywhere so we at least have the generalized Hessian), and there is a wide range of choices for it, depending on the scenario. Note that it is usually hard to solve (7) to optimality unless B_t is diagonal. Therefore, we consider only attaining approximate solutions for (7). We will discuss the selection of B_t in Section 3. The general analysis in Section 4 shows that as long as the objective of (7) is strongly convex enough (in the sense that the strong convexity parameter is large enough), even if B_t is indefinite and (7) is solved only roughly, $\boldsymbol{\Delta\alpha}^t$ will be a descent direction. On the other hand, when B_t approximates $\nabla^2 G^*(\boldsymbol{\alpha}^t)$ well as in our choice, the analysis in Section 4.4 shows that the convergence speed can be highly improved, making the algorithm more communication-efficient.

Regarding the step size η_t, we investigate two line search strategies. The first is the exact line search strategy, in which we minimize the objective function along the update direction:

$$\eta_t = \arg\min_{\eta\in\mathbf{R}} \quad f(\boldsymbol{\alpha}^t + \eta\boldsymbol{\Delta\alpha}^t). \tag{8}$$

However, this approach is not practical unless (8) can be solved easily. Therefore, in general, we apply a backtracking line search strategy using a modified Armijo rule suggested by [43]. Given $\beta, \tau \in (0,1)$, our procedure finds the smallest integer $i \geq 0$ such that $\eta = \beta^i$ satisfies

$$f(\boldsymbol{\alpha}^t + \eta\boldsymbol{\Delta\alpha}^t) \leq f(\boldsymbol{\alpha}^t) + \eta\tau\Delta_t, \tag{9}$$

where

$$\Delta_t \coloneqq \nabla G^*(\boldsymbol{\alpha}^t)^T\boldsymbol{\Delta\alpha}^t + \boldsymbol{\xi}^*(-\boldsymbol{\alpha}^t - \boldsymbol{\Delta\alpha}^t) - \boldsymbol{\xi}^*(-\boldsymbol{\alpha}^t), \tag{10}$$

and takes $\eta_t = \eta$. Notice that as we are approximating the Hessian, similar to Newton and quasi-Newton methods, our backtracking always starts from trying the unit step size $\eta = 1$.

3 Distributed Implementation for Dual ERM

In this section, we provide technical details on how to apply the algorithm framework discussed in Section 2 in a distributed environment. In particular, we will discuss the choice of B_t in (7) such that the communication overhead can be reduced. We will also propose a trick to make line search efficient.

For the ease of algorithm description, we denote the i-th column of X by $\boldsymbol{x}_i$, and the corresponding element of $\boldsymbol{\alpha}$ by α_i. We also denote the number of columns of X, which is equivalent to the dimension of $\boldsymbol{\alpha}$, by

$$N := \sum_{i=1}^{l} c_i.$$

The index sets corresponding to the columns of the instances in J_k are denoted by $\tilde{J}_k \subseteq \{1, \dots, N\}, k = 1, \dots, K$. We define

$$\pi(i) = k, \quad \text{if } i \in \tilde{J}_k. \tag{11}$$

3.1 Update Direction

In the following, we discuss how to select B_t such that the objective of (7) is 1) strongly convex, 2) easy to optimize with low communication cost, and 3) a good approximation of (2).

In our assumption, the k-th machine stores and handles only X_i and the corresponding $\boldsymbol{\alpha}_i$ for $i \in J_k$. In order to reduce the communication cost, we need to pick B_t in a way such that (7) can be decomposed into independent sub-problems, of which each involves only data points stored on the same machine. In such a way, each sub-problem can be solved locally on one node without any inter-machine communication. Motivated by this, B_t should be block-diagonal (up to permutations of the instance indices) such that

$$(B_t)_{i,j} = 0, \text{ if } \pi(i) \neq \pi(j), \tag{12}$$

where π is defined in (11).

The ideal choice for B_t is to set it to be the Hessian matrix $H_{\boldsymbol{\alpha}^t}$ of $G^*(\boldsymbol{\alpha}^t)$:

$$H_{\boldsymbol{\alpha}^t} := \nabla^2 G^*(\boldsymbol{\alpha}^t) = X^T \nabla^2 g^* \left(X\boldsymbol{\alpha}^t\right) X.$$

This choice leads to the proximal Newton methods that enjoys rapid convergence in both theory and practice. However, the Hessian matrix is usually dense and does not satisfy the condition (12), incurring significant communication cost in the distributed scenario we consider here.

Therefore, we consider a block-diagonal approximation $\tilde{H}_{\boldsymbol{\alpha}^t}$ instead.

$$\left(\tilde{H}_{\boldsymbol{\alpha}^t}\right)_{i,j} = \begin{cases} (H_{\boldsymbol{\alpha}^t})_{i,j} & \text{if } \pi(i) = \pi(j), \\ 0 & \text{otherwise.} \end{cases} \tag{13}$$

Note that since

$$\nabla^2_{i,j} G^* \left(X\boldsymbol{\alpha}^t\right) = \boldsymbol{x}_i^T \nabla^2 g^* \left(X\boldsymbol{\alpha}^t\right) \boldsymbol{x}_j,$$

if each machine maintains the whole vector of $X\boldsymbol{\alpha}^t$, entries of (13) can be decomposed into parts such that each one is constructed using only data points stored on one machine. Thus, the sub-problems can be solved separately on different machines without communication. The Hessian matrix may be only positive semi-definite. In this case, when $\boldsymbol{\xi}^*(-\boldsymbol{\alpha})$ is not strongly convex, neither is problem (7), and the sub-problem can therefore be ill-conditioned. To remedy this issue, we add a damping term to B_t to ensure strong convexity of problem (7) when needed.

To summarize, our choice for B_t in distributed environments can be represented by the following formulation.

$$B_t = a_1^t \tilde{H}_{\boldsymbol{\alpha}_t} + a_2^t I, \quad \text{for some } a_1^t, a_2^t \geq 0. \tag{14}$$

The values of a_1^t and a_2^t depend on the problem structure and the applications. In most cases, we set $a_2^t = 0$, especially when it is known that either $\boldsymbol{\xi}^*(-\boldsymbol{\alpha})$ is strongly convex, or $\tilde{H}_{\boldsymbol{\alpha}_t}$ is positive definite. For a_1^t, practical results [35, 49] suggest that $a_1^t \in [1, K]$ leads to good empirical performance, while we prefer $a_1^t \equiv 1$ as it is a closer approximation to the Hessian.

In solving (7) with our choice (14) and $a_1^t \neq 0$, each machine needs the information of $X\boldsymbol{\alpha}^t$ to calculate both $(B_t)_{\tilde{J}_k, \tilde{J}_k}$ and

$$\nabla_{\tilde{J}_k} G^* \left(\boldsymbol{\alpha}^t\right) = X_{:,\tilde{J}_k}^T \nabla g^* \left(X\boldsymbol{\alpha}^t\right). \tag{15}$$

Therefore, after updating $\boldsymbol{\alpha}^t$, we need to synchronize the information

$$\boldsymbol{v}^t := X\boldsymbol{\alpha}^t = \sum_{k=1}^{K} \sum_{j \in J_k} X_j \boldsymbol{\alpha}_j^t$$

through one round of inter-machine communication. Synchronizing this n-dimensional vector across machines is more effective than transmitting either the Hessian or the whole X together with $\boldsymbol{\alpha}^t$. However, we also need the update direction for line search; therefore, instead of $\boldsymbol{v}^{t+1}$, we synchronize

$$\Delta\boldsymbol{v}^t := X\boldsymbol{\Delta\alpha}^t = \sum_{k=1}^{K} \sum_{j \in J_k} X_j \Delta\boldsymbol{\alpha}_j^t \tag{16}$$

over machines and then update $\boldsymbol{v}^{t+1}$ locally on all machines by

$$\boldsymbol{v}^{t+1} = \boldsymbol{v}^t + \eta_t \Delta\boldsymbol{v}^t$$

after the step size η_t is determined. Details of the communication overhead will be discussed in Sections 3.6 and 4.

3.2 Line Search

After the update direction $\boldsymbol{\Delta\alpha}^t$ is decided by solving (7) (approximately), we need to conduct line search to find a step size satisfying condition (9) to ensure sufficient function value decrease. On the right-hand side of (9), the first term is available from the previous iteration; therefore, we only need to evaluate (10). From (15), this can be calculated by

$$\Delta_t = \nabla g^* \left(\boldsymbol{v}^t\right)^T \Delta\boldsymbol{v}^t + \left(\boldsymbol{\xi}^* \left(-\boldsymbol{\alpha}^t - \boldsymbol{\Delta\alpha}^t\right) - \boldsymbol{\xi}^* \left(-\boldsymbol{\alpha}^t\right)\right). \tag{17}$$

We require only $O(1)$ communication overhead to evaluate the $\boldsymbol{\xi}^*$ functions in Eq. (17), and no additional computation is needed because the information of $\boldsymbol{\xi}^*(-\boldsymbol{\alpha}^t - \boldsymbol{\Delta\alpha}^t)$ is maintained when solving (7). Furthermore, because each machine has full information of $\boldsymbol{\Delta v}^t$, $\boldsymbol{v}^t$, and hence $g^*(\boldsymbol{v}^t)$, the first term in (17) can be calculated in a distributed manner as well to reduce the computational cost per machine. Thus, we can combine the local partial sums of all terms in (17) as a scalar value and synchronize it across machines. One can also see from this calculation that synchronizing $\boldsymbol{\Delta v}^t$ is inevitable for computing the required values efficiently.

For the left-hand side of (9), the calculation of the ξ_i^* terms is distributed by nature as discussed above. If $g^*(\boldsymbol{v})$ is separable, its computation can also be parallelized. Furthermore, in some special cases, we are able to evaluate $g^*(\boldsymbol{v} + \eta\boldsymbol{\Delta v})$ using a closed-form formulation cheaply. For example, when

$$g^*(\boldsymbol{v}) = \frac{1}{2}\left\|\boldsymbol{v}\right\|^2,$$

we have

$$g^*\left(\boldsymbol{v} + \eta\boldsymbol{\Delta v}\right) = \frac{1}{2}\left(\left\|\boldsymbol{v}\right\|^2 + \eta^2\left\|\boldsymbol{\Delta v}\right\|^2 + 2\eta\boldsymbol{v}^T\boldsymbol{\Delta v}\right). \tag{18}$$

In this case, we can precompute $\|\boldsymbol{\Delta v}\|^2$ and $\boldsymbol{v}^T\boldsymbol{\Delta v}$, then the calculation of (18) with different η requires only $O(1)$ computation without any communication. For the general case, though the computation might not be this low, by maintaining both $\boldsymbol{v}$ and $\boldsymbol{\Delta v}$, the calculation of $g(\boldsymbol{v} + \eta\boldsymbol{\Delta v})$ requires no additional communication and at most $O(n)$ computation locally, and this cost is negligible as other parts of the algorithm incur more expensive computation. The line search procedure is summarized in Algorithm 1.

The exact line search strategy is possible only when

$$\frac{\partial f(\boldsymbol{\alpha} + \eta\boldsymbol{\Delta\alpha})}{\partial\eta} = 0$$

has an analytic solution. For example, when f is quadratic, we can compute

$$\frac{\partial f(\boldsymbol{\alpha} + \eta\boldsymbol{\Delta\alpha})}{\partial\eta} = 0 \quad\Rightarrow\quad \eta = \frac{-\nabla f(\boldsymbol{\alpha})^T\boldsymbol{\Delta\alpha}}{\boldsymbol{\Delta\alpha}^T\nabla^2 f(\boldsymbol{\alpha})\boldsymbol{\Delta\alpha}}, \tag{19}$$

and then project η back to the interval $\{\eta \mid \boldsymbol{\alpha} + \eta\boldsymbol{\Delta\alpha} \in \Omega\}$.

3.3 Sub-Problem Solver on Each Machine

If B_t satisfies (12), (7) can be decomposed into K independent sub-problems:

$$\min_{\boldsymbol{\Delta\alpha}_{J_k}} \nabla_{J_k} G^*(\boldsymbol{\alpha}^t)^T\boldsymbol{\Delta\alpha}_{J_k} + \frac{1}{2}\boldsymbol{\Delta\alpha}_{J_k}^T(B_t)_{J_k,J_k}\boldsymbol{\Delta\alpha}_{J_k} + \sum_{i\in J_k}\xi_i^*(-\boldsymbol{\alpha}_i^t - \boldsymbol{\Delta\alpha}_i). \tag{20}$$

Since all the information needed for solving (20) is available on machine k, the sub-problems can be solved without any inter-machine communication.

Algorithm 1: Distributed backtracking line search

```
Input: $\boldsymbol{\alpha}, \boldsymbol{\Delta\alpha} \in \mathbf{R}^N$, $\beta, \tau \in (0,1)$, $f(\boldsymbol{\alpha}) \in \mathbf{R}$, $\boldsymbol{v} = X\boldsymbol{\alpha}, \Delta\boldsymbol{v} = X\Delta\boldsymbol{\alpha}$
Form a partition $\{\hat{J}_k\}_{k=1}^K$ of $\{1,\dots,n\}$
Calculate $\Delta_t$ in parallel:                          ▷ $O(1)$ communication
    $\Delta_t = \sum_{k=1}^K \left( \nabla_{\hat{J}_k} g^* (\boldsymbol{v}^t)^T \Delta\boldsymbol{v}^t_{\hat{J}_k} + \sum_{j \in J_k} \xi_j^* (-\boldsymbol{\alpha}_j + \boldsymbol{\Delta\alpha}_j) - \xi_j^* (-\boldsymbol{\alpha}_j) \right).$
$\eta \leftarrow 1$
Calculate $f(\boldsymbol{\alpha} + \eta\boldsymbol{\Delta\alpha})$ using $\boldsymbol{v}$ and $\eta\Delta\boldsymbol{v}$     ▷ $O(1)$ communication
while $f(\boldsymbol{\alpha} + \eta\boldsymbol{\Delta\alpha}) > f(\boldsymbol{\alpha}) + \eta\tau\Delta_t$ do
    $\eta \leftarrow \eta\beta$
    Calculate $f(\boldsymbol{\alpha} + \eta\boldsymbol{\Delta\alpha})$ using $\boldsymbol{v}$ and $\eta\Delta\boldsymbol{v}$ ▷ $O(1)$ communication
end
Output: $\eta$, $f(\boldsymbol{\alpha} + \eta\boldsymbol{\Delta\alpha})$
```

Our framework does not pose any limitation on the solver for (7). For example, (7) can be solved by (block) coordinate descent, (accelerated) proximal methods, just to name a few. In our experiment, we use a random-permutation cyclic coordinate descent method for the dual ERM problem [13,50,5] as our local solver. This method has been proven to be efficient in the single-core setting empirically; theoretically, it is guaranteed to converge globally linearly [47] and can outperform other variants of coordinate descent on some cases [23,48]. Other options can be adopted for specific problems or datasets under discretion, but such discussion is beyond the scope of this work.

3.4 Output the Best Primal Solution

The proposed algorithm is a descent method for the dual problem (2). In other words, it guarantees that $f(\boldsymbol{\alpha}^{t_1}) < f(\boldsymbol{\alpha}^{t_2})$ for $t_1 > t_2$. However, there is no guarantee that the corresponding primal solution $\boldsymbol{w}$ calculated by (5) decreases monotonically as well.[3] This is a common issue for all dual methods. To deal with it, we keep track of the primal objectives of all iterates, and when the algorithm is terminated, we report the model with the lowest primal objective. This is known as the pocket approach in the literature of Perceptron [10].

3.5 Stopping Condition

It is impractical to solve problem (2) exactly, as a model that is reasonably close to the optimum can achieve similar or even identical accuracy performance compared to the optimum. In practice, one can design the stopping

[3] We will show in Section 4 that the primal objective converges R-linearly, but there is no guarantee on monotonic decrease.

Algorithm 2: Distributed block-diagonal approximation method for the dual ERM problem (2).

```
Input: A feasible $\boldsymbol{\alpha}^0$ for (2), $\epsilon \geq 0$
$\bar{f} \leftarrow \infty$, $\bar{w} \leftarrow \mathbf{0}$
Compute $\boldsymbol{v}^0 = \sum_{k=1}^{K} \sum_{j \in J_k} X_j \boldsymbol{\alpha}_j^0$ and $\boldsymbol{\xi}^*(-\boldsymbol{\alpha}^0)$        ▷ $O(n)$ communication
Compute $f(\boldsymbol{\alpha}^0)$ by $\boldsymbol{v}^0$ and $\boldsymbol{\xi}^*(-\boldsymbol{\alpha}^0)$
for $t = 0, 1, 2, \dots$ do
    Compute $f^P(\boldsymbol{w}(\boldsymbol{\alpha}^t))$ by (5)        ▷ $O(1)$ communication
    if $f^P(\boldsymbol{w}(\boldsymbol{\alpha}^t)) < \bar{f}$ then
        $\bar{f} \leftarrow f^P(\boldsymbol{w}(\boldsymbol{\alpha}^t))$, $\bar{w} \leftarrow \boldsymbol{w}(\boldsymbol{\alpha}^t)$
    end
    if $f(\boldsymbol{\alpha}^t) + f^P(\boldsymbol{w}(\boldsymbol{\alpha}^t)) \leq \epsilon(f(\boldsymbol{\alpha}^0) + f^P(\boldsymbol{w}(\boldsymbol{\alpha}^0)))$ then
        Output $\bar{w}$ and terminate
    end
    Decide $a_1^t, a_2^t \geq 0$ but not both 0
    Each machine obtains $\boldsymbol{\Delta\alpha}_{J_k}^t$ by approximately solving (20) independently and
     in parallel using the local data, with $B$ decided by (14)
    Communicate $\Delta \boldsymbol{v}^t = \sum_{k=1}^{K} \left( \sum_{j \in J_k} X_j \Delta \boldsymbol{\alpha}_j^t \right)$        ▷ $O(n)$ communication
     – Variant I: Conduct line search through Algorithm 1 to obtain $\eta_t$
     – Variant II: $\eta_t \leftarrow \arg\min_\eta f(\boldsymbol{\alpha}^t + \eta \boldsymbol{\Delta\alpha}^t)$
    Each machine conducts in parallel: $\boldsymbol{\alpha}_{J_k}^{t+1} \leftarrow \boldsymbol{\alpha}_{J_k}^t + \eta_t \boldsymbol{\Delta\alpha}_{J_k}^t$, $\boldsymbol{v}^{t+1} \leftarrow \boldsymbol{v}^t + \eta_t \Delta \boldsymbol{v}^t$
end
```

condition for the training process by using the norm of the update direction, the size of Δ_t, or the decrement of the objective function value. We consider the following practical stopping criterion:

$$f(\boldsymbol{\alpha}^t) + f^P(\boldsymbol{w}(\boldsymbol{\alpha}^t)) \leq \epsilon \left(f\left(\boldsymbol{\alpha}^0\right) + f^P\left(\boldsymbol{w}\left(\boldsymbol{\alpha}^0\right)\right)\right),$$

where $\epsilon \geq 0$ is a user-specified parameter. This stopping condition directly reflects the model quality and is easy to verify as the primal and dual objectives are computed at every iteration.

The overall distributed procedure for optimizing (2) discussed in this section is described in Algorithm 2.

3.6 Cost per Iteration

In the following, we analyze the time complexity of each component in the optimization process and summarize the cost per iteration of the proposed algorithm. For the ease of analysis, we assume that the number of columns of X on each machine is $O(N/K)$, and the corresponding non-zero entries on each machine is $O(\#\text{nnz}/K)$, where #nnz is the number of non-zero elements in X. We consider general ξ^* and g^* and assume that the evaluations for $g(\boldsymbol{w})$, $g^*(\boldsymbol{v})$, and $\nabla g^*(\boldsymbol{v})$ all cost $O(n)$.[4] The part of $\nabla^2 g^*(\boldsymbol{v})$ is assumed to cost at

[4] We do not consider special cases such as $g(\boldsymbol{w}) = \|\boldsymbol{w}\|^2/2$. In those cases, further acceleration can be derived depending on the specific function structure.

most $O(n)$ (both for forming it and for its product with another vector), for otherwise we can simply replace it with a diagonal matrix as an approximation. In practice, performing exact line search is impractical unless the problem structure allows. Therefore, in the following, we only analyze the backtracking line search strategy. We also assume without loss of generality that the cost for evaluating one ξ_i^* is proportional to the dimension of the domain, namely $O(c_i)$, so the evaluation of $\boldsymbol{\xi}^*$ costs $O(N/K)$ on each machine.

We first check the cost for forming the problem (7). Note that we do not explicitly compute the values of B_t and $\nabla G^*(\boldsymbol{\alpha}^t)$. Instead, we compute only $\nabla G(\boldsymbol{\alpha}^t)^T \boldsymbol{\Delta\alpha}^t$, through $\nabla g^*(\boldsymbol{v}^t)^T \Delta\boldsymbol{v}^t$, and the part $(\boldsymbol{\Delta\alpha})^T B_t \boldsymbol{\Delta\alpha}$ under the choice (14) is obtained through $\|\boldsymbol{\Delta\alpha}\|^2$ and $(\Delta\boldsymbol{v}^t)^T \nabla^2 g^*(\boldsymbol{v}^t) \Delta\boldsymbol{v}^t$. Therefore, for the linear term, we need to compute only $\nabla g^*(\boldsymbol{v}^t)$, which costs $O(n)$ under our assumption given that $\boldsymbol{v}^t$ is already available on all the machines. For the quadratic term, it takes the same effort of $O(n)$ to get $\nabla^2 g^*(\boldsymbol{v}^t)$. Thus, forming the problem (7) costs $O(n)$ in computation and no communication is involved. Note that when considering the local sub-problems (20), the cost remains $O(n)$ as we just replace $\Delta\boldsymbol{v}$ with the local part $X_{:,J_k}\boldsymbol{\alpha}_{J_k}$, which is still a vector of dimension n.

Next, the cost of solving (20) approximately by passing through the data for a constant number of iterations T is $O(T\#\text{nnz}/K)$, as noted in most state-of-the-art single-core optimization methods for the dual ERM (e.g., [13,50]). This part involves no communication between machines as well.

For the line search, as discussed in Section 3.2, we first need to make $\Delta\boldsymbol{v}^t$ available on all machines. The computational complexity for calculating $\Delta\boldsymbol{v}^t$ through (16) is $O(\#\text{nnz}/K)$, and since the vector is of length n, it takes $O(n)$ communication cost to gather information from all machines. After $\Delta\boldsymbol{v}^t$ is available on all machines and $\nabla g^*(\boldsymbol{v}^t)$ is obtained, we can calculate the first term of (17). This step costs $O(n/K)$ and $O(1)$ in computation and communication, respectively. The term related to $\boldsymbol{\xi}^*$ is a sum over N individual functions and therefore costs $O(N/K)$. Thereafter, summing them up requires a $O(1)$ communication that can be combined with the communication for obtaining $\nabla g^*(\boldsymbol{v}^t)^T \Delta\boldsymbol{v}^t$. Given $\boldsymbol{v}^t$ and $\Delta\boldsymbol{v}^t$, for each evaluation of f under different η, it takes $O(n)$ to compute $\boldsymbol{v}^t + \eta\Delta\boldsymbol{v}^t$ and evaluate the corresponding g^*. For the part of $\boldsymbol{\xi}^*$, it costs $O(N/K)$ and $O(1)$ in computation and communication as it is a sum over N individual functions. In total, each backtracking line search iteration costs $O(n + N/K)$ computation and $O(1)$ communication.

Finally, from (5), the vector $\boldsymbol{w}(\boldsymbol{\alpha}^t)$ is the same as the gradient vector we need in (7), so there is no additional cost to obtain the primal iterate, and evaluating the primal objective costs $O(n)$ for $g(\boldsymbol{w}(\boldsymbol{\alpha}^t))$ and $O(\#\text{nnz}/K)$ for $X^T \boldsymbol{w}(\boldsymbol{\alpha}^t)$ in computation. Thus the cost of the primal objective computation is $O(\#\text{nnz}/K + n)$. It also takes $O(1)$ communication to gather the summation of ξ_i over the machines.

By assuming that each row and each column of X has at least one nonzero entry (for otherwise we can simply remove that row or column), we have

$n + N = O(\#\text{nnz})$. In summary, each iteration of Algorithm 2 costs

$$O\left(\frac{\#\text{nnz}}{K} + n + \left(\frac{N}{K} + n\right) \times \#(\text{line search})\right)$$

in computation and

$$O\left(n + \#(\text{line search})\right)$$

in communication. Later, we will show in Section 4 that the number of line search iterations is upper-bounded by a constant. Therefore, the overall cost per iteration is $O(\#\text{nnz}/K + n)$ in computation and $O(n)$ in communication.

4 Analysis

Our analysis consists of four parts. The first three parts consider a general scenario and provide worst-case convergence guarantees of the proposed algorithm. The last part considers a special choice of B_t (see (13)), which leads to a better convergence rate when the learning objective satisfies certain conditions. Specifically, we first consider the situation when the sub-problem (7) is solved to optimality every time. We demonstrate that when the objective function satisfies the Kurdyka-Łojasiewicz inequality [28,29,15],[5] the proposed algorithm converge globally linearly. Next, we show that even if the sub-problem is solved only approximately, global linear convergence is still retained under the same condition. Then, based on the relation (5), we show that as long as the dual objective converges globally linearly, so does the primal objective. Finally, we investigate the choice of B_t and provide an explanation on why the special choice of (14) (in particular with $a_1^t > 0$ and a_2^t small) improves the convergence rate and therefore the communication complexity.

Throughout this section, we assume that either of the following holds.

Assumption 2 *The function $\boldsymbol{\xi}$ is differentiable and its gradient is ρ-Lipschitz continuous for some $\rho > 0$. That is,*

$$\|\nabla \boldsymbol{\xi}(\boldsymbol{z}_1) - \nabla \boldsymbol{\xi}(\boldsymbol{z}_2)\| \leq \rho \|\boldsymbol{z}_1 - \boldsymbol{z}_2\|, \quad \forall \boldsymbol{z}_1, \boldsymbol{z}_2.$$

Assumption 3 *The function $\boldsymbol{\xi}$ is L-Lipschitz continuous for some $L > 0$.*

$$|\boldsymbol{\xi}(\boldsymbol{z}_1) - \boldsymbol{\xi}(\boldsymbol{z}_2)| \leq L \|\boldsymbol{z}_1 - \boldsymbol{z}_2\|, \quad \forall \boldsymbol{z}_1, \boldsymbol{z}_2.$$

These assumptions on $\boldsymbol{\xi}$ are less strict than requiring each sub-component ξ_i to satisfy certain properties.

[5] The Kurdyka-Łojasiewicz inequality is much weaker than strong convexity.

4.1 Convergence Analysis when Sub-Problems are Solved Exactly

We assume the following condition based on the Kurdyka-Łojasiewicz (KL) inequality [28,29,15] holds.

Assumption 4 *The objective function in the dual problem (2) satisfies the Kurdyka-Lojasiewicz inequality with exponent* $1/2$ *for some* $\mu > 0$*. That is,*

$$f(\boldsymbol{\alpha}) - f^* \leq \frac{\min_{\hat{\boldsymbol{s}} \in \partial f(\boldsymbol{\alpha})} \|\hat{\boldsymbol{s}}\|^2}{2\mu} = \frac{\min_{\boldsymbol{s} \in \partial \boldsymbol{\xi}^*(-\boldsymbol{\alpha})} \|\nabla G(\boldsymbol{\alpha}) + \boldsymbol{s}\|^2}{2\mu}, \forall \boldsymbol{\alpha} \in \Omega. \quad (21)$$

where f^* *is the optimal objective value of the dual problem (2), and* $\partial \boldsymbol{\xi}^*(-\boldsymbol{\alpha})$ *is the set of sub-differential of* $\boldsymbol{\xi}^*$ *at* $-\boldsymbol{\alpha}$*.*

The following lemma shows that Assumption 2 implies Assumption 4.

Lemma 1 *Consider the primal problem (1). If Assumption 2 holds, then the dual problem satisfies (21) with* $\mu = 1/\rho$*.*

We start from showing that the update direction is indeed a descent direction and Algorithm 1 terminates within a bounded number of steps.

Lemma 2 *If* B_t *is chosen so that the smallest eigenvalue of* B_t *is no smaller than some constant* C_1 *(which can be nonpositive) for all* t*, and* $Q_{B_t}^{\boldsymbol{\alpha}^t}$ *is* C_2*-strongly convex for some* $C_2 > 0$ *for all* t *and* $C_1 + C_2 > 0$*, then the update direction obtained by solving (7) exactly is a descent direction, and Algorithm 1 terminates in finite steps, with the generated step size lower bounded by*

$$\eta_t \geq \min\left(1, \frac{\beta(1-\tau)\sigma\left(C_1 + C_2\right)}{\|X^T X\|}\right), \forall t.$$

In contrast to most Newton-type methods such as [43,25] that require B_t to be positive definite, the conditions required in Lemma 2 are weaker, as even if B_t is not positive definite, $Q_{B_t}^{\boldsymbol{\alpha}^t}$ can be strongly convex when Assumption 2 holds. As our framework is more general, we allow a broader choice of B_t. In Lemma 2, consider the choice of B_t in (14), since $\tilde{H}_{\boldsymbol{\alpha}_t}$ is positive semidefinite, we have that $C_1 = a_2^t$. For C_2, if Assumption 2 holds, then since $\boldsymbol{\xi}^*$ is $(1/\rho)$-strongly convex, we have that $C_2 = C_1 + 1/\rho$, and otherwise $C_2 = C_1$.

Now, we are ready to show the global linear convergence of the proposed Algorithm 2 for solving (2).

Theorem 1 *If Assumption 4 holds, there exists* $C_3 > 0$ *such that* $\|B_t\| \leq C_3$ *for all* t*, and that the conditions in Lemma 2 are satisfied for all iterations for some* C_2 *and* $C_1 \leq C_3$*, then the sequence of dual objective values generated by Algorithm 2 converges Q-linearly to the optimum, with a rate of*

$$\frac{f(\boldsymbol{\alpha}^{t+1}) - f^*}{f\left(\boldsymbol{\alpha}^t\right) - f^*}$$
$$\leq 1 - \frac{\mu\left(C_1 + C_2\right)\tau}{\mu\left(C_1 + C_2\right)\tau + 2\left(\frac{\|X^T X\|^2}{\sigma^2} + C_3^2\right)} \min\left\{1, \frac{\beta\left(1-\tau\right)\sigma\left(C_1 + C_2\right)}{\|X^T X\|}\right\}, \forall t.$$

4.2 Convergence Analysis when Sub-Problems are Solved Approximately

In practice, when B_t is not diagonal, the sub-problem (7) is usually solved by an iterative solver and it is time consuming to obtain an exact solution. In this subsection, we show that Algorithm 2 still converges linearly with inexact sub-problem solutions. Our analysis is based on that in [22,36] to assume that (7) is solved γ-approximately for some $\gamma \in [0, 1)$, defined below.

Definition 1 We say that $\boldsymbol{\Delta\alpha}^t$ solves (7) γ-approximately for some γ if

$$Q_{B_t}^{\boldsymbol{\alpha}^t}(\boldsymbol{\Delta\alpha}^t) - \min_{\boldsymbol{\Delta\alpha}} Q_{B_t}^{\boldsymbol{\alpha}^t}(\boldsymbol{\Delta\alpha}) \le \gamma \left(Q_{B_t}^{\boldsymbol{\alpha}^t}(\mathbf{0}) - \min_{\boldsymbol{\Delta\alpha}} Q_{B_t}^{\boldsymbol{\alpha}^t}(\boldsymbol{\Delta\alpha}) \right). \tag{22}$$

We will show that linear convergence can be obtained as long as the problem (2) is convex and the quadratic growth condition holds.

Assumption 5 *The function f in the dual problem (2) satisfies the quadratic growth condition with some $\mu > 0$. That is, let A be the solution set, then*

$$f(\boldsymbol{\alpha}) - f^* \ge \frac{\mu}{2} \min_{\boldsymbol{\alpha}^* \in A} \|\boldsymbol{\alpha} - \boldsymbol{\alpha}^*\|^2, \forall \boldsymbol{\alpha}. \tag{23}$$

Notice that [2, Theorem 5] has shown that (23) and (21) are equivalent when f is convex. Here we use this equivalent condition for the ease of the convergence proof.

We are now able to present the convergence results of the inexact version of our algorithm.

Lemma 3 *If B_t is chosen so that the smallest eigenvalue of B_t is no smaller than some constant C_1 (can be nonpositive) for all t, $Q_{B_t}^{\boldsymbol{\alpha}^t}$ is C_2-strongly convex for some $C_2 > 0$ for all t, and $(1+\sqrt{\gamma})C_1 + (1-\sqrt{\gamma})C_2 > 0$, then the update direction obtained by solving (7) at least γ-approximately for some $\gamma \in [0, 1)$ is a descent direction, and Algorithm 1 terminates in finite steps, with the generated step size lower bounded by*

$$\eta_t \ge \min\left(1, \frac{\beta(1-\tau)\sigma\left(\left(1+\sqrt{\gamma}\right)C_1 + \left(1-\sqrt{\gamma}\right)C_2\right)}{\|X^TX\|\left(1+\sqrt{\gamma}\right)}\right). \tag{24}$$

Theorem 2 *If Assumption 5 holds, there exists $C_3 > 0$ such that both $\|B_t\| \le C_3$ and the conditions in Lemma 3 are satisfied for all t for some $C_1 \in [0, C_3]$, C_2, and $\gamma \in [0, 1)$, then the dual objective sequence generated by Algorithm 2 converges Q-linearly as follows.*

$$\begin{aligned} &\frac{f\left(\boldsymbol{\alpha}^{t+1}\right) - f^*}{f\left(\boldsymbol{\alpha}^t\right) - f^*} \\ \le &\begin{cases} 1 - \frac{\tau\mu}{4C_3} \min\left\{1-\gamma, \frac{\beta(1-\tau)\sigma\left((1-\gamma)C_1 + \left(1-\sqrt{\gamma}\right)^2 C_2\right)}{\|X^TX\|}\right\}, & \text{if } \mu \le 2C_3, \\ 1 - \tau\left(1 - \frac{C_3}{\mu}\right) \min\left\{1-\gamma, \frac{\beta(1-\tau)\sigma\left((1-\gamma)C_1 + \left(1-\sqrt{\gamma}\right)^2 C_2\right)}{\|X^TX\|}\right\}, & \text{else.} \end{cases} \end{aligned} \tag{25}$$

4.3 Convergence of the Primal Objective Using $\boldsymbol{w}(\boldsymbol{\alpha})$

Next, we show the convergence rate of the primal problem (1) based on the above linear convergence results for the dual problem. Our analysis here needs neither Assumption 4 nor Assumption 5 to hold. The following theorem is obtained from the duality gap guarantees of the algorithms in [1,38], through taking the dual iterates generated by Algorithm 2 and their corresponding primal iterates in (5) as the initial point for their algorithms.

Theorem 3 *For any $\epsilon > 0$ and any ϵ-accurate solution $\boldsymbol{\alpha}$ for (2), the $\boldsymbol{w}$ obtained through (5) is:*

1. *$(\epsilon(1 + \rho\|X^T X\|/\sigma))$-accurate for (1), if Assumption 2 holds, or*
2. *$(\max\{2\epsilon, \sqrt{8\epsilon\|X^T X\|L^2/\sigma}\})$-accurate for (1), if Assumption 3 holds.*

By noting that $\log \sqrt{1/\epsilon} = \log(1/\epsilon)/2$, we get the following corollary.

Corollary 1 *If we apply Algorithm 2 to solve a regularized ERM problem that satisfies either Assumption 2 or Assumption 3, and the dual objective at the iterates $\boldsymbol{\alpha}^t$ converges Q-linearly to the optimum, then the primal objective evaluated at the iterates $\boldsymbol{w}^t$ obtained from the dual iterates $\boldsymbol{\alpha}^t$ via (5) converges R-linearly to the optimum at the rate given by Theorem 2.*

4.4 Convergence Improvement by Using the Specific Choice of B_t

Our analysis so far does not consider the effect of the specific choices of (14) in the quadratic approximation. These results provided a worst-case guarantee that ensures the proposed algorithms are provably efficient even for ill-conditioned problems. In the following, we demonstrate that in most cases, the choice of B_t in (14) leads to a better convergence rate because it leverages the curvature information of the original problem to improve the problem condition.

In particular, we assume the following.

Assumption 6 *In (2), the smooth term G^* is L_B-Lipschitz continuously differentiable with respect to the seminorms induced by the matrices B_t defined in (14) around the point $\boldsymbol{\alpha}^t$:*

$$G^*(\boldsymbol{\alpha}) \leq G^*\left(\boldsymbol{\alpha}^t\right) + \nabla G^*(\boldsymbol{\alpha}^t)^T\left(\boldsymbol{\alpha} - \boldsymbol{\alpha}^t\right) + \frac{L_B}{2}\|\boldsymbol{\alpha} - \boldsymbol{\alpha}^t\|_{B_t}^2, \quad \forall \boldsymbol{\alpha} \in \Omega \text{ close enough to } \boldsymbol{\alpha}^t, \forall t. \tag{26}$$

The objective function f has quadratic growth with factor μ_B with respect to the same seminorms:

$$f(\boldsymbol{\alpha}) - f^* \geq \frac{\mu_B}{2} \min_{\boldsymbol{\alpha}^* \in A} \|\boldsymbol{\alpha} - \boldsymbol{\alpha}^*\|_{B_t}^2, \forall t, \forall \boldsymbol{\alpha} \in \Omega. \tag{27}$$

Without loss of generality, we assume $\mu_B \leq 2$.

The assumption states that using the matrix B_t defined in (14) gives a better problem condition under the norm change (note that $\|X^T X\|/(\mu\sigma)$ is the condition number of the dual problem under the Euclidean norm). The Lipschitz continuity part is only needed locally as in our convergence proof, this constant is only used for the step size bound, which is for the local behavior in the region between the current and the next iterates. When G^* is quadratic, i.e., when $g(\cdot) = \|\cdot\|^2/2$ in (1), B_t becomes a fixed matrix if a_1^t and a_2^t are fixed over t, and this seminorm definition is more intuitive and can be verified easily. In particular, in this case $\nabla^2 G(\boldsymbol{\alpha}) \equiv X^T X$, so

$$\begin{cases} \nabla^2 G & \preceq K\tilde{H}_{\boldsymbol{\alpha}}, \forall \boldsymbol{\alpha}, \\ \nabla^2 G & \preceq \|X^T X\| I, \end{cases} \tag{28}$$

where the first inequality is from the observation that

$$\left\| \sum_{i=1}^{l} \boldsymbol{\alpha}_i X_i \right\|^2 \leq K \sum_{k=1}^{K} \left\| \sum_{i \in J_k} \boldsymbol{\alpha}_i X_i \right\|^2. \tag{29}$$

We can now show the improved convergence results. We start from that the preconditioner can increase the step size.

Lemma 4 *Given $\boldsymbol{\alpha}^t$, if we use B_t defined in (14) to form the sub-problem (7), the sub-problem is solved at least γ-approximately for some $\gamma \in [0,1)$, and Assumption 6 holds, then the obtained update direction $\Delta\boldsymbol{\alpha}^t$ satisfies*

$$\Delta_t \leq -\frac{1}{1+\sqrt{\gamma}} \left\|\Delta\boldsymbol{\alpha}^t\right\|_{B_t}^2. \tag{30}$$

Moreover, given $\beta, \tau \in (0,1)$, Algorithm 1 produces a step size lower-bounded by

$$\eta_t \geq \min\left\{1, \frac{2\beta(1-\tau)}{L_B(1+\sqrt{\eta})}\right\}.$$

If L_B is much smaller than L, Lemma 4 provides a step size larger than what we had from the general analysis. By utilizing the inequalities in (28) for an upper bound on L_B, we can see that usually a step size no smaller than a_1^t/K can be expected if $a_1^t \in [0, K]$.

Note that we can take the larger of the bound from Lemma 3 and that from Lemma 4, so we are always guaranteed to have a bound that is no worse. We therefore assume without loss of generality that the bound from Lemma 4 is the larger one.

We proceed on to the improved convergence speed on the dual problem.

Theorem 4 *If Assumption 6 holds, and that the conditions in Lemma 4 are satisfied for all iterations, then the sequence of dual objective values generated* *by Algorithm 2 with B_t defined in (14) converges Q-linearly to the optimum, with the rate being*

$$\frac{f\left(\boldsymbol{\alpha}^{t+1}\right) - f^*}{f\left(\boldsymbol{\alpha}^{t}\right) - f^*} \leq 1 - \frac{\tau\mu_B(1-\gamma)\eta_t}{4} \tag{31}$$

$$\leq 1 - \frac{\tau\mu_B}{2}\min\left\{frac1-\gamma2, \frac{\beta\left(1-\tau\right)\left(1-\sqrt{\gamma}\right)}{L_B}\right\}. \tag{32}$$

When L_B/μ_B is much smaller than the condition number $\|X^TX\|/(\mu\sigma)$ defined by the Euclidean norm, which is usually the case as B_t approximates the Hessian closely, the rate in (32) is significantly faster than the best possible rates in Theorems 1 and 2 obtained by making the algorithm the proximal gradient method with a fixed step size. Finally, by combining Theorem 4 and Theorem 3, we get a faster convergence rate for the primal objective as well.

Corollary 2 *If we apply Algorithm 2 with B_t defined in (14) to solve a regularized ERM problem that satisfies either Assumption 2 or Assumption 3, and Assumption 6 holds, then the primal iterates $\boldsymbol{w}(\boldsymbol{\alpha}^t)$ obtained from the dual iterates $\boldsymbol{\alpha}^t$ via (5) give primal objectives that converge to the optimum R-linearly at the rate given by Theorem 4.*

5 Related Works

The general convergence theory of the inexact version of our general framework in Section 2 follows from [22,36] on the line of inexact variable metric methods for regularized optimization. The analysis of the exact version, derived independent of the theory in [22,36], is applicable to a broader choice of sub-problems, in the sense that indefinite B_t is allowed in the exact version.

On the other hand, our focus is on how to devise a good approximation of the Hessian matrix of the smooth term that makes distributed optimization efficient. Works focusing on this direction for dual ERM problems include [35, 49,30]. [35] discusses how to solve the SVM dual problem in a distributed manner. This problem is a special case of (2); see Section 6.1 for more details. They proposed a method called DSVM-AVE that iteratively solves (7) using the B_t defined in (14) with $a_1^t \equiv 1, a_2^t \equiv 0$ to obtain the update direction, while the step size η_t is fixed to $1/K$. Though they did not provide theoretical convergence guarantee in [35], with the understanding the SVM dual problem is quadratic, our analysis in Lemma 4 and the bound in (28) gives convergence guarantee for their choice.

In [49], the algorithm DisDCA is proposed to solve (2) under the assumption that g is strongly convex. They consider the case $c_i \equiv 1$ for all i, but the algorithm can be directly generalized to $c_i > 1$. DisDCA specifically uses the stochastic dual coordinate descent (SDCA) method [39] to solve the local sub-problems, while the choice of B_t is picked according to the algorithm parameters. To solve the sub-problem on machine k, each time SDCA samples one entry i_k from J_k with replacement and minimizes the local objective with respect to $\boldsymbol{\alpha}_{i_k}$. At each iteration of their algorithm, each time machine k selects m_k entries in uniform random to form the sub-problem, and let us denote $m := \sum_{k=1}^{K} m_k$. The first variant of DisDCA, called the basic variant in [49], sets B_t in (7) as

$$(B_t)_{i,j} = \begin{cases} \frac{m}{\sigma} \boldsymbol{x}_i^T \boldsymbol{x}_j & \text{if } \boldsymbol{x}_i, \boldsymbol{x}_j \text{ are from the same } X_k \text{ for some } k \text{ sampled,} \\ 0 & \text{else,} \end{cases}$$

and the step size is fixed to 1. In this case, it is equivalent to splitting the data into l blocks, and the minimization is conducted only with respect to the blocks selected. If we let I be the indices not selected, then following the same reasoning for (29), we have

$$\|\boldsymbol{d}\|^2_{\nabla^2 G^*(\boldsymbol{\alpha})} \leq \boldsymbol{d}^T B_t \boldsymbol{d}, \quad \forall \boldsymbol{d} \text{ such that } \boldsymbol{d}_I = \boldsymbol{0}, \forall \boldsymbol{\alpha}, \tag{33}$$

Therefore, by (33) and the Lipschitz-continuous differentiability of G^*, it is not hard to see that in this case minimizing $Q^{\boldsymbol{\alpha}}_{B_t}$ directly results in a certain amount of function value decrease. The analysis in [49] then shows that the primal iterates $\{\boldsymbol{w}_t\}$ obtained by substituting the dual iterates $\{\boldsymbol{\alpha}_t\}$ into (5) converges linearly to the optimum when all ξ_i have Lipschitz continuous gradient and converges with a rate of $O(1/\epsilon)$ when all ξ_i are Lipschitz continuous by using some proof techniques similar to that in [38]. As we noted in Section 4, this is actually the same as showing the convergence rate of the dual objective and then relating it to the primal objective.

The second approach in [49], called the practical variant, considers

$$(B_t)_{i,j} = \begin{cases} \frac{K}{\sigma} \boldsymbol{x}_i^T \boldsymbol{x}_j & \text{if } \pi(i) = \pi(j), \\ 0 & \text{else,} \end{cases}$$

and takes unit step sizes. Similar to our discussion above for their basic variant, $Q^{\boldsymbol{\alpha}}_{B_t}$ in this case is also an upper bound for the function value decrease if the step size is fixed to 1, and we can expect this method to work better than the basic variant as the approximation is closer to the real Hessian and the scaling factor is closer to one. Empirical results show that this variant is as expected faster than the basic variant, despite the lack of theoretical convergence guarantee in [49].

Both DSVM-AVE and the practical variant of DisDCA are generalized to a framework proposed in [30] that discusses the relation between the second-order approximation and the step size. In particular, their theoretical analysis for fixed step sizes starts from (28) and the Lipschitz continuous differentiability of G^* to form safe upper bounds for the function value decrease. They considered solving (7) with B_t defined as

$$(B_t)_{i,j} = \begin{cases} \frac{a}{\sigma} \boldsymbol{x}_i^T \boldsymbol{x}_j & \text{if } \pi(i) = \pi(j), \\ 0 & \text{else,} \end{cases} \tag{34}$$

and showed that for $a \in [1, K]$, a step size of a/K is enough to ensure convergence of the dual objective, similar to our result in Lemma 4. As we discussed above for [35] and [49], this choice can be proven to ensure objective value decrease. Unlike DisDCA which is tied to SDCA, their framework allows arbitrary solver for the local sub-problems, and relates how precisely the local sub-problems are solved with the convergence rate. If we ignore the part of local precision, the convergence rates of their framework shown in [30] is similar to that of [49] for the basic variant of DisDCA. This work therefore provides theoretical convergence rates similar to the basic variant of DisDCA for both DSVM-AVE and the practical variant of DisDCA, and their experimental results shows that the practical variant of DisDCA is indeed the most efficient. Notice that despite empirically bettering the proximal gradient method, these works all provide convergence rates no better than it. Fortunately, as these methods can be seen as special cases of our framework, with our analysis in Section 4.4, we can explain why they are all faster than the proximal gradient method. Our analysis in Section 4.4 is, up to our best knowledge, a novel result that shows how improved convergence speed can be obtained when the second-order approximation is selected properly.

When we use the B_t considered by those works in (7), the major algorithmic difference is that we do not take a pre-specified safe step size. Instead, we dynamically find a possibly larger step size that, according to (31), can provide more function decrease than directly applying the lower bound in Lemma 4. We can see that the choice of $a = 1$ in (34) gives too conservative the step size, while the choice of $a = K$ might make the quadratic approximation in (7) deviate from the real Hessian too far. In particular, assuming $\nabla^2 g \equiv I/\sigma$, the case of $a = 1$ makes the Frobenius norm of the difference between $\nabla^2 G^*$ and B_t the smallest, while other choices increase this value. This suggests that using $a = 1$ should be the best approximation one can get, but even directly using $\nabla^2 G^*$ might not guarantee decrease of the objective value. Our method thus provides a way to deal with this problem by adding a low-cost line search step. Moreover, by adding Assumption 5 that holds true for most ERM losses (see discussion in the next section), we are able to show linear convergence for a broader class of problems.

Most other distributed ERM solvers directly optimize (1). Primal batch solvers for ERM that require computing the full gradient or Hessian-vector products are inherently parallelizable and can be easily extended to distributed environments as the main computations are matrix-vector products like $X^T \boldsymbol{w}$. It mostly takes only some implementation modifications to make these approaches efficient in distributed environments. Among them, it has been shown that distributed truncated-Newton [54,26], distributed limited-memory BFGS [6], and the limited-memory common-directions method [21] are the most efficient ones in practice. These methods have the advantage that their convergences are invariant of the data partition, though with the additional requirement that the primal objective is differentiable or even twice-differentiable in comparison with ours. However, there are important cases of ERM problems that do not possess differentiable primal objective function such as the SVM problem. In these cases, one still needs to consider the dual approaches, for otherwise the convergence might be extremely slow. Another popular distributed algorithm for solving (1) without requiring differentiability is the alternating direction method of multipliers (ADMM) [4], which is widely used in consensus problems. However, it has been shown in [49] and [54] that DisDCA and truncated-Newton outperforms ADMM on various ERM problems.

Many lately proposed distributed optimization methods focus on the communication efficiency. By increasing the computation per iteration, they are able to use fewer communication rounds to obtain the same level of objective value. However, these approaches either rely on the stronger assumption that data points across machines are independent and identically distributed (i.i.d.), or has higher computational dependency on the dimensionality of the problem. The former assumption may not hold in practice, while the latter results in computational-inefficient and thus impractical methods. For example, [52] consider a damping Newton method with a preconditioned conjugate gradient (PCG) method to solve the Newton linear system, with the preconditioner being the Hessian from a specific machine. However, the computational cost of PCG is much higher because each iteration of which involves inverting the local Hessian. The distributed SVRG method proposed by [24] has good computational and communication complexity simultaneously, but needs the assumption that data points on each machine follow a certain distribution and requires overlapping data points on different machines. This implies more communication in advance to distribute data points, which is prohibitively expensive when the data volume is huge. Indeed, instead of a real distributed environment, their experiment is simulated in a multi-core environment because of this constraint. We therefore exclude comparison with these methods.

Recently, [53] adopted for DisDCA an acceleration technique in [27,40], resulting in a theoretically faster algorithm. This technique repeatedly uses DisDCA to solve a slightly modified objective every time to some given precision, and reconstruct a new objective function based on the obtained iterate and the previous iterate. The same technique can also be applied to this work in the same fashion by replacing DisDCA with the proposed method. Therefore, we focus on the comparison with methods before applying the acceleration technique, with the understanding that the faster method of the same type before acceleration will result in a faster method after acceleration as well. Moreover, what is the best way to apply the acceleration technique to obtain the best efficiency for distributed optimization of ERM problems is itself another open research problem. Issues including whether to apply it on the primal or the dual problem, should restarting be considered, how to estimate the unknown parameters, and so on, are left to future work.

6 Applications

In this section, we apply the proposed algorithm in Section 2 to solve various regularized ERM problems and discuss techniques for improving the efficiency by utilizing the problem structures. We will demonstrate the empirical performance of the proposed algorithms in Section 7.

We first show that a class of problems satisfies Assumption 5.

Lemma 5 ([34, Theorem 10]) *Consider a problem of the following form*

$$\begin{aligned} \min_{\boldsymbol{\alpha}} \quad & F(\boldsymbol{\alpha}) := g(A\boldsymbol{\alpha}) + \boldsymbol{b}^T\boldsymbol{\alpha} && (35a)\\ \textit{subject to} \quad & C\boldsymbol{\alpha} \leq \boldsymbol{d}, && (35b)\end{aligned}$$

where g is strongly convex with any feasible initial point $\boldsymbol{\alpha}^0$. Then F satisfies the condition (21) in the level set $\{\boldsymbol{\alpha} \mid C\boldsymbol{\alpha} \leq \boldsymbol{d}, \quad F(\boldsymbol{\alpha}) \leq F(\boldsymbol{\alpha}^0)\}$ for some $\mu > 0$ that depends on the initial point $\boldsymbol{\alpha}^0$. If the constraint is a polytope, then the condition (21) holds for all feasible $\boldsymbol{\alpha}$.

6.1 Binary Classification and Regression

The first case is the SVM problem [3,45] where $c_i \equiv 1$, and given $C > 0$,

$$\xi_i(z) \equiv C\max(1 - y_i z, 0), \quad g(\boldsymbol{w}) = \frac{1}{2}\|\boldsymbol{w}\|^2,$$

with $y_i \in \{-1, 1\}, \forall i$. Obviously, Assumption 1 holds with $\sigma = 1$ in this case, and ξ_i are 1-Lipschitz continuous. Based on a straightforward derivation, we have that

$$g^*(X\boldsymbol{\alpha}) = \frac{1}{2}\|X\boldsymbol{\alpha}\|^2, \quad \xi_i^*(-\alpha_i) = \mathbb{1}_{[0,C]}(\alpha_i y_i) - \alpha_i y_i, \tag{36}$$

where

$$\mathbb{1}_{[0,C]}(x) := \begin{cases} 0 & \text{if } x \in [0, C], \\ \infty & \text{else.} \end{cases}$$

It is clear that ξ_i are C-Lipschitz continuous. For the dual problem, we see that the constraints are of the form (35b), g^* is strongly convex with respect to $X\boldsymbol{\alpha}$, and the remaining term $-\alpha_i y_i$ is linear, so the objective function satisfies the form (35a). Therefore, by Lemma 1, (21) is satisfied. Therefore all conditions of Assumption 3 are satisfied. Hence from Corollary 1, our algorithm enjoys linear convergence in solving the SVM dual problem.

Besides, we can replace the hinge loss (L1 loss) in SVM with the squared-hinge loss (L2 loss):

$$\xi_i(z) \equiv C\max(1 - y_i z, 0)^2,$$

and then ξ_i becomes differentiable, with the gradient being Lipschitz continuous. Therefore, Assumption 2 is satisfied, and we can apply Corollary 1. We have that

$$\xi_i^*(-\alpha_i) = \mathbb{1}_{[0,\infty)}(\alpha_i y_i) - \alpha_i y_i + \frac{\alpha_i^2}{4C}.$$

Loss name	$\xi_i(\boldsymbol{z})$	Assumption	$\xi_i^*(-\boldsymbol{\alpha})$
L1-loss SVM	$C\max(1-y_i z, 0)$	3	$\mathbb{1}_{[0,C]}(\alpha y_i) - \alpha y_i$
L2-loss SVM	$C\max(1-y_i z, 0)^2$	2	$\mathbb{1}_{[0,\infty)}(\alpha y_i) - \alpha y_i + \frac{\alpha^2}{4C}$
logistic regression	$C\log(1+\exp(-y_i z))$	2	$\mathbb{1}_{[0,C]}(\alpha y_i) + \alpha y_i \log(\alpha y_i)$ $+(C-\alpha y_i)\log((C-\alpha y_i))$
SVR	$C\max(\lvert z-y_i\rvert-\epsilon, 0)$	3	$\mathbb{1}_{[-C,C]}(\alpha) + \epsilon\lvert\alpha_i\rvert - \alpha y_i$
L2-loss SVR	$C\max(\lvert z-y_i\rvert-\epsilon, 0)^2$	2	$\epsilon\lvert\alpha_i\rvert - \alpha y_i + \frac{1}{4C}\alpha^2$
Least-square regression	$C(z-y_i)^2$	2	$-\alpha y_i + \frac{1}{4C}\alpha^2$

Table 1: Summary of popular ERM problems for binary classification (the range of $y = \{1,-1\}$) and for regression (the range of $y = R$), where our approach is applicable. Our approach is also applicable to the extensions of these methods for multi-class classification and structured prediction.

One can observe that the dual objectives of the hinge loss and the squared-hinge loss SVMs are both quadratic, hence we can apply the exact line search approach in (19) with very low cost by utilizing the $\Delta\boldsymbol{v}$ and $\boldsymbol{v}$ vectors.

Another widely used classification model is logistic regression, where

$$\xi_i(z) \coloneqq C\log(1+\exp(-y_i z)).$$

It can then be shown that the logistic loss is infinitely differentiable, and its gradient is Lipschitz continuous. Thus, Assumption 2 is satisfied.

An analogy of SVM to regression is support vector regression (SVR) by [3, 45] such that the g function is the same and given $C > 0$ and $\epsilon \geq 0$,

$$\xi_i(z) \coloneqq \begin{cases} C\max(|z-y_i|-\epsilon, 0), & \text{or} \\ C\max(|z-y_i|-\epsilon, 0)^2, \end{cases}$$

with $y_i \in \mathbf{R}$ for all i. Similar to the case of SVM, the first case satisfies Assumption 3[6] and the latter satisfies Assumption 2. Often the first variant is called SVR and the second variant is called L2-loss SVR. Note that the degenerate case of $\epsilon = 0$ corresponds to the absolute-deviation loss and the least-square loss. In the case of the least-square loss, we again can use the exact line search approach because the objective is a quadratic function.

Note that one can also replace g with other strongly convex functions, but it is possible that (21) is not satisfied. In this case, one can establish some sublinear convergence rates by applying similar techniques in our analysis, but we omit these results to keep the description straightforward.

A short summary of various ξ_i's we discussed in this section is in Table 1.

6.2 Multi-class Classification

For the case of multi-class classification models, we assume without loss of generality that $c_i \equiv T$ for some $T > 1$, and $y_i \in \{1,\dots,T\}$ for all i. The

[6] Up to an equivalent reformulation of the dual problem by setting $\boldsymbol{\alpha} = \boldsymbol{\alpha}^+ - \boldsymbol{\alpha}^-$ and $\boldsymbol{\alpha}^+, \boldsymbol{\alpha}^- \geq 0$.

first model we consider is the multi-class SVM model proposed by [8]. Given an original feature vector $\boldsymbol{x}_i \in \mathbf{R}^{\tilde{n}}$, the data matrix X_i is defined as $(I_T - \boldsymbol{e}_{y_i}\mathbf{1}^T) \otimes \boldsymbol{x}_i$, where I_T is the T by T identity matrix, $\boldsymbol{e}_i$ is the unit vector of the i-th coordinate, $\mathbf{1}$ is the vector of ones, and $\otimes$ denotes the Kronecker product. We then get that $n = T\tilde{n}$, and the multi-class SVM model uses

$$\begin{aligned} g(\boldsymbol{w}) &:= \frac{1}{2}\|\boldsymbol{w}\|^2, \\ \xi_i(\boldsymbol{z}) &:= C \max\left(\max_{1\leq j\leq T} 1 - z_i, 0\right). \end{aligned} \tag{37}$$

From the first glance, this ξ seems to be not even Lipschitz continuous. However, its dual formulation is

$$\begin{aligned} \min_{\boldsymbol{\alpha}} \quad & \frac{1}{2}\|X\boldsymbol{\alpha}\|^2 + \sum_{i=1}^{l}\sum_{j\neq y_i} (\boldsymbol{\alpha}_i)_j \\ \text{subject to} \quad & \boldsymbol{\alpha}_i^T \mathbf{1} = 0, i = 1, \dots, l, \\ & (\boldsymbol{\alpha}_i)_j \leq 0, \forall j \neq y_i, i = 1, \dots, l, \\ & (\boldsymbol{\alpha}_i)_{y_i} \leq C, i = 1, \dots, l, \end{aligned} \tag{38}$$

showing the boundedness of the dual variables $\boldsymbol{\alpha}$. Thus, the primal variable $\boldsymbol{w}(\boldsymbol{\alpha}) = X\boldsymbol{\alpha}$ also lies in a bounded area. Therefore, $\xi_i(X_i^T \boldsymbol{w}(\boldsymbol{\alpha}))$ also has a bounded domain, indicating that by compactness we can find $L \geq 0$ such that this continuous function is Lipschitz continuous within this domain. Moreover, the formulation (38) satisfies the form (35), so (21) holds by Lemma 1. Thus, Assumption 3 is satisfied. Note that in this case the objective of (38) is a quadratic function so once again we can apply the exact line search method on this problem.

As an analogy of SVM, one can also use the squared-hinge loss for multi-class SVM [19].

$$\xi_i(\boldsymbol{z}) := C \max\left(\max_{1\leq j\leq T} 1 - z_i, 0\right)^2. \tag{39}$$

The key difference to the binary case is that the squared-hinge loss version of multi-class SVM does not possess a differentiable objective function. We need to apply a similar argument as above to argue the Lipschitzness of ξ. The dual formulation from the derivation in [19] is

$$\begin{aligned} \min_{\boldsymbol{\alpha}} \quad & \frac{1}{2}\|X\boldsymbol{\alpha}\|^2 + \sum_{i=1}^{l}\sum_{j\neq y_i} (\boldsymbol{\alpha}_i)_j + \sum_{i=1}^{l} \frac{((\boldsymbol{\alpha}_i)_{y_i})^2}{4C} \\ \text{subject to} \quad & \boldsymbol{\alpha}_i^T \mathbf{1} = 0, i = 1, \dots, l, \\ & (\boldsymbol{\alpha}_i)_j \leq 0, \forall j \neq y_i, i = 1, \dots, l, \end{aligned}$$

suggesting that each coordinate of $\boldsymbol{\alpha}$ is only one-side-bounded, so this is not the case that $\boldsymbol{\alpha}$ lies explicitly in a compact set. However, from the constraints and the objective, we can see that given any initial point $\boldsymbol{\alpha}^0$, the level set $\{\boldsymbol{\alpha} \mid f(\boldsymbol{\alpha}) \leq f(\boldsymbol{\alpha}^0)\}$ is compact. Because our algorithm is a descent method, throughout the optimization process, all iterates lie in this compact set. This again indicates that $\boldsymbol{w}(\boldsymbol{\alpha})$ and $X_i^T \boldsymbol{w}(\boldsymbol{\alpha})$ are within a compact set, proving the Lipschitzness of ξ_i within this set. The condition (21) is also satisfied following the same argument for the hinge-loss case. Therefore, Assumption 3 still holds, and it is obvious that we can use the exact line search method here as well.

We can also extend the logistic regression model to the multi-class scenario. The loss function, usually termed as multinomial logistic regression or maximum entropy, is defined as

$$\xi_i(\boldsymbol{z}) \coloneqq -\log\left(\frac{\exp(z_{y_i})}{\sum_{k=1}^{T}\exp(z_k)}\right).$$

It is not hard to see that Assumption 2 holds for this problem. For more details of its dual problem and an efficient local sub-problem solver, interested readers are referred to [50].

6.3 Structured Prediction Models

In many real-world applications, the decision process involves making multiple predictions over a set of interdependent output variables, whose mutual dependencies can be modeled as a structured object such as a linear chain, a tree, or a graph. As an example, consider recognizing a handwritten word, where characters are output variables and together form a sequence structure. It is important to consider the correlations between the predictions of adjacent characters to aid the individual predictions of characters. A family of models designed for such problems are called structured prediction models. In the following, we discuss how to apply Algorithm 2 in solving SSVM [44,42], a popular structured prediction model.

Different from the case of binary and multi-class classifications, the output in a structured prediction problem is a set of variables $\boldsymbol{y}_i \in \mathcal{Y}_i$, and $\mathcal{Y}_i$ is the set of all feasible structures. The sizes of the input and the output variables are often different from instance to instance. For example, in the handwriting recognition problem, each element in $\boldsymbol{y}$ represents a character and $\mathcal{Y}$ is the set of all possible words. Depending on the number of characters in the words, the sizes of inputs and outputs vary.

Given a set of observations $\{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^{l}$, SSVM solves

$$\begin{aligned} \min_{\boldsymbol{w},\boldsymbol{\psi}} \quad & \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{l}\ell(\psi_i) \\ \text{subject to} \quad & \boldsymbol{w}^T\phi(\boldsymbol{y},\boldsymbol{y}_i,\boldsymbol{x}_i) \geq \Delta(\boldsymbol{y}_i,\boldsymbol{y}) - \psi_i, \quad \forall \boldsymbol{y} \in \mathcal{Y}_i, \quad i = 1,\dots,l, \end{aligned} \tag{40}$$

where $C > 0$ is a predefined parameter, $\phi(\boldsymbol{y},\boldsymbol{y}_i,\boldsymbol{x}_i) = \Phi(\boldsymbol{x}_i,\boldsymbol{y}_i) - \Phi(\boldsymbol{x}_i,\boldsymbol{y})$, and $\Phi(\boldsymbol{x},\boldsymbol{y})$ is the generated feature vector depending on both the input $\boldsymbol{x}$

and the output structure $\boldsymbol{y}$. By defining features depending on the output, one can encode the output structure into the model and learn parameters to model the correlation between output variables. The constraints in problem (40) specify that the difference between the score assigned to the correct output structure should be higher than a predefined scoring metric $\Delta(\boldsymbol{y}, \boldsymbol{y}_i) \geq 0$ that represents the distance between output structures. If the constraints are not satisfied, then a penalty term ψ_i is introduced to the objective function, where $\ell(\psi)$ defines the loss term. Similar to the binary and multi-class classifications cases, common choices of the loss functions are the L2 loss and the L1 loss. The SSVM problem (40) fits in our framework, depending on the definition of the features, one can define X_i to encode the output $\boldsymbol{y}$. One example is to set every column of X_i as a vector of the form $\phi(\boldsymbol{y}, \boldsymbol{y}_i, \boldsymbol{x}_i)$ with different $\boldsymbol{y} \in \mathcal{Y}_i$, and let $\xi_i(X_i^T \boldsymbol{w})$ in problem (1) be

$$\xi_i(\boldsymbol{z}) = C \max_{\boldsymbol{y} \in \mathcal{Y}_i} \ell(\Delta(\boldsymbol{y}_i, \boldsymbol{y}) - \boldsymbol{z}_{\boldsymbol{y}}). \tag{41}$$

Here, we use the order of $\boldsymbol{y}$ appeared in the columns of X_i as the enumerating order for the coordinates of $\boldsymbol{z}$.

We consider solving problem (40) in its dual form [44]. One can clearly see the similarity between (41) and the multi-class losses (37) and (39), where the major difference is that the value 1 in the multi-class losses is replaced by $\Delta(\boldsymbol{y}_y, \boldsymbol{y})$. Thus, it can be expected that the dual problem of SSVM is similar to that of multi-class SVM. With the L1 loss, the dual problem of (40) can be written as,

$$\begin{aligned} \min_{\boldsymbol{\alpha}} \quad & \frac{1}{2}\|X\boldsymbol{\alpha}\|^2 - \sum_{i=1}^{l} \sum_{\boldsymbol{y} \in \mathcal{Y}_i, \boldsymbol{y} \neq \boldsymbol{y}_i} \Delta(\boldsymbol{y}_i, \boldsymbol{y})(\boldsymbol{\alpha}_i)_{\boldsymbol{y}_i} \\ \text{subject to} \quad & \boldsymbol{\alpha}_i^T \mathbf{1} = 0, i = 1, \dots, l, \\ & (\boldsymbol{\alpha}_i)_{\boldsymbol{y}} \leq 0, \forall \boldsymbol{y} \in \mathcal{Y}_i, \boldsymbol{y} \neq \boldsymbol{y}_i, i = 1, \dots, l, \\ & (\boldsymbol{\alpha}_i)_{\boldsymbol{y}_i} \leq C, i = 1, \dots, l. \end{aligned} \tag{42}$$

With the L2 loss, the dual of (40) is

$$\begin{aligned} \min_{\boldsymbol{\alpha}} \quad & \frac{1}{2}\|X\boldsymbol{\alpha}\|^2 - \sum_{i=1}^{l} \sum_{\boldsymbol{y} \neq \boldsymbol{y}_i} \Delta(\boldsymbol{y}_i, \boldsymbol{y})(\boldsymbol{\alpha}_i)_{\boldsymbol{y}_i} + \sum_{i=1}^{l} \frac{((\boldsymbol{\alpha}_i)_{\boldsymbol{y}_i})^2}{4C} \\ \text{subject to} \quad & \boldsymbol{\alpha}_i^T \mathbf{1} = 0, i = 1, \dots, l, \\ & (\boldsymbol{\alpha}_i)_{\boldsymbol{y}} \leq 0, \forall \boldsymbol{y} \in \mathcal{Y}_i, \boldsymbol{y} \neq \boldsymbol{y}_i, i = 1, \dots, l. \end{aligned} \tag{43}$$

As the dual forms are almost identical to that shown in Section 6.2, it is clear that all the analysis and discussion can be directly used here.

The key challenge of solving problems (42) and (43) is that for most applications, the size of $\mathcal{Y}_i$ and thus the dimension of $\boldsymbol{\alpha}$ is exponentially large (with respect to the length of $\boldsymbol{x}_i$), so optimizing over all variables is unrealistic. Efficient dual methods [44,16,5] maintain a small working set of dual variables to optimize such that the remaining variables are fixed to be zero. These methods then iteratively enlarge the working set until the problem is well-optimized.[7] The working set is selected using the sub-gradient of (40) with respect to the current iterate. Specifically, for each training instance $\boldsymbol{x}_i$, we add the dual variable $\alpha_{i,\hat{\boldsymbol{y}}}$ corresponding to the structure $\hat{\boldsymbol{y}}$ into the working set, where

$$\hat{\boldsymbol{y}} = \arg\max_{\boldsymbol{y}\in\mathcal{Y}_i} \boldsymbol{w}^T\phi(\boldsymbol{y},\boldsymbol{y}_i,\boldsymbol{x}_i) - \Delta(\boldsymbol{y}_i,\boldsymbol{y}). \tag{44}$$

Once $\boldsymbol{\alpha}$ is updated, we update $\boldsymbol{w}$ accordingly. We call the step of computing eq. (44) "inference", and call the part of optimizing Eq. (42) or (43) over a fixed working set "learning". When training SSVM in a distributed manner, the learning step involves communication across machines. Therefore, inference and learning steps are both expensive. Our algorithm can be applied in the learning step to reduce the rounds of communication, and linear convergence rate for solving the problem under a fixed working set can be obtained.

SSVM is an extension of multi-class SVM for structured prediction. Similarly, conditional random fields (CRF) [17] extends multinomial logistic regression. The loss function in CRF is defined as the negative log-likelihood:

$$\xi_i(\boldsymbol{z}) := -\log\left(\frac{\exp(\boldsymbol{z}_{\boldsymbol{y}_i})}{\sum_{\boldsymbol{y}\in\mathcal{Y}_i}\exp(\boldsymbol{z}_{\boldsymbol{y}})}\right). \tag{45}$$

Similar to multinomial logistic regression, Assumption 2 holds for (45).

7 Experiments

We conduct experiments on different ERM problems to examine the efficiency of variant realizations of our framework. The problems range from binary classification (i.e., $c_i \equiv 1$) to problems with complex output structures (i.e., each c_i is different), and from that exact line search can be conducted to that backtracking using Algorithm 1 is applied. For each problem, we compare our method with the state of the art, and the dataset is partitioned evenly across machines in terms of the number of data points without randomly shuffling the instances in advance, so it is possible that the data distributions on different machines vary.

For the case of $c_i \equiv 1$, we consider two linear classification tasks. To evaluate the situation of larger c_i, we take SSVM as the exemplifying application.

7.1 Binary Linear Classification

The proposed framework is suitable for training large machine learning models on data where numbers of instances and features are both large. It is especially

[7] This approach is related to applying the cutting-plane methods to solve the primal problem (40) [44, 14].

Table 2: Data statistics.

Data set	#instances (l)	#features (n)	#nonzeros
webspam	350,000	16,609,143	1,304,697,446
url	2,396,130	3,231,961	277,058,644
KDD2010-b	19,264,097	29,890,095	566,345,888

useful in a practical setting where data instances are stored distributedly on multiple machines. This setting is common on web data due to efficiency and privacy concerns. We therefore consider the following large-scale datasets in our experiments.

– webspam [?] is a binary classification task aiming at detecting if a web page is created to manipulate search engines. We use bag-of-words model with n-gram ($n \leq 3$) to extract features.
– url [?] detects malicious URLs based on their lexical and host-based features.
– KDD2010-b is the dataset used in KDD Cup 2010 with the goal to predict students' performance based on logs of their interaction with an educational system. We follow [?] to extract features.

The statistics of the data are summarized in Table 2.[8]

We consider both linear SVM and L2-regularized logistic regression discussed in Section 6.1. The comparison criteria are the relative primal and dual objective distances to the optimum, respectively defined as

$$\left| \frac{f^P\left(\boldsymbol{w}\left(\boldsymbol{\alpha}^t\right)\right) - f^*}{f^*} \right|, \quad \left| \frac{f(\boldsymbol{\alpha}^t) - (-f^*)}{f^*} \right|, \tag{46}$$

where f^* is the optimum we obtained approximately by running our algorithm with a tight stopping condition. Note that the optimum for the dual and the primal problems are identical except the flip of the sign, according to strong duality. We examine the relation between these values and the training time. We fix $C = 1$ in this experiment. The distributed environment is a cluster of 16 machines connected through MPI.

We compare the methods below whenever applicable.

– BDA: the Block-Diagonal Approximation method proposed in this paper. For the dual SVM problem, we utilize its quadratic objective to conduct exact line search while backtracking line search is used with the parameters being $\tau = 10^{-2}, \beta = 0.5$ for logistic regression. For B_t in (14), we use $a_1^t \equiv 1$ for all problems, as it is the closest block-diagonal approximation of the Hessian. We set $a_2^t = 10^{-3}$ in the hinge-loss SVM problem and $a_2^t = 0$ in the other two whose dual objectives are strongly convex.
– DisDCA [49]: we use the practical variant for it outperforms th basic variant empirically. Moreover, experimental result in [30] showed that this algorithm (under a different name CoCoA+) is faster than DSVM-AVE, and the best solver for the local sub-problems is indeed SDCA used in [49].

[8] Downloaded from http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/.

– L-CommDir [21]: a state-of-the-art distributed primal ERM solver that has been shown to empirically outperform existing distributed primal approaches. We take the experimental setting in [21] to use historical information from the latest five iterations.
– TRON [54,12]: a distributed implementation for ERM problems of the trust-region truncated Newton method proposed by [41]

All methods are implemented in C++. We use the implementation of L-CommDir and TRON in the package MPI-LIBLINEAR 2.11.[9] These two methods require differentiability of the primal objective, so we apply them only on squared-hinge loss SVM and logistic regression problems. We implement DisDCA and BDA with the local sub-problem solver being the random permutation cyclic coordinate descent (RPCD) for dual SVM [13] and for dual logistic regression [50]. Note that the original solver in [49] is the dual stochastic coordinate descent algorithm in [39] that samples the coordinates with replacement, but it has been shown in [39] that empirically RPCD is faster, and therefore we apply it in DisDCA as well. At each iteration, we run one epoch of RPCD on each machine, namely we pass through the whole dataset once, before communication. This setting ensures that DisDCA and BDA have computation-to-communication ratios similar to that of L-CommDir and TRON, so our results represent both a comparison for the training time and a comparison for the number of communication rounds.

The comparison of the dual and primal objectives are shown in Figures 1-3. For webspam and url that are easier to solve, we present the result of running different algorithms for 500 seconds. For the more difficult problem KDD2010-b, we run all algorithms for $10,000$ seconds.

We first discuss the dual objectives. We can see that our approach is always better than state of the art for the dual problem. The difference is more significant in the SVM problems, showing that low-cost exact line search has its advantage over backtracking, while backtracking is still better than the fixed step size scheme. The reason behind is that although the approach of DisDCA provides a safe upper bound model for the objective difference such that the local updates can be directly applied to ensure the objective decrease, this upper bound might be too conservative as suggested by [30], but more aggressive upper bound modelings might be computationally impractical to obtain. On the other hand, our approach provides an efficient way to dynamically estimate how aggressive the updates can be, depending to the current iterate. Therefore, the objective can decrease faster as the update is more aggressive but still safe in terms of ensuring sufficient objective value decrease.

Now we turn to the primal objectives. Note that the step-like behavior of BDA is from that we use the best primal objective up to the current iterate discussed in Section 3.4. Although aggressive step sizes in BDA results in less stable primal objective progress especially in the beginning, we observe that BDA still reaches lower primal objective faster than DisDCA, and the behavior of the early stage is less important. For the case of hinge-loss SVM, BDA is

[9] http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/distributed-liblinear/.

always the best, and note that only dual approaches are feasible for hinge loss as it is not differentiable. When it comes to squared-hinge loss SVM, in which case exact line search for the dual problem can still be conducted, BDA outperforms all primal and dual approaches. The dual problem of logistic regression is not a quadratic one, hence we cannot easily implement exact line search and need to resort to the backtracking approach in Algorithm 1. We can see that for this problem, L-CommDir has an advantage in the later stage of optimization, while BDA and DisDCA are competitive till a medium precision, which is usually enough for linear classification tasks. In most cases, TRON is the slowest method.

7.2 Structured Learning

We perform experiments on two benchmark tasks for structured prediction, part-of-speech tagging (POS) and dependency parsing (DEP). For both tasks, we use the Wall Street Journal portion of the Penn Treebank [31] with the standard split for training (section 02-21), development (section 22), and test (section 23). POS is a sequential labeling task, where we aim at learning part-of-speech tags assigned to each word in a sentence. Each tag assignment (there are 45 possible tag assignments) depends on the associated word, the surrounding words, and their part-of-speech tags. The inference in POS is solved by the Viterbi algorithm [46]. We evaluate our model by the per-word tag accuracy. For DEP, the goal is to learn, for each sentence, a tree structure which describes the syntactic dependencies between words. We use the graph-based parsing formulation and the features described in [33], where we find the highest scoring parse using the Chu-Liu-Edmonds algorithm [7,9]. We evaluate the parsing accuracy using the unlabeled attachment score, which measures the fraction of words that have correctly assigned parents.

We compare the following algorithms using eight nodes in a local cluster. All algorithms are implemented in JAVA, and the distributed platform is MPI.

- BDA: the proposed algorithm. We take $a_1^t \equiv K$ and $a_2^t \equiv 10^{-3}$ as $a_1^t \equiv 1$ is less stable in the primal objectives, which is essential for the sub-problem solver in this application.
- ADMM-Struct: distributed alternating directions method of multiplier discussed in [4].
- Distributed Perceptron: a parallel structured perceptron algorithm described in [32].
- Simple average: each machine trains a separate model using the local data. The final model is obtained by averaging all local models.

For BDA and ADMM-Struct, the problem considered is SSVM in (40) with L2 loss. Distributed Perceptron, on the other hand, solves a similar but different problem such that no regularization is involved. We set $C = 0.1$ for SSVM. Empirical experience suggests that structured SVM is not sensitive to C, and the model trained with $C = 0.1$ often attains reasonable test performance.

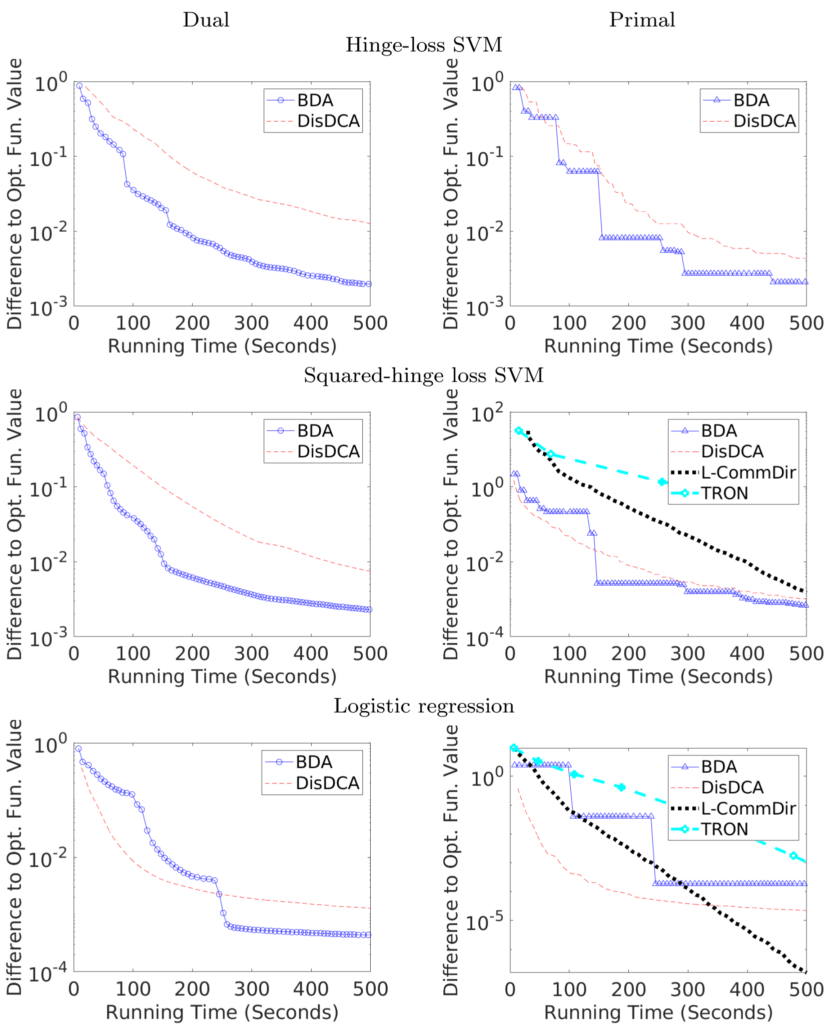


Fig. 1: Comparison of different algorithms for optimizing the ERM problem on webspam with $C = 1$. We show training time v.s. relative difference of the objectives to the optimal function value.

Both ADMM-Struct and BDA decompose the original optimization problem into sub-problems, and we solve the sub-problems by the dual coordinate descent solver for L2-loss SSVM proposed in [5], which is shown to be empirically efficient comparing to other existing methods. By solving the sub-problems using the same optimizers, we can investigate the algorithmic difference between ADMM-Struct and BDA. For all algorithms, we fix the number of passes through all instances to make inferences between any two rounds of commu-

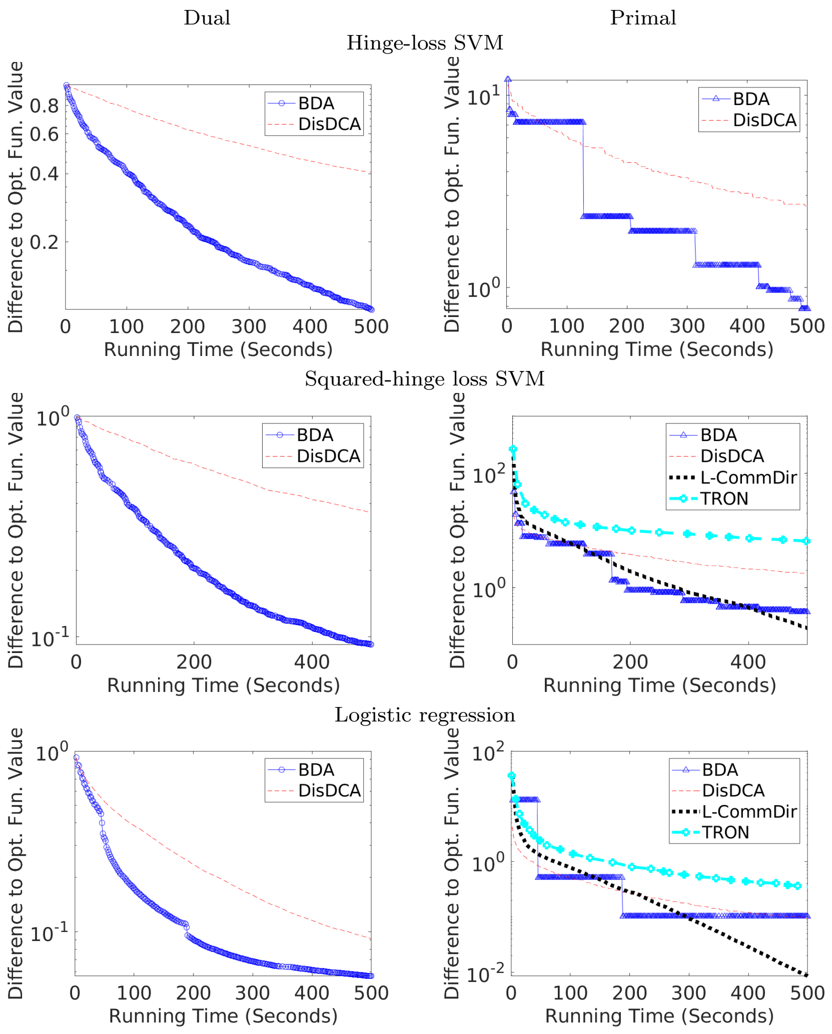


Fig. 2: Comparison of different algorithms for optimizing the ERM problem on url with $C = 1$. We show training time v.s. relative difference of the objectives to the optimal function value.

nication to be one, so that the number of inference rounds is identical to the number of communication rounds. Although it is possible to alter the number of inferences between two rounds of communication (or the number of communication between two rounds of inferences) to obtain a faster running time, fine-tuning this parameter is not realistic for users because this parameter does not affect the prediction performance, and thus there is no reason to spend time retrain the model several times. For BDA and ADMM-Struct, each time in

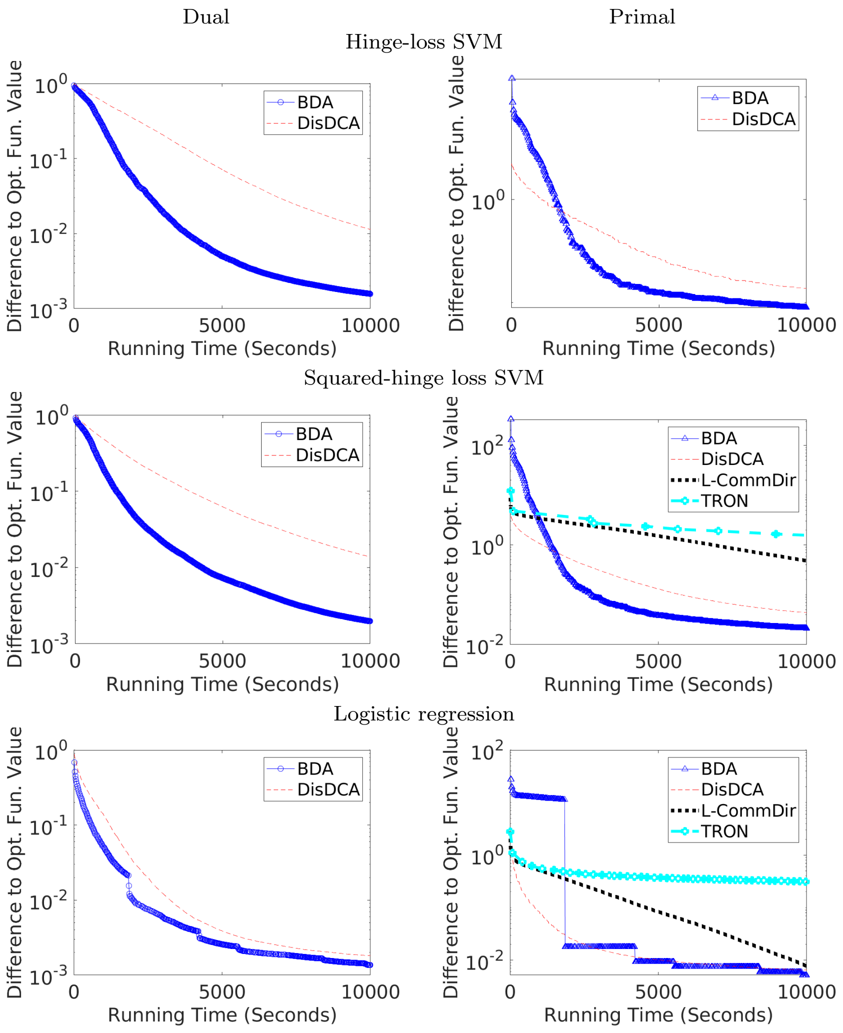
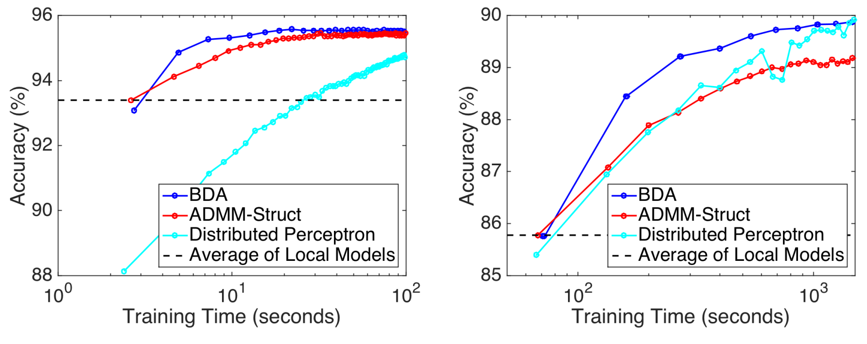


Fig. 3: Comparison of different algorithms for optimizing the ERM problem on KDD2010-b with $C = 1$. We show training time v.s. relative difference of the objectives to the optimal function value.

solving the local sub-problem with a fixed working set, we let the local RPCD solver pass through the local instances ten times. We note that this number of iterations may also affect the convergence speed but we do not fine-tune this parameter for the same reason above. For ADMM-Struct, the weight for the penalty term in the augmented Lagrangian also affects the convergence speed.[10] Instead of fine-tuning it, a fixed value of 1.0 is used. Note that since

[10] See, for example, [4] for details.



(a) POS (b) DEP

Fig. 4: Comparison between different algorithms for structured learning using eight nodes. Training time is in *log scale*.

Distributed Perceptron and BDA/ADMM-Struct consider different problems, instead of showing objective function values, we compare the test performance along training time of these methods.

Figure 4 shows the results. The x-axis is in *log-scale*. Although averaging local classifiers achieves reasonable performance, all other methods improve the performance of the models with multiple rounds of communications. This indicates that training models jointly on all parts of data is necessary. Among different algorithms, BDA performs the best in both tasks. It achieves the final accuracy performance (indicated when the accuracy stops improving) with shorter training time comparing to other approaches. This result confirms that BDA enjoys a fast convergence rate.

7.3 Line Search

The major difference between our approach and most other dual distributed optimization methods for ERM is the line search part. In this subsection we examine the step sizes obtained in practice and show that there is still a gap between (31) and (32) as the Lipschitz parameter can be smaller in a local region. We also investigate the empirical cost of line search. For this investigation, we use the information from the L2-regularized linear classification experiments in Section 7.1.

In Table 3, we show the proportion of time spent on line search to the overall training time. As the results indicate, the cost of line search is relatively low in comparison to solving the local sub-problem and communicating $\Delta \boldsymbol{v}$. Note that the cost of line search for hinge and squared-hinge loss is independent to the final step size, as exact line search instead of backtracking is applied.

Table 3: Percentage of training time spent on line search. For hinge and squared-hinge loss, Variant II of Algorithm 2 is used, while Variant I is applied for logistic loss.

Loss	webspam	url	KDD2010-b
Hinge	6.25%	7.82%	2.05%
Squared-hinge	9.87%	10.64%	7.50%
Logistic	0.47%	5.00%	4.55%

7.4 Speedup

Finally, we examine the practical speedup of BDA. We pick webspam on L2-loss SVM as a representative example for this experiment. We run different algorithms on $\{1, 2, 4, 8, 16\}$ machines and see how the training time and the overall running time (training time plus data loading time) differ. We record the time for (46) to reach 10^{-2} in Figure 5. The left column represents the time measured using the primal objective, while the right column represents that using the dual objective. We can see that when it comes to the training time, TRON has a better speedup because its algorithmic behavior is invariant of how the data are distributed, while BDA still enjoys better speedup than the state of the art dual solver DisDCA.

When the data loading time is combined, we can see that BDA has the best speedup, and the reason can be seen from the third row of time profiling. We see that the bottleneck in the single-machine case is data loading which is embarrassingly parallel, and the training time of BDA is insignificant in comparison with the I/O time. Therefore, although the training time speedup of BDA is not that significant, the running time speedup is very promising. Another reason we cannot obtain good speedup in the training time is that the single-machine case is already very efficient in comparison with TRON, so it is rather difficult to have further improvement.

8 Discussion

As Section 4.4 suggests, if the block-diagonal matrix B_t is a tight approximation to the Hessian of G^*, BDA is expected to enjoy fast convergence. To achieve so, we might partition the data in a better way such that those off diagonal-block entries in the matrix $X^T X$ are as small as possible, then the Hessian will also have smaller off-diagonal terms. However, repartitioning the data across machines involves a significant amount of data transmission, and designing an efficient mechanism to split the data into blocks with desirable properties is challenging. One practically feasible scenario is the case where the data points are streamed in and partitioned in an online fashion.

Notice that in (32), having a larger step size while maintaining a large μ_B leads to fast convergence. However, balancing these two factors is not an easy

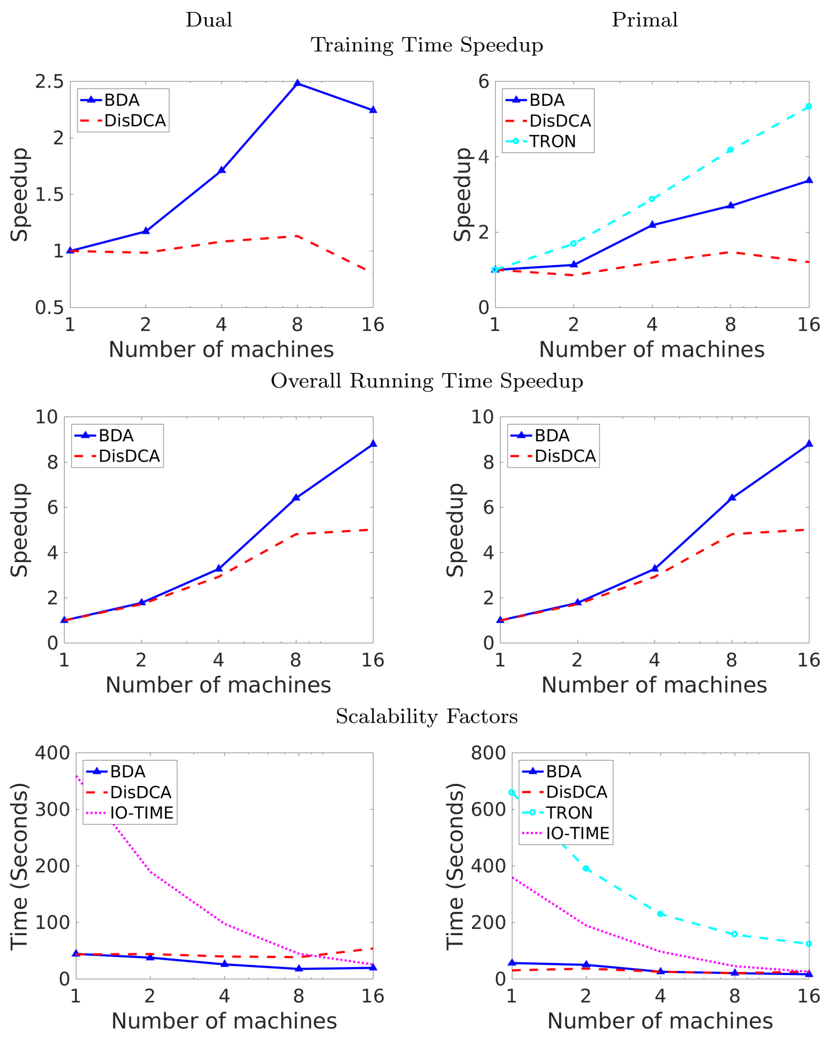


Fig. 5: Speedup of different algorithms for training L2-loss SVM on webspam with $C = 1$.

task. One potential heuristic is to adjust a_1^t and a_2^t dynamically based on the step size in the previous iteration.

One limitation of our current approach is that the algorithm does not scale strongly with the number of machines when the data size is fixed. If the number of machines increases, B_t will contain more zero entries. This means the algorithm will be closer to a proximal gradient method and converge slowly. This is inevitable for all distributed dual optimizers we discussed in Section 5. However, in many real applications, distributed optimization techniques are used to protect privacy or handle distributional data. In such applications, repartitioning data is costly and may not be feasible. Therefore, the number of machines is predefined and practitioners are concerned more about how to make the optimization procedure more efficient given the fixed number of machines and the fixed data partitions, but not how to use more machines for the same data to speed up the training process.

As mentioned in Section 5, just like [53] applied existing acceleration techniques on top of DisDCA, our algorithm can also be combined with the acceleration techniques proposed by [40,27] to obtain a faster algorithm, and we expect using our algorithm instead of DisDCA will be faster than the result in [53] as our algorithm is faster than DisDCA in practice. This comparison will be an interesting future work.

9 Conclusions

In this work, we proposed a distributed optimization framework for the dual problem of regularized empirical risk minimization. Our theoretical results show linear convergence for both the dual problem and the corresponding primal problem for a variety class of popular problems whose dual problem is non-strongly convex. Our analysis further shows that when the sub-problem can serve as a preconditioner to improve the problem condition, much better convergence speed can be expected. Our approach is most powerful when it is difficult to directly solve the primal problem. Experimental results show that our method outperforms state-of-the-art distributed dual approaches for regularized empirical risk minimization, and is competitive to cutting-edge distributed primal methods when those primal methods are feasible.

Acknowledgement. The authors would like to thank the action editor and the anonymous reviewers for their valuable comments and Dan Roth, Shyam Upadhyay, Chih-Jen Lin, Cho-Jui Hsieh, Martin Jaggi, and Shai Shalev-Shwartz for their feedback and suggestions on the early version of this paper. This work was supported in part by NSF Grant IIS-1760523.

References

1. Bach, F.: Duality between subgradient and conditional gradient methods. SIAM Journal on Optimization **25**(1), 115–129 (2015)
2. Bolte, J., Nguyen, T.P., Peypouquet, J., Suter, B.W.: From error bounds to the complexity of first-order descent methods for convex functions. Mathematical Programming **165**(2), 471–507 (2017)
3. Boser, B.E., Guyon, I., Vapnik, V.: A training algorithm for optimal margin classifiers. In: Proceedings of the Fifth Annual Workshop on Computational Learning Theory, pp. 144–152. ACM Press (1992)
4. Boyd, S., Parikh, N., Chu, E., Peleato, B., Eckstein, J.: Distributed optimization and statistical learning via the alternating direction method of multipliers. Foundations and Trends in Machine Learning **3**(1), 1–122 (2011)

5. Chang, M.W., Yih, W.t.: Dual coordinate descent algorithms for efficient large margin structural learning. Transactions of the Association for Computational Linguistics (2013)
6. Chen, W., Wang, Z., Zhou, J.: Large-scale L-BFGS using MapReduce. In: Advances in Neural Information Processing Systems 27, pp. 1332–1340 (2014)
7. Chu, Y.J., Liu, T.H.: On shortest arborescence of a directed graph. Scientia Sinica **14**(10), 1396 (1965)
8. Crammer, K., Singer, Y.: On the learnability and design of output codes for multiclass problems. Machine Learning (2–3), 201–233 (2002)
9. Edmonds, J.: Optimum branchings. Journal of Research of the national Bureau of Standards B **71**(4), 233–240 (1967)
10. Gallant, S.I.: Perceptron-based learning algorithms. Neural Networks, IEEE Transactions on **1**(2), 179–191 (1990)
11. Hiriart-Urruty, J.B., Lemaréchal, C.: Fundamentals of convex analysis. Springer Science & Business Media (2001)
12. Hsia, C.Y., Zhu, Y., Lin, C.J.: A study on trust region update rules in newton methods for large-scale linear classification. In: Asian Conference on Machine Learning, pp. 33–48 (2017)
13. Hsieh, C.J., Chang, K.W., Lin, C.J., Keerthi, S.S., Sundararajan, S.: A dual coordinate descent method for large-scale linear SVM. In: Proceedings of the Twenty Fifth International Conference on Machine Learning (2008)
14. Joachims, T., Finley, T., Yu, C.N.J.: Cutting-plane training of structural SVMs. Machine Learning **77**(1), 27–59 (2009)
15. Kurdyka, K.: On gradients of functions definable in o-minimal structures. In: Annales de l'institut Fourier, vol. 48, pp. 769–783 (1998)
16. Lacoste-Julien, S., Jaggi, M., Schmidt, M., Pletscher, P.: Block-coordinate Frank-Wolfe optimization for structural svms. In: Proceedings of the Thirtieth International Conference on Machine Learning (2013)
17. Lafferty, J., McCallum, A., Pereira, F.: Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In: Proceedings of the Eighteenth International Conference on Machine Learning (2001)
18. Lee, C.p., Chang, K.W., Upadhyay, S., Roth, D.: Distributed training of structured SVM. In: NIPS Workshop on Optimization for Machine Learning (2015)
19. Lee, C.p., Lin, C.J.: A study on L2-loss (squared hinge-loss) multi-class SVM. Neural Computation **25**(5), 1302–1323 (2013)
20. Lee, C.p., Roth, D.: Distributed box-constrained quadratic optimization for dual linear SVM. In: Proceedings of the Thirty Second International Conference on Machine Learning (2015)
21. Lee, C.p., Wang, P.W., Chen, W., Lin, C.J.: Limited-memory common-directions method for distributed optimization and its application on empirical risk minimization. In: Proceedings of SIAM International Conference on Data Mining (2017)
22. Lee, C.p., Wright, S.J.: Inexact successive quadratic approximation for regularized optimization. Computational Optimization and Applications **72**, 641–674 (2019)
23. Lee, C.p., Wright, S.J.: Random permutations fix a worst case for cyclic coordinate descent. IMA Journal on Numerical Analysis **39**(3), 1246–1275 (2019)
24. Lee, J.D., Lin, Q., Ma, T., Yang, T.: Distributed stochastic variance reduced gradient methods and a lower bound for communication complexity. Tech. rep. (2015). ArXiv:1507.07595
25. Lee, J.D., Sun, Y., Saunders, M.A.: Proximal Newton-type methods for minimizing composite functions. SIAM Journal on Optimization **24**(3), 1420–1443 (2014)
26. Lin, C.Y., Tsai, C.H., Lee, C.p., Lin, C.J.: Large-scale logistic regression and linear support vector machines using Spark. In: Proceedings of the IEEE International Conference on Big Data, pp. 519–528 (2014)
27. Lin, H., Mairal, J., Harchaoui, Z.: A universal catalyst for first-order optimization. In: Advances in Neural Information Processing Systems 28, pp. 3384–3392 (2015)
28. Łojasiewicz, S.: Une propriété topologique des sous-ensembles analytiques réels. In: Les Équations aus Dérivées Partielles. Éditions du centre National de la Recherche Scientifique (1963)

29. Łojasiewicz, S.: Sur la géométrie semi-et sous-analytique. In: Annales de l'institut Fourier, vol. 43, pp. 1575–1595 (1993)
30. Ma, C., Konečný, J., Jaggi, M., Smith, V., Jordan, M.I., Richtárik, P., Takáč, M.: Distributed optimization with arbitrary local solvers. Optimization Methods and Software pp. 1–36 (2017)
31. Marcus, M.P., Marcinkiewicz, M.A., Santorini, B.: Building a large annotated corpus of English: The Penn Treebank. Computational linguistics **19**(2), 313–330 (1993)
32. McDonald, R., Hall, K., Mann, G.: Distributed training strategies for the structured perceptron. In: Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics (2010)
33. McDonald, R., Pereira, F., Ribarov, K., Hajič, J.: Non-projective dependency parsing using spanning tree algorithms. In: Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing (2005)
34. Necoara, I., Nesterov, Y., Glineur, F.: Linear convergence of first order methods for non-strongly convex optimization. Mathematical Programming **175**(1-2), 69–107 (2019)
35. Pechyony, D., Shen, L., Jones, R.: Solving large scale linear SVM with distributed block minimization. In: NIPS 2011 Workshop on Big Learning: Algorithms, Systems, and Tools for Learning at Scale (2011)
36. Peng, W., Zhang, H., Zhang, X.: Global complexity analysis of inexact successive quadratic approximation methods for regularized optimization under mild assumptions. Tech. rep. (2018)
37. Rockafellar, R.T.: Convex Analysis. Princeton University Press, Princeton, NJ, USA (1970)
38. Shalev-Shwartz, S., Zhang, T.: Proximal stochastic dual coordinate ascent. Tech. rep. (2012). ArXiv:1211.2717
39. Shalev-Shwartz, S., Zhang, T.: Stochastic dual coordinate ascent methods for regularized loss minimization. Journal of Machine Learning Research **14**, 567–599 (2013)
40. Shalev-Shwartz, S., Zhang, T.: Accelerated proximal stochastic dual coordinate ascent for regularized loss minimization. Mathematical Programming **155**(1-2), 105–145 (2016)
41. Steihaug, T.: The conjugate gradient method and trust regions in large scale optimization. SIAM Journal on Numerical Analysis **20**, 626–637 (1983)
42. Taskar, B., Guestrin, C., Koller, D.: Max-margin Markov networks. In: Advances in Neural Information Processing Systems 16, pp. 25–32 (2004)
43. Tseng, P., Yun, S.: A coordinate gradient descent method for nonsmooth separable minimization. Mathematical Programming **117**, 387–423 (2009)
44. Tsochantaridis, I., Joachims, T., Hofmann, T., Altun, Y.: Large margin methods for structured and interdependent output variables. Journal of Machine Learning Research **6**, 1453–1484 (2005)
45. Vapnik, V.: The Nature of Statistical Learning Theory. Springer-Verlag, New York, NY, USA (1995)
46. Viterbi, A.: Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. IEEE transactions on Information Theory **13**(2), 260–269 (1967)
47. Wang, P.W., Lin, C.J.: Iteration complexity of feasible descent methods for convex optimization. Journal of Machine Learning Research **15**, 1523–1548 (2014)
48. Wright, S.J., Lee, C.p.: Analyzing random permutations for cyclic coordinate descent. Tech. rep. (2017). URL http://www.optimization-online.org/DB_HTML/2017/06/6054.html.
49. Yang, T.: Trading computation for communication: Distributed stochastic dual coordinate ascent. In: Advances in Neural Information Processing Systems 26, pp. 629–637 (2013)
50. Yu, H.F., Huang, F.L., Lin, C.J.: Dual coordinate descent methods for logistic regression and maximum entropy models. Machine Learning **85**(1-2), 41–75 (2011)
51. Yuan, G.X., Ho, C.H., Lin, C.J.: Recent advances of large-scale linear classification. Proceedings of the IEEE **100**(9), 2584–2603 (2012)
52. Zhang, Y., Lin, X.: DiSCO: Distributed optimization for self-concordant empirical loss. In: Proceedings of the Thirty Second International Conference on Machine Learning (2015)
53. Zheng, S., Xia, F., Xu, W., Zhang, T.: A general distributed dual coordinate optimization framework for regularized loss minimization. Tech. rep. (2017). ArXiv:1604.03763

54. Zhuang, Y., Chin, W.S., Juan, Y.C., Lin, C.J.: Distributed Newton method for regularized logistic regression. In: Proceedings of the Pacific-Asia Conference on Knowledge Discovery and Data Mining (2015)
55. Zou, H., Hastie, T.: Regularization and variable selection via the elastic net. Journal of the Royal Statistical Society: Series B (Statistical Methodology) **67**(2), 301–320 (2005)

A Proofs

A.1 Proof of Lemma 1

Proof By [11, Part E, Theorem 4.2.1], if Assumption 2 holds, then $\boldsymbol{\xi}^*(\cdot)$ and hence f is $(1/\rho)$-strongly convex. We thus have that for any $\boldsymbol{\alpha}_1, \boldsymbol{\alpha}_2 \in \Omega$ and any $\lambda \in (0,1]$,

$$f\left(\lambda \boldsymbol{\alpha}_1 + (1-\lambda)\boldsymbol{\alpha}_2\right) \leq \lambda f(\boldsymbol{\alpha}_1) + (1-\lambda) f(\boldsymbol{\alpha}_2) - \frac{\lambda(1-\lambda)}{2\rho}\left\|\boldsymbol{\alpha}_1 - \boldsymbol{\alpha}_2\right\|^2,$$

which implies

$$f(\boldsymbol{\alpha}_1) - f(\boldsymbol{\alpha}_2) \geq \frac{1-\lambda}{2\rho}\|\boldsymbol{\alpha}_1 - \boldsymbol{\alpha}_2\|^2 + \frac{f\left(\boldsymbol{\alpha}_2 + \lambda\left(\boldsymbol{\alpha}_1 - \boldsymbol{\alpha}_2\right)\right) - f(\boldsymbol{\alpha}_2)}{\lambda}.$$

Let $\lambda \to 0^+$, we get

$$f(\boldsymbol{\alpha}_1) - f(\boldsymbol{\alpha}_2) \geq \frac{1}{2\rho}\|\boldsymbol{\alpha}_1 - \boldsymbol{\alpha}_2\|^2 + \boldsymbol{s}^T(\boldsymbol{\alpha}_1 - \boldsymbol{\alpha}_2), \forall \boldsymbol{s} \in \partial f(\boldsymbol{\alpha}_2).$$

By taking $\boldsymbol{\alpha}_2 = \boldsymbol{\alpha}$ with any $\boldsymbol{s} \in \partial f(\boldsymbol{\alpha})$ and minimizing both sides with respect to $\boldsymbol{\alpha}_1$ simultaneously, we get (21) as $\boldsymbol{s}$ is arbitrary.

A.2 Proof of Lemma 2 and 3

We can see that Lemma 2 is a special case of Lemma 3 with $\gamma = 0$, so we provide detailed proof for the latter only.

This result follows directly from [22, Lemma 3], which implies (in our notation)

$$\Delta_t \leq -\frac{1}{2}\left(\frac{\left(1-\sqrt{\gamma}\right)C_2}{\left(1+\sqrt{\gamma}\right)} + C_1\right)\left\|\boldsymbol{\Delta\alpha}^t\right\|^2 \tag{47}$$

and

$$\eta_t \geq \min\left(1, \frac{\beta(1-\tau)\sigma\left(\left(1+\sqrt{\gamma}\right)C_1 + \left(1-\sqrt{\gamma}\right)C_2\right)}{\|X^T X\|\left(1+\sqrt{\gamma}\right)}\right).$$

A.3 Proof of Theorem 1

Proof We first show the result for the variant of using backtracking line search. From (6), (47) with $\gamma = 0$, and (9), we have that

$$f(\boldsymbol{\alpha}^{t+1}) - f(\boldsymbol{\alpha}^t) \leq -\eta_t \tau \frac{C_1 + C_2}{2}\|\boldsymbol{\Delta\alpha}^t\|^2. \tag{48}$$

From the optimality of $\boldsymbol{\Delta\alpha}^t$ in (7), we get that

$$\nabla G^*(\boldsymbol{\alpha}^t) + B_t \boldsymbol{\Delta\alpha}^t + \tilde{\boldsymbol{s}}^{t+1} = 0, \tag{49}$$

for some $\tilde{\boldsymbol{s}}^{t+1} \in \partial \boldsymbol{\xi}^*(-\boldsymbol{\alpha}^t - \boldsymbol{\Delta\alpha}^t)$. By convexity, that the step size is in $[0,1]$, and the condition (21), we have

$$\begin{aligned} f\left(\boldsymbol{\alpha}^{t+1}\right) - f^* &\leq \eta_t \left(f\left(\boldsymbol{\alpha}^t + \boldsymbol{\Delta\alpha}^t\right) - f^*\right) + (1-\eta_t)\left(f\left(\boldsymbol{\alpha}^t\right) - f^*\right) \\ &\leq \eta_t \frac{\|\nabla G\left(\boldsymbol{\alpha}^t + \boldsymbol{\Delta\alpha}^t\right) + \tilde{\boldsymbol{s}}^{t+1}\|^2}{2\mu} + (1-\eta_t)\left(f\left(\boldsymbol{\alpha}^t\right) - f^*\right). \end{aligned} \tag{50}$$

Now to relate the first term to the decrease, we use (49) to get

$$\begin{aligned} \|\nabla G(\boldsymbol{\alpha}^t + \boldsymbol{\Delta\alpha}^t) + \tilde{\boldsymbol{s}}^{t+1}\|^2 &\leq \|\nabla G^*(\boldsymbol{\alpha}^t + \boldsymbol{\Delta\alpha}^t) - \nabla G^*(\boldsymbol{\alpha}^t) + \nabla G^*(\boldsymbol{\alpha}^t) + \tilde{\boldsymbol{s}}^{t+1}\|^2 \\ &\leq 2\|\nabla G^*(\boldsymbol{\alpha}^t + \boldsymbol{\Delta\alpha}^t) - \nabla G^*(\boldsymbol{\alpha}^t)\|^2 + 2\|B_t \boldsymbol{\Delta\alpha}^t\|^2 \\ &\leq 2\left(\frac{\|X^T X\|}{\sigma}\right)^2 \|\boldsymbol{\Delta\alpha}^t\|^2 + 2\|B_t\|^2\|\boldsymbol{\Delta\alpha}^t\|^2, \end{aligned} \tag{51}$$

where in the second inequality, we used $(a+b)^2 \leq 2(a^2+b^2)$ for all a, b, and in the last inequality we used the Lipschitz continuity of ∇G^*. We therefore get the following by combining (50), (51), and (48).

$$\begin{aligned} f\left(\boldsymbol{\alpha}^{t+1}\right) - f^* &\leq \frac{\eta_t}{\mu}\left(\left(\frac{\|X^T X\|}{\sigma}\right)^2 + C_3^2\right)\|\boldsymbol{\Delta\alpha}^t\|^2 + (1-\eta_t)\left(f\left(\boldsymbol{\alpha}^t\right) - f^*\right) \\ &\leq \left(\left(\frac{\|X^T X\|}{\sigma}\right)^2 + C_3^2\right)\frac{2\left(f\left(\boldsymbol{\alpha}^t\right) - f\left(\boldsymbol{\alpha}^{t+1}\right)\right)}{\mu(C_1+C_2)\tau} + (1-\eta_t)\left(f\left(\boldsymbol{\alpha}^t\right) - f^*\right). \end{aligned} \tag{52}$$

Let us define

$$C_4 := \left(\left(\frac{\|X^T X\|}{\sigma}\right)^2 + C_3^2\right)\frac{2}{\mu\left(C_1+C_2\right)\tau},$$

then rearranging (52) gives

$$\left(f(\boldsymbol{\alpha}^{t+1}) - f^*\right) \leq \frac{(1-\eta_t + C_4)}{1+C_4}\left(f(\boldsymbol{\alpha}^t) - f^*\right). \tag{53}$$

Combining the above result with the lower bound of η_t from (24) shows the desired Q-linear convergence rate. As of the exact line search variant, it produces an objective no larger than the left-hand side of (53), so the same rate holds.

A.4 Proof of Theorem 2

Proof This a direct application of [36, Theorem 1]. Their result implies

$$\frac{f\left(\boldsymbol{\alpha}^{t+1}\right) - f^*}{f\left(\boldsymbol{\alpha}^t\right) - f^*} \leq \begin{cases} 1 - \eta_t \tau (1-\gamma)\frac{\mu}{4C_3}, & \text{if } \mu \leq 2C_3, \\ 1 - \eta_t \tau (1-\gamma)\left(1 - \frac{C_3}{\mu}\right), & \text{else.} \end{cases} \tag{54}$$

Using (24) in (54), we obtain (25).

A.5 Proof of Theorem 3

Proof Our proof consists of using $\boldsymbol{\alpha}$ as the initial point, applying one step of some primal-dual algorithm, then utilizing the algorithm-specific relation between the decrease in one iteration and the duality gap to obtain the bound. Therefore we will obtain an algorithm-independent result from some algorithm-specific results.

When Assumption 2 holds, (1) is the type of problems considered in [38], and we have that $\boldsymbol{\xi}^*$ is $(1/\rho)$-strongly convex. If we take $\boldsymbol{\alpha}$ as the initial point, and apply one step of their method to obtain the next iterate $\boldsymbol{\alpha}^+$, from [38, Lemma 1], we get that for any $s \in [0,1]$,

$$\begin{aligned}\epsilon = f(\boldsymbol{\alpha}) - f(\boldsymbol{\alpha}^*) &\geq f(\boldsymbol{\alpha}) - f(\boldsymbol{\alpha}^+) \geq s\left(f^P(\boldsymbol{w}(\boldsymbol{\alpha})) + f(\boldsymbol{\alpha})\right) - \frac{s^2 G_s}{2\sigma}\\ &\geq s\left(f^P(\boldsymbol{w}(\boldsymbol{\alpha})) - f^P(\boldsymbol{w}^*)\right) - \frac{s^2 G_s}{2\sigma},\end{aligned} \tag{55}$$

where $\boldsymbol{w}^*$ is the optimal solution of (1), and

$$G_s := \left(\|X^T X\| - \frac{\sigma(1-s)}{s\rho}\right)\|\boldsymbol{u} - \boldsymbol{\alpha}\|^2, \quad -\boldsymbol{u}_i \in \partial\xi_i\left(X_i^T \boldsymbol{w}(\boldsymbol{\alpha})\right).$$

To remove the second term in (55), we set

$$\|X^T X\| - \frac{\sigma(1-s)}{s\rho} = 0 \quad \Rightarrow \quad s = \frac{\sigma}{\sigma + \rho\|X^T X\|} \in [0,1].$$

This then gives

$$\left(1 + \frac{\rho\|X^T X\|}{\sigma}\right)\epsilon \geq f^P(\boldsymbol{w}(\boldsymbol{\alpha})) - f^P(\boldsymbol{w}^*).$$

Although [38, Lemma 1] is for the expected value of the dual objective decrease at the current iteration and the expected duality gap at the previous iteration, we can remove the expectations as the expected duality gap is actually a constant for the initial point, and the expected function decrease cannot exceed the distance from the current objective to the optimum.

When Assumption 3 holds, (1) falls in the type of problems discussed in [1]. If we take $\boldsymbol{\alpha}$ as the initial point, and apply one step of their method to obtain the next iterate $\boldsymbol{\alpha}^+$, from the final inequality in the proof of Proposition 4.2 in [1] and weak duality, we get

$$\epsilon \geq s\left(f^P(\boldsymbol{w}(\boldsymbol{\alpha})) - f^P(\boldsymbol{w}^*)\right) - \frac{(sR)^2}{2\sigma}, \quad \forall s \in [0,1], \tag{56}$$

where

$$R^2 := \max_{\boldsymbol{\alpha},\boldsymbol{\beta}\in\Omega} \|X(\boldsymbol{\alpha} - \boldsymbol{\beta})\|^2 \leq \left\|X^T X\right\| \max_{\boldsymbol{\alpha},\boldsymbol{\beta}\in\Omega} \|\boldsymbol{\alpha} - \boldsymbol{\beta}\|^2 \leq 4\left\|X^T X\right\| L^2. \tag{57}$$

In the last equality we used [37, Corollary 13.3.3] such that if $\phi(\cdot)$ is L-Lipschitz continuous, then the radius of $\mathrm{dom}(\phi^*)$ is no larger than L. Now take $s = \min\{1, \sqrt{2\sigma\epsilon/R^2}\}$, we get that

$$\begin{cases} 2\epsilon \geq \epsilon + \frac{R^2}{2\sigma} \geq f^P(\boldsymbol{w}) - f^P(\boldsymbol{w}^*), & \text{if } \epsilon \geq \frac{R^2}{2\sigma},\\ \sqrt{\frac{2R^2\epsilon}{\sigma}} \geq f^P(\boldsymbol{w}) - f^P(\boldsymbol{w}^*), & \text{else.}\end{cases}$$

These conditions and (57) indicate that

$$f^P(\boldsymbol{w}) - f^P(\boldsymbol{w}^*) \leq \max\left\{2\epsilon, \sqrt{\frac{2\epsilon R^2}{\sigma}}\right\} \leq \max\left\{2\epsilon, \sqrt{\frac{8\epsilon\|X^T X\|L^2}{\sigma}}\right\}.$$

A.6 Proof of Lemma 4

Proof The part of (30) follows directly from the proof of [22, Corollary 1]. Notice that they required positive definiteness of the matrix for other parts stated in that corollary but the part for (30) holds true as long as B_t is positive semidefinite.

For the lower bound on the step size, we notice that for any $\eta \in [0,1]$,

$$\begin{aligned}
&f\left(\boldsymbol{\alpha}^t + \eta \Delta\boldsymbol{\alpha}^t\right) - f\left(\boldsymbol{\alpha}^t\right) \\
&= G^*\left(\boldsymbol{\alpha}^t + \eta \Delta\boldsymbol{\alpha}^t\right) - G^*\left(\boldsymbol{\alpha}\right) + \boldsymbol{\xi}^*\left(-\boldsymbol{\alpha}^t - \eta\Delta\boldsymbol{\alpha}^t\right) - \boldsymbol{\xi}^*\left(-\boldsymbol{\alpha}^t\right) \\
&\leq \eta \nabla G^*\left(\boldsymbol{\alpha}^t\right)^T \Delta\boldsymbol{\alpha}^t + \frac{\eta^2 L_B}{2}\left\|\Delta\boldsymbol{\alpha}^t\right\|_{B_t}^2 + \boldsymbol{\xi}^*\left(-\boldsymbol{\alpha}^t - \eta\Delta\boldsymbol{\alpha}^t\right) - \boldsymbol{\xi}^*\left(-\boldsymbol{\alpha}^t\right) && (58) \\
&\leq \eta \nabla G^*\left(\boldsymbol{\alpha}^t\right)^T \Delta\boldsymbol{\alpha}^t + \eta\left(\boldsymbol{\xi}^*\left(-\boldsymbol{\alpha}^t - \Delta\boldsymbol{\alpha}^t\right) - \boldsymbol{\xi}^*\left(-\boldsymbol{\alpha}^t\right)\right) + \frac{\eta^2 L_B}{2}\left\|\Delta\boldsymbol{\alpha}^t\right\|_{B_t}^2 && (59) \\
&= \eta\Delta_t + \frac{L_B\eta^2}{2}\left\|\Delta\boldsymbol{\alpha}^t\right\|_{B_t}^2 \leq \eta\Delta_t - \frac{L_B\eta^2\left(1+\sqrt{\gamma}\right)}{2}\Delta_t, && (60)
\end{aligned}$$

where we used (26) in (58), the convexity of $\boldsymbol{\xi}^*(-\cdot)$ in (59), and (30) in (60). We can therefore see that (9) is satisfied when

$$\left(\eta - \frac{L_B\eta^2\left(1+\sqrt{\gamma}\right)}{2}\right)\Delta_t \leq \eta\tau\Delta_t.$$

As $\Delta_t \leq 0$ from (30), we have that (9) holds whenever

$$\eta \leq \frac{2\left(1-\tau\right)}{L_B\left(1+\sqrt{\gamma}\right)}.$$

After considering the overshoot of backtracking by a factor of β, this inequality leads to the desired step size bound.

A.7 Proof of Theorem 4

Proof We define $(\Delta\boldsymbol{\alpha}^t)^*$ as the optimal solution for (7) at the t-th iteration and start from [22, Lemma 5], which states that when f is convex, we have the following inequality.

$$Q_{B_t}^{\boldsymbol{\alpha}^t}\left(\left(\Delta\boldsymbol{\alpha}^t\right)^*\right) \leq -\lambda\left(f\left(\boldsymbol{\alpha}^t\right) - f^*\right) + \frac{\lambda^2}{2}\min_{\boldsymbol{\alpha}^*\in A}\left\|\boldsymbol{\alpha}^t - \boldsymbol{\alpha}^*\right\|_{B_t}^2, \forall\lambda\in[0,1].$$

By utilizing (27), we further deduce that

$$Q_{B_t}^{\boldsymbol{\alpha}^t}\left(\left(\Delta\boldsymbol{\alpha}^t\right)^*\right) \leq -\lambda\left(f\left(\boldsymbol{\alpha}^t\right) - f^*\right) + \frac{\lambda^2}{\mu_B}\left(f\left(\boldsymbol{\alpha}^t\right) - f^*\right), \forall\lambda\in[0,1], \tag{61}$$

which attains the minimal value at $\lambda = \mu_B/2$, which is in $[0,1]$ according to our assumption that $\mu_B \leq 2$. By considering the line search stopping condition (9) and using $\lambda = \mu_B/2$ in (61), we can see that since B_t is positive semidefinite,

$$\begin{aligned}
f\left(\boldsymbol{\alpha}^t + \eta_t\Delta\boldsymbol{\alpha}^t\right) - f\left(\boldsymbol{\alpha}^t\right) &\leq \eta_t\tau\Delta_t \leq \eta_t\tau\left(\Delta_t + \frac{1}{2}\left\|\Delta\boldsymbol{\alpha}^t\right\|_{B_t}^2\right) = \eta_t\tau Q_{B_t}^{\boldsymbol{\alpha}^t}\left(\Delta\boldsymbol{\alpha}^t\right) \\
&\leq \eta_t\tau\left(1-\gamma\right)Q_{B_t}^{\boldsymbol{\alpha}^t}\left(\left(\Delta\boldsymbol{\alpha}^t\right)^*\right) \leq -\eta_t\tau\left(1-\gamma\right)\frac{\mu_B}{4}\left(f\left(\boldsymbol{\alpha}^t\right) - f^*\right).
\end{aligned}$$

Finally, by taking in the step size in Lemma 4, we see that the convergence speed is

$$\begin{aligned}
f\left(\boldsymbol{\alpha}^{t+1}\right) - f^* &= f\left(\boldsymbol{\alpha}^t + \eta_t\Delta\boldsymbol{\alpha}^t\right) - f^* \\
&\leq \left(1 - \frac{\tau\left(1-\gamma\right)\mu_B}{4}\min\left\{1, \frac{2\beta\left(1-\tau\right)}{\left(1+\sqrt{\gamma}\right)L_B}\right\}\right)\left(f\left(\boldsymbol{\alpha}^t\right) - f^*\right) \\
&= \left(1 - \frac{\tau\mu_B}{2}\min\left\{\frac{1-\gamma}{2}, \frac{\beta\left(1-\tau\right)\left(1-\sqrt{\gamma}\right)}{L_B}\right\}\right)\left(f\left(\boldsymbol{\alpha}^t\right) - f^*\right).
\end{aligned}$$