Abstract

Designing distributed algorithms for empirical risk minimization (ERM) has become an active research topic in recent years because of the practical need to deal with the huge volume of data. In this paper, we propose a general framework for training an ERM model via solving its dual problem in parallel over multiple machines. Our method provides a versatile approach for many large-scale machine learning problems, including linear binary/multi-class classification, regression, and structured prediction. Comparing with existing approaches, we show that our method has faster convergence under weaker conditions both theoretically and empirically.

1. Introduction

With the rapid growth of data volume and model complexity, designing efficient learning algorithms has become increasingly important. Distributed optimization techniques, which decompose a large optimization problem into sub-problems and distribute the computational burden across multiple machines, have gained a great amount of interest. This type of approaches are particularly useful when the optimization problem involves massive computation and/or the data set is stored on multiple machines as it could not fit the capacity of a single node. However, the communication cost and the asynchronous nature of the process challenge the design of efficient optimization algorithms in distributed environments.

In this paper, we study distributed optimization algorithms for training machine learning models that can be represented under the regularized empirical risk minimization (ERM) framework. These models include binary/multi-class classification, regression, and structured prediction models, covering a variety of learning tasks. We specifically focus on linear models, which have been shown successful in dealing with large-scale data thanks to their efficiency and interpretability.\(^1\)

Given a set of training data, \(\{X_i\}_{i=1}^l, X_i \in \mathbb{R}^{n \times c_i}, c_i \in \mathbb{N}\), regularized ERM models solve the following optimization problem:

\[
\min_{w \in \mathbb{R}^n} \quad f^P(w) := g(w) + \sum_{i=1}^l \xi_i (X_i^T w),
\]

where \(g\) and \(\xi_i\) are the regularization term and the loss term, respectively. We assume that \(f^P\) is a proper, closed convex function.

\(^1\) Linear models allow us to interpret the value of each feature from the model.
Note that the definition of ERM problems in problem (1) is general and covers a variety types of learning problems. To unify the definitions of different learning problems, we encoded the true labels (i.e., $y_i \in Y_i$) in the loss term $\xi_i$ 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 several words, $c_i$ represents the number of words in the $i$-th sentence. We provide a detailed discussion about the loss terms for different types of learning problems in Section 6.

Regarding the regularization term, common choices of $g$ include the squared $\ell_2$ norm, the $\ell_1$ norm, and the elastic net (Zou and Hastie, 2005) that combines the both.

In many applications, it is preferable to solve the Lagrange dual problem of problem (1), because the dual problem has better mathematical properties, making optimization easier. The Lagrange dual problem of (1) is

$$
\min_{\alpha \in \Omega} \ f(\alpha) := g^*(X\alpha) + \sum_{i=1}^l \xi_i^*(-\alpha_i), 
$$

where

$$
X := [X_1, \ldots, X_l], \quad \alpha := \begin{bmatrix} \alpha_1 \\ \vdots \\ \alpha_l \end{bmatrix},
$$

$\alpha_i \in \mathbb{R}^{c_i}$ is the dual variable associated with $X_i$, and for any function $h(\cdot)$, $h^*(\cdot)$ is its convex conjugate:

$$
h^*(w) := \max_{z \in \text{dom}(h)} z^T w - h(w), \quad \forall w.
$$

The domain of $\alpha$, $\Omega$, is defined to be $\prod_{i=1}^l \text{dom}(\xi_i^*) \subseteq \mathbb{R}^{\sum_{i=1}^l c_i}$. We use the following notations to simplify our description.

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

In a single-core setting, optimization methods for the dual ERM (2) have been widely studied (see, for example, Yuan et al. (2012) for a review). However, most of the state-of-the-art algorithms for the dual ERM are inherently sequential and cannot be parallelized easily. On the other hand, in a distributed setting, and the cost of communication is relatively high that it is usually the bottleneck for distributed optimization. Therefore, it is important to reduce the communication cost to make the optimization efficient. Consequently, algorithms with faster convergence rates are desirable for distributed optimization, because fewer iterations implies fewer communication rounds and thus lower communication cost. We observe that without careful consideration of this issue, existing distributed dual solvers do not achieve satisfactory training speed. As a result, existing distributed algorithms like Chen et al. (2014); Zhuang et al. (2015); Lin et al. (2014); Lee et al. (2017) often focus on using the gradient and/or the Hessian information to solve the primal form of ERM (1), whose computation either can be parallelized naturally or require only some modification in the implementation to adapt for distributed environments. However, in this paper, we show that with a careful design, dual approaches can be competitive with primal approaches and enjoy sound theoretical properties.

Note that some methods, such as Zhang and Lin (2015), reduce the number of communication rounds significantly; however, their computational cost per iteration is excessively high. These methods, though theoretically appealing in terms of communication efficiency, are impractical as their total running time is high because of the computational cost. In contrast, our focus in this work is a practical algorithm that works well for practitioners such that both computation and communication concerns are properly dealt with so that the overall running time is shorter.
In this work, we propose an efficient distributed framework for solving (2). At each iteration, our framework considers minimizing a problem consisting of a positive semidefinite block-diagonal second-order approximations of the non-separable \( g^*(X\alpha) \) term in (2), plus the original separable loss conjugate \( \xi^*(-\alpha) \). We consider the setting that 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 \( \bigcup_{k=1}^{K} J_k = \{1, \ldots, l\} \), \( J_i \cap J_k = \emptyset \), \( \forall i \neq k \).

Without loss of generality, we assume that \( j_0 := 0 \), \( j_K := l \), \( J_k := \{j_{k-1} + 1, \ldots, j_k\} \), \( k = 1, \ldots, K \). For any vector \( v \in \mathbb{R}^l \), \( v_{J_k} \) denotes the sub-vector in \( \mathbb{R}^{\left| J_k \right|} \) that contains the coordinates of \( v \) indexed by \( J_k \).

Our framework iteratively solves a second-order approximation problem of (2) in a distributed environment. We discuss how to choose the approximation to make our method efficient in terms of convergence and communication. After solving the approximated problem, we conduct a line search that only requires negligible computational cost and \( O(1) \) communication. By utilizing a relaxed condition, our method is able to achieve global linear convergence for many non-strongly convex dual problems, including the popular support vectors machines (SVM) model proposed by Boser et al. (1992); Vapnik (1995), and the structured SVM (SSVM) problem (Tsochantaridis et al., 2005; Taskar et al., 2004). In other words, our algorithm only requires \( O(\log(1/\epsilon)) \) iterations, or equivalently, rounds of communication, to obtain an \( \epsilon \)-accurate solution for (2). We also discuss the main differences between our approach and existing distributed algorithms for (2). Experiments show that our algorithm is significantly faster than existing distributed solvers for (2).

Special cases of our framework were published earlier as conference and workshop papers (Lee and Roth, 2015; Lee et al., 2015). In this work we unify the results and extend the previous work to a general setting. We also provide more thorough theoretical and empirical analysis. We note that recently, in the work Zheng et al. (2017) an accelerated method for solving the dual ERM problem distributedly similar to the catalyst for convex optimization (Lin et al., 2015) is proposed. Like the catalyst framework that can be combined with any convex optimization algorithm, this acceleration in Zheng et al. (2017) can also be applied on top of our framework easily. Therefore, we focus our comparison on methods that directly solve the original optimization problem, since methods that are faster for the original problem are expected to perform better as well when combined with the acceleration framework.

This paper is organized as follows. We describe the algorithm overview in Section 2. The implementation details for distributed environments are shown in Section 3. In Section 4, we analyze the convergence property of our algorithm. Section 5 discusses related works for distributed ERM optimization. Details of applying our algorithm on specific ERM problems are described in Section 6. Numerical experiments are provided in Section 7. Some possible extensions and limitations of our work are discussed in Section 8. We then make some concluding remarks in Section 9. The program used in this work is available at \texttt{http://github.com/leepei/blockERM}.

2. Algorithm

Throughout this work, we make the following assumption.

**Assumption 2.1** There is \( \sigma > 0 \) such that the regularizer \( g \) in the primal problem (1) is \( \sigma \)-strongly convex. Namely,

\[
g(\alpha w_1 + (1-\alpha)w_2) \leq \alpha g(w_1) + (1-\alpha)g(w_2) - \frac{\sigma\alpha(1-\alpha)}{2} \|w_1 - w_2\|^2, \quad \forall w_1, w_2 \in \mathbb{R}^n, \quad \forall \alpha \in [0, 1].
\]
Note that the goal of solving the dual problem is to obtain a solution to the original primal problem \((1)\). It can easily be shown that strong duality between \((1)\) and \((2)\) holds, which means that any pair of primal and dual optimal solutions \((\mathbf{w}^*, \alpha^*)\) satisfies the following relation.

\[
\mathcal{f}^P(\mathbf{w}^*) = -\mathcal{f}(\alpha^*).
\]

By (Hiriart-Urruty and Lemaréchal, 2001, Part E, Theorem 4.2.2), since \(g\) is \(\sigma\)-strongly convex, \(g^*\) is differentiable and has \((1/\sigma)\)-Lipschitz continuous gradient everywhere. This also indicates that even if \(g\) is extended-valued, \(g^*\) is still finite everywhere, hence the feasible region of \(\alpha\) is only restricted to \(\Omega\). From the KKT optimality conditions, we further have

\[
\mathbf{w}^* = \nabla g^*(X\alpha^*). \tag{5}
\]

Although \((5)\) only holds at the optimum, we can still define \(\mathbf{w}(\alpha)\) as the primal solution associated with \(\alpha\), when \(\alpha\) is a sub-optimal solution to \((2)\),

\[
\mathbf{w}(\alpha) := \nabla g^*(X\alpha). \tag{6}
\]

Our algorithm is an iterative descent method for solving \((2)\). Start with any arbitrary feasible \(\alpha^0\), it generates a sequence of iterates \(\{\alpha^1, \alpha^2, \ldots\}\) with a property that \(\mathcal{f}(\alpha^i) \leq \mathcal{f}(\alpha^j)\) if \(i > j\). The iterates are updated by an update direction \(\Delta \alpha^t\) and a step size \(\eta_t \geq 0\), where \(\eta_t\) will be specified later.

\[
\alpha^{t+1} = \alpha^t + \eta_t \Delta \alpha^t, \quad \forall t \geq 0. \tag{7}
\]

In the objective function in \((2)\), each element in \(\xi^t\) usually can be computed separately. Therefore, it can be optimized directly in a coordinate-wise manner. However, \(g^*\) is complex and hard to optimize. Thus, we use its approximation as a surrogate such that the resulting sub-problems are easier to solve. As we mentioned above, \(g^*\) and thus \(G^*\) are differentiable and the gradients are Lipschitz continuous. Given the current iterate \(\alpha^t\), we solve

\[
\Delta \alpha^t := \arg\min_{\Delta \alpha} Q_{B_t}^\alpha(\Delta \alpha) := \nabla G^\alpha(\alpha^t)^T \Delta \alpha + \frac{1}{2}(\Delta \alpha)^T B_t \Delta \alpha + \xi^t(-\alpha^t - \Delta \alpha) \tag{8}
\]

for some symmetric \(B_t\), and then take \(\Delta \alpha^t\) as the update direction. The matrix \(B_t\) can be varied over iterations and has a wide range of choices, depending on the goal. One usually wants to pick \(B_t\) such that \((8)\) is easy to solve to reduce the cost per iteration, or that \(Q_{B_t}^\alpha(\Delta \alpha^t)\) is a tighter approximation of \(\mathcal{f}(\alpha^t + \Delta \alpha^t) - \mathcal{f}(\alpha^t)\) so that it takes fewer iterations to obtain a good enough solution. We will show in Section 4 that as long as the whole objective of \((8)\) is strongly convex, the obtained \(\Delta \alpha^t\) will be a descent direction. We then consider two different line search possibilities to obtain the step size \(\eta_t\). The first is the exact line search.

\[
\eta_t = \arg\min_{\eta \in \mathbb{R}} \mathcal{f}(\alpha^t + \eta \Delta \alpha^t). 
\]

However, this approach can be conducted only when the objective function has some specific structure to allow so. In the general case, we consider a backtracking line search with the Armijo rule. Given \(\gamma \geq 1, \beta \in (0,1), \tau \in (0,1)\), this procedure finds the smallest \(i \geq 0\) such that \(\eta = \gamma \beta^i\) satisfies

\[
f(\alpha^t + \eta \Delta \alpha^t) \leq f(\alpha^t) + \eta \tau \Delta_t,
\]

where

\[
\Delta_t := \nabla G^*(\alpha^t)^T \Delta \alpha^t + \xi^*(-\alpha^t - \Delta \alpha^t) - \xi^*(-\alpha^t). \tag{10}
\]
3. Distributed Implementation for Dual ERM

We now discuss how to adapt the algorithm framework in Section 2 to make distributed optimization for dual ERM efficient. In particular, we will discuss the choice of $B$ in (8), the trick of making line search efficient, and how to solve (8) with the minimum amount of communication.

For the ease of algorithm description, we will use the following representation of $X$ and $\alpha$.

$$X = [x_1, \ldots, x_N], \quad \alpha = \begin{bmatrix} \alpha_1 \\ \vdots \\ \alpha_N \end{bmatrix},$$

(11)

where

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

is the number of columns of $X$ as well as the dimension of $\alpha$. The index sets for the representation (11) corresponding to $J_k$ is denoted by $\tilde{J}_i \subseteq \{1, \ldots, N\}, i = 1, \ldots, K$. We define $\pi(i) = k$ if $i \in J_k$.

3.1 Update Direction

In this section, we discuss how to select $B_t$ such that the objective of (8) is strongly convex, and the update direction can be uniquely decided. Notice that we distribute the data into multiple machines, and the $k$-th machine only stores and handles $J_k$ and corresponding dual variables $\alpha_i$ associated with $X_i$. Therefore, in order to reduce the communication cost, we need to pick $B_t$ in a way that (8) can be decomposed into independent sub-problems, and each sub-problem only involves a subset of data. In such way, each sub-problem can be solved locally on one machine without communicating with other machines. According to the partition (3), we should consider a block-diagonal $B_t$ such that

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

(12)

Any $B_t$ satisfying (12) falls in the block-diagonal approximation framework of our method. In most cases, we would like to incorporate higher order information as much as possible to obtain fast convergence, which will then reduces the rounds of expensive communication. A natural choice would then be the Hessian matrix.

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

(13)

However, the Hessian matrix is usually dense and does not satisfy (12). Therefore, we consider its block-diagonal approximation $\tilde{H}_{\alpha^t}$.

$$
\begin{cases} 
(H_{\alpha^t})_{i,j} & \text{if } \pi(i) = \pi(j), \\
0 & \text{otherwise.}
\end{cases}
$$

(14)

When the function $g^*$ is complicated such that $\nabla^2 g^*$ cannot be calculated easily, one can use the identity matrix, or some diagonal approximation, to substitute it in (13). Note that since

$$(\nabla^2 G^*(X\alpha^t))_{i,j} = x_i^T \nabla^2 g^*(X\alpha^t) x_j,$$

if each machine maintains the information of $X\alpha^t$, entries of (14) can be decomposed into parts that each only require information from data in one partition of $\tilde{J}_k$. Thus the sub-problems can indeed be solved independently on corresponding machines without communication.

Another concern of using the Hessian matrix is that it might be only positive semidefinite, so that the requirement of the objective of (8) is not satisfied when $\xi^*$ is not strongly convex. In this case, we can add a damping term to $B_t$ to ensure the strong convexity.
Therefore, our choice for $B_t$ can be generalized to the following formulation.

$$B_t = a_1^t H_{\alpha_t} + a_2^t I,$$

for some $a_1^t, a_2^t \geq 0$. (15)

The choices of $a_1^t$ and $a_2^t$ might depend on problem structures and the applications. In most cases, $a_2^t = 0$ should be considered, especially when it is known that $Q_{H_{\alpha_t}}$ is already strongly convex. For $a_1^t$, practical results (Pechyony et al., 2011; Yang, 2013) suggest that $a_1^t \in [1, K]$ leads to fast convergence, while we prefer $a_1^t \equiv 1$ as it is a tighter approximation of the Hessian matrix. On the other hand, for the special case of $a_1^t = 0$, our framework reduces to proximal gradient, and when $K = 1$ and $a_1^t = 1, a_2^t = 0$, it turns into the proximal Newton method.

In solving (8) using our choice of (15), we notice that each machine needs the information of $X \alpha_t$ to calculate both $B_t J_k, J_k$ and $\nabla G^* (X \alpha_t) J_k = X^T J_k \nabla g^* (X \alpha_t)$. (16)

Therefore, after each iteration of updating $\alpha_t$, we need to synchronize the information $v_t := X \alpha_t = \sum_{k=1}^{K} \sum_{j \in J_k} x_j \alpha_j^t$ through communication. This communication of an $O(n)$ vector is more suitable than communicating either the Hessian or the whole $X$ together with $\alpha_t$. However, because we need to use the update direction to conduct line search, the better approach is to synchronize the following vector instead.

$$\Delta v_t := X \Delta \alpha_t = \sum_{k=1}^{K} \sum_{j \in J_k} x_j \Delta \alpha_j^t,$$

and then update

$$v_{t+1} = v_t + \eta_t \Delta v_t$$

locally after the step size $\eta_t$ is determined. More explanations of communicating this vector will be explained in the next section when we discuss the details for conducting line search.

### 3.2 How to Conduct Line Search Efficiently

After the update direction $\Delta \alpha$ is decided from solving (8), we need to conduct line search to ensure sufficient function decrease. For the backtracking variant, In the right-hand side of (9), the first term is already available from the previous iteration, and we thus only need to calculate (10). From (16), this value can be calculated by

$$\Delta_t = \nabla g^* (v^t) \Delta v^t + (\xi^* (\alpha^t - \Delta \alpha^t) - \xi^* (-\alpha^t)) .$$

(19)

Notice that to obtain the terms related to $\xi^*$, we need an $O(1)$ communication. Because each machine has full information of $v^t$ and hence $g' (v^t)$, the calculation of the first term can also be conducted distributedly, and we can combine the partial sums of all terms in (19) in one scalar to compute to avoid redundant communication. One can also see from this calculation that synchronizing $\Delta v^t$ is inevitable to compute the required value efficiently.

For the left-hand side of (9), in this calculation we use the original formulation in (2) because this simplifies the computation. The $\xi_i^*$ term is calculated distributedly in nature, while for the $g^* (v)$ term, if this term is separable with respect to $v$, then the computation is also parallelizable. Further, we may even have a closed form solution to calculate it in $O(1)$ time for different $\eta$ if some values are precomputed in some special cases. For example, when

$$g^* (v) = \frac{1}{2} \|v\|^2 ,$$
Algorithm 1: Backtracking line search

**Input:** $\gamma \geq 1, \beta \in (0, 1), \tau \in (0, 1), \alpha, \Delta \alpha \in \mathbb{R}^l, f(\alpha) \in \mathbb{R}, v = X\alpha, \Delta v = X\Delta \alpha$

Distributedly calculate $\Delta_t$: $O(1)$ communication

\[
\Delta_t = \sum_{k=1}^{K} \left( \nabla g^* (v^t)^T J_k + \sum_{j \in J_k} \xi^*_j (-\alpha_j + \Delta \alpha_j) - \xi^*_j (-\alpha_j) \right).
\]

Let $\eta \leftarrow \gamma$

Calculate $f(\alpha + \eta \Delta \alpha)$ using $v$ and $\eta \Delta v$ $O(1)$ communication

while $f(\alpha + \eta \Delta \alpha) > f(\alpha) + \eta \tau \Delta_t$ do

$\eta \leftarrow \eta \beta$

Calculate $f(\alpha + \eta \Delta \alpha)$ using $v$ and $\eta \Delta v$ $O(1)$ communication

end

Output: $\eta, f(\alpha + \eta \Delta \alpha)$

we have that

\[
g^*(v + \eta \Delta v) = \frac{1}{2} \left( \|v\|^2 + \eta^2 \|\Delta v\|^2 + 2\eta v^T \Delta v \right). \tag{20}
\]

In this case we can compute $\|\Delta v\|^2$ and $v^T \Delta v$ distributedly in advance, then the calculation of (20) with different $\eta$ requires only $O(1)$ computation and does not need any communication. For the general case, though the computation might not be this low, by maintaining both the current $v$ and $\Delta v$, the computation of $g(v + \eta \Delta v)$ requires no more communication but just at most $O(\eta)$ computation locally, and this cost is negligible as other parts of the algorithm already involves computational cost more expensive than $O(\eta)$. Another advantage of using $\eta \Delta v$ is that when the backtracking procedure is terminated by satisfying (9), the value of $v^{t+1}$ can be easily obtained and we do not need to conduct (18) again. The line search procedure is summarized in Algorithm 1.

For the exact line search variant, it is possible only when

\[
\frac{\partial f(\alpha + \eta \Delta \alpha)}{\partial \eta} = 0
\]

has an analytic solution, which requires both $g^*$ and $\xi^*$ to be differentiable at least in some open set. For example, when the objective of $f$ is quadratic, the optimal step size can be obtained by

\[
\frac{\partial f(\alpha + \eta \Delta \alpha)}{\partial \eta} = 0 \Rightarrow \eta = \frac{-\nabla f(\alpha)^T \Delta \alpha}{\Delta \alpha^T \nabla^2 f(\alpha) \Delta \alpha}. \tag{21}
\]

Another way to approximate the exact line search variant is to consider a bisection method if the above assumption for analytic solution does not hold. In this case, we can utilize the trick of maintaining both $v$ and $\Delta v$ mentioned above to reduce the communication cost of re-evaluating the objective value.

3.3 Local Solver at Each Machine

When we pick $B_t$ satisfying (12), the problem (8) can be decomposed into $K$ independent subproblems of the following form.

\[
\min_{\Delta \alpha_{J_k} \in \mathbb{R}^{|J_k|}} \nabla G^*(\alpha^t)^T J_k \Delta \alpha_{J_k} + \frac{1}{2} \Delta \alpha_{J_k}^T (B_t)_{J_k, J_k} \Delta \alpha_{J_k} + \sum_{i \in J_k} \xi^*(-\alpha^t_i - \Delta \alpha_i). \tag{22}
\]
Note that since all the information needed in (22) is available on machine $k$, the sub-problems can be solved without any inter-machine communication.

Our framework does not limit the solver for (8). Instead, one can take arbitrary local solver that is suitable for a specific problem. For example, we can consider (block) coordinate descent, projected gradient, proximal methods, and so on. Because each sub-problem is very close to the original dual problem of regularized ERM shown in (2), the cost is also similar, and one can consider methods that are efficient in the single-machine setting. In our experiment, we will use random-permuted cyclic coordinate descent (Hsieh et al., 2008; Yu et al., 2011; Chang and Yih, 2013) as our local solver, as this approach has been proven empirically to be efficient in the optimizing dual ERM problems in the single-core setting. Note that although the theoretical convergence rate of cyclic coordinate descent can be as worse as $O(l^2)$ times slower than its randomized, non-cyclic counter part (Sun and Ye, 2016), its random-permuted version is known to behave similarly to the randomized and non-cyclic version, while it still has the deterministic convergence guarantee, and thus is the most favorable version in practice. Other options can be adopted in discretion for specific problems or data sets, 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 is guaranteed that $f(\alpha^{t_1}) < f(\alpha^{t_2})$ provided $t_1 > t_2$. However, when we obtain the primal iterates $w^t$ from (6), there is no guarantee that the primal objective is strictly decreasing. Although we are able to prove that the primal objective converges R-linearly in Section 4, we still cannot guarantee that the latest primal iterate is the one with the lowest primal objective. This situation happens for all methods that solve the dual problem. To deal with this problem, we keep track of the primal objective of each iterate, and when the algorithm is terminated, we take the iterate with the lowest primal objective as the final model. This is known as the pocket approach in the literature of the perceptron algorithm (Gallant, 1990).

3.5 Stopping criterion

If we consider only the sub-optimality of $f(\alpha^t)$, then we can use some simple indicators of sub-optimality such as the norm of the update direction, or the norm of the projected gradient, whose values are zero at the optima, as the practical stopping condition. However, since we already computed the primal objective at each iteration as discussed above, we can directly use this as the stopping criterion with no additional cost, and this criterion should be more useful as the indicator of the model quality than others. Therefore, we consider the following stopping criterion:

$$f(\alpha^t) + f^P(w(\alpha^t)) \leq \epsilon (f(\alpha^0) + f^P(w(\alpha^0))),$$

with some user-specified $\epsilon \geq 0$. The overall procedure for distributedly optimizing (2) discussed in this section is described in Algorithm 2.

3.6 Cost per Iteration

For the ease of description and analysis, we assume that the data entries are distributed evenly across machines. Namely, we assume that each machine has $O(\#nnz/K)$ non-zero entries, where where $\#nnz$ is the number of non-zero elements in $X$, and the number of data points on each machine is $O(l/K)$. We will separate the cost for computation and for communication in our discussion. In our cost analysis below, we do not make those assumptions of low computational cost for general $f$ and $g^*$ resulted from the specific structures we discussed above. Instead, we only assume that the computation of $g^*(v)$ and $\nabla g^*(v)$ are non-parallelizable and cost $O(n)$, as further acceleration is a case-by-case possibility provided the special problem structure. The part of $\nabla^2 g^*(v)$ is assumed
to be negligible, for otherwise we will simply replace it with a multiple of the identity matrix as discussed in Section 3.1.

We first check the cost of forming the problem (8). The calculation of $\Delta v^t$ through (17) costs $O(\#\text{nnz}/K)$ computation time and $O(n)$ communication time. Calculating $\nabla g^*(v^t)$ using the known $v^t$ costs $O(n)$, and therefore computing $\nabla G(\alpha^t)^T_{\alpha} \Delta J_k$ costs $O((n + N)/K)$ computationally, as we never form $\nabla G(\alpha^t)$ explicitly, but only compute $\nabla G(\alpha^t)^T \Delta \alpha^t$ through $\nabla g^*(v^t) \Delta v^t$.

Next, the cost of solving (8) is $O(\#\text{nnz}/K)$, as noted in most state-of-the-art single-core optimization methods for dual ERM. For the line search part, calculating $\Delta \alpha$ has computational cost $O((n + N)/K)$ and $O(1)$ communication cost. Each line search iteration costs $O(n + N/K)$ computation and $O(1)$ communication. Finally, evaluating the primal objective costs $O(n)$ for $g(w(\alpha^t))$ and $O(\#\text{nnz}/K)$ for $X^T w(\alpha^t)$ in computation. It also takes $O(1)$ communication to gather the summation over $\xi_i$.

In summary, each iteration of Algorithm 2 costs

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

in computation and

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

in communication. We will show in the next section that the number of line search is upper-bounded by a constant, so the overall cost per iteration is $O((\#\text{nnz} + N)/K + n)$ computationally and $O(n)$ for communication.

4. Analysis

We now show that the update direction is indeed a descent one, and the backtracking line search procedure terminates within a bounded number of steps.
Lemma 4.1  Given the current iterate $\alpha^t$, if $B_t$ is chosen so that $Q_{B_t}^\alpha$ is $c_2$-strongly convex for some $c_2 > 0$ and $c_1 + c_2 > 0$, where $c_1$ is the smallest eigenvalue of $B_t$, then the solution $\Delta \alpha_t$ obtained by solving (8) is a descent direction for $f$ at $\alpha^t$, namely $\Delta_t < 0$. In particular,

$$
\Delta_t \leq -\frac{c_1 + c_2}{2} \| \Delta \alpha_t \|^2 \leq 0.
$$

(23)

Moreover, the backtracking line search procedure in Algorithm 1 terminates to satisfy (9) within $\max(0, \lceil \log_\alpha (1 - \tau) \sigma(c_1 + c_2)/(\|X^T X\|) \rceil)$ steps, and the step size is lower bounded by

$$
\eta_t \geq \min \left( 1, \frac{\beta(1 - \tau) \sigma(c_1 + c_2)}{\|X^T X\|} \right).
$$

(24)

**Proof**  For any $\lambda \in [0, 1)$, we have

$$
Q_{B_t}^{\alpha^t}(\Delta \alpha^t) \leq Q_{B_t}^{\alpha^t}(\lambda \Delta \alpha^t)
$$

$$
\leq \lambda Q_{B_t}^{\alpha^t}(\Delta \alpha^t) + (1 - \lambda) Q_{B_t}^{\alpha^t}(0) - \frac{c_2}(1 - \lambda) \| \Delta \alpha^t \|^2
$$

$$
= \lambda \left( \nabla G^*(\alpha^t)^T \Delta \alpha^t + \frac{1}{2} \| \Delta \alpha^t \|^2 \right)
$$

$$
= \frac{\lambda}{2} \left( \nabla G^*(\alpha^t)^T B_t \Delta \alpha^t + \frac{1}{2} \| \Delta \alpha^t \|^2 \right)
$$

where in the first inequality we used that $\Delta \alpha_t$ is the minimizer of (8), and in the second one we used the strong convexity of $Q_{B_t}^{\alpha^t}$. Rearranging this inequality gives us

$$
(1 - \lambda) \left( G^*(\alpha^t)^T \Delta \alpha^t + \xi^*(-\alpha^t - \Delta \alpha^t) - \xi^*(-\alpha^t) \right)
$$

$$
\leq \frac{(1 - \lambda)}{2} \| \Delta \alpha^t \|^2 - \frac{c_2}{2} \| \Delta \alpha^t \|^2.
$$

Divide both sides by $(1 - \lambda)$, we get

$$
G^*(\alpha^t)^T \Delta \alpha^t + \xi^*(-\alpha^t - \Delta \alpha^t) - \xi^*(-\alpha^t)
$$

$$
\leq -\frac{1}{2} \| \Delta \alpha^t \|^2 B_t \Delta \alpha^t - \frac{c_2}{2} \| \Delta \alpha^t \|^2 \leq -\frac{c_1 - c_2}{2} \| \Delta \alpha^t \|^2.
$$

Let $\lambda \to 1^-$, we obtain (23). Note in the last inequality of (23), equality holds if and only if $\Delta \alpha_t = 0$, in which case $G^*(\alpha^t)^T d + \xi^*(-\alpha^t - d) - \xi^*(-\alpha^t) > 0$ for all $d \neq 0$, meaning that $\alpha^t$ is the optimum of (2). Therefore (23) indicates that $\Delta \alpha_t$ is a descent direction when the current iterate is not optimal because from Taylor expansion of the smooth term we have

$$
f(\alpha^t + \eta \Delta \alpha^t) - f(\alpha^t) = \eta \nabla G^*(\alpha^t)^T \Delta \alpha^t + O(\eta^2) + \xi^*(-(\alpha^t + \eta \Delta \alpha^t)) - \xi^*(-\alpha^t)
$$

$$
\leq \eta \nabla G^*(\alpha^t)^T \Delta \alpha^t + O(\eta^2) + \eta \xi^*(-(\alpha^t + \Delta \alpha^t)) - \xi^*(-\alpha^t)
$$

$$
= \eta \Delta_t + O(\eta^2),
$$

which implies that when $\eta$ is small enough and positive, $f(\alpha^t + \eta \Delta \alpha^t)$ will be smaller than $f(\alpha^t)$.

Regarding the backtracking termination, from the $(1/\sigma)$-Lipschitz continuity of $\nabla g$ and the convexity, we have that for any $\eta \in [0, 1]$ and any vector $d$,

$$
f(\alpha^t + \eta d) - f(\alpha^t)
$$

$$
= G^*(\alpha^t + \eta d) - G^*(\alpha^t) + \xi^*(-\alpha^t - \eta d) - \xi^*(-\alpha^t)
$$

$$
\leq \eta \left( \nabla G^*(\alpha^t)^T d + \xi^*(-\alpha^t - d) - \xi^*(-\alpha^t) \right) + \frac{\eta^2\|X^t d\|^2}{2\sigma}.
$$

(25)
Take \( d = \Delta \alpha^t \) in (25), we see that it suffices to have \( \eta \in [0, 1] \) such that

\[
\eta \Delta_t + \frac{\eta^2}{2\sigma} \left\| X^T X \right\| \left\| \Delta \alpha^t \right\|^2 \leq \eta \tau \Delta_t
\]

(26)
to satisfy the Armijo rule (9). We directly see from (23) that (26) holds for any \( \eta \in [0, 1] \) satisfying

\[
\eta \leq -\frac{2(1-\tau)\sigma}{\|X^T X\|} \frac{\Delta_t}{\left\| \Delta \alpha^t \right\|^2} \leq \frac{(1-\tau)\sigma(c_1 + c_2)}{\|X^T X\|}.
\]

Therefore, we have that under Algorithm 1, the step size generated is guaranteed to be

\[
\eta_t \geq \min\left(1, \frac{\beta(1-\tau)\sigma(c_1 + c_2)}{\|X^T X\|}\right).
\]

We will then establish convergence in the dual, and use this result to deduct the convergence in (1) of the primal iterates obtained by (6), which is the convergence that we really care about for solving ERM problems. We assume that in addition to Assumption 2.1, either of the following assumptions holds.

**Assumption 4.2** The function \( \xi \) is differentiable and its gradient is \( \rho \)-Lipschitz continuous for some \( \rho > 0 \).

That is,

\[
\| \nabla \xi(z_1) - \nabla \xi(z_2) \| \leq \rho \| z_1 - z_2 \|, \quad \forall z_1, z_2.
\]

**Assumption 4.3** The loss function \( \xi \) is \( L \)-Lipschitz continuous for some \( L > 0 \).

Moreover, the dual problem satisfies the Kurdyka-Lojasiewicz inequality with exponent 1/2 (Lojasiewicz, 1963, 1993; Kurdyka, 1998), for some \( \mu > 0 \). That is,

\[
f(\alpha) - f^* \leq \frac{\min_{s \in \partial \xi^*(-\alpha)} \| \nabla G(\alpha) + s \|^2}{2\mu}, \forall \alpha \in \Omega.
\]

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

These assumptions made on the sum \( \xi \) are less strict than demanding each \( \xi_i \) to satisfy certain properties.

We note that the conditions in Lemma 4.1 is weaker than that of most proximal methods such as Tseng and Yun (2009); Lee et al. (2014) as we do not need \( B_i \) to be positive definite, in which case \( Q_{B_i}^* \) is still strongly convex when Assumption 4.2 holds. In this situation we can have a broader choice of \( B_i \).

We first show that Assumption 4.2 also implies that (29) holds for the dual problem.

**Lemma 4.4** If \( f \) is \( \mu \)-strongly convex as defined in (4), then it satisfies (29) with the same \( \mu \).

**Proof** For any \( \lambda \in (0, 1] \), and any \( \alpha_1, \alpha_2 \), from the definition of strong convexity we have that

\[
f(\alpha_1) - f(\alpha_2) - \frac{\mu(1-\lambda)}{2} \| \alpha_1 - \alpha_2 \|^2 \geq \frac{f(\lambda \alpha_1 + (1-\lambda)\alpha_2) - f(\alpha_2)}{\lambda} \geq s^T(\alpha_1 - \alpha_2),
\]

for all \( s \in \partial f(\alpha_2) \). Let \( \lambda \to 0^+ \), we have that

\[
f(\alpha_1) - f(\alpha_2) \geq s^T(\alpha_1 - \alpha_2) + \frac{\mu}{2} \| \alpha_1 - \alpha_2 \|^2, \quad \forall s \in \partial f(\alpha_2).
\]

(30)
Now we fix $\alpha_2 = \alpha \in \Omega$, and minimize both sides of (30) with respect to $\alpha_1$ to get

$$f^* - f(\alpha) \geq -\frac{\|s\|^2}{2\mu} \quad \forall s \in \partial f(\alpha).$$

Since it holds for all $s \in \partial f(\alpha)$, we have proven (29) with parameter $\mu$ because $\partial f(\alpha) = \nabla G(\alpha) + \partial \xi(-\alpha)$. \hfill \Box

To prove the convergence of the whole algorithm, we need the following definition in our convergence discussion.

**Definition 4.5** Given any optimization problem

$$\min_{x \in X} f(x) \quad (31)$$

whose minimum is attainable and we denote it by $f^*$, we say that $x \in X$ is an $\epsilon$-accurate solution for (31) if

$$f(x) - f^* \leq \epsilon.$$

Now we are ready to show the global linear convergence of our algorithm for solving (2).

**Theorem 4.6** If (29) holds with $\mu > 0$ for the objective of (2), there exists $c_3$ such that $\|B_t\| \leq c_3$ for all $t$, and that the conditions in Lemma 4.1 are also satisfied for all iterations for some $c_1, c_2$, then the iterate generated by Algorithm 2 converges $Q$-linearly, and to obtain an $\epsilon$-accurate solution for (2), it takes at most

$$\max \left\{ O \left( \left( \left( \frac{\|X^T X\|}{\sigma} \right)^2 + c_3^2 \right) \frac{1}{\mu(c_1 + c_2)} \log \left( \frac{1}{\epsilon} \right) \right), \right. \\
O \left( \left( \left( \frac{\|X^T X\|}{\sigma} \right)^2 + c_3^2 \right) \frac{\|X^T X\|}{\mu(c_1 + c_2)^2 \beta \tau(1 - \tau)} \log \left( \frac{1}{\epsilon} \right) \right) \right\}$$

iterations.

**Proof** From (7), (23) and (9), we have that

$$f(\alpha^{t+1}) - f(\alpha^t) \leq -\eta_t \frac{c_1 + c_2}{2} \|\Delta \alpha^t\|^2. \quad (32)$$

From the optimality of $\Delta \alpha^t$ in (8), we get that

$$\nabla G^*(\alpha^t) + B_t \Delta \alpha^t + \hat{s}^{t+1} = 0, \quad (33)$$

for some $\hat{s}^{t+1} \in \partial \xi^*(-\alpha^t - \Delta \alpha^t)$. By convexity, that the step size is in $[0, 1]$ from Lemma 4.1, and the condition (29), we have

$$f(\alpha^{t+1}) - f^* \leq \eta_t \left( f(\alpha^t + \Delta \alpha^t) - f^* \right) + \left( 1 - \eta_t \right) \left( f(\alpha^t) - f^* \right)$$

$$\leq \eta_t \frac{\|\nabla G(\alpha^t + \Delta \alpha^t) + \hat{s}^{t+1}\|^2}{2\mu} + \left( 1 - \eta_t \right) \left( f(\alpha^t) - f^* \right). \quad (34)$$

Now to relate the first term to the decrease, we use (33) to get

$$\|\nabla G(\alpha^t + \Delta \alpha^t) + \hat{s}^{t+1}\|^2 \leq \|\nabla G^*(\alpha^t + \Delta \alpha^t) - \nabla G^*(\alpha^t) + \nabla G^*(\alpha^t) + \hat{s}^{t+1}\|^2$$

$$\leq 2\|\nabla G^*(\alpha^t + \Delta \alpha^t) - \nabla G^*(\alpha^t)\|^2 + 2\|B_t \Delta \alpha^t\|^2$$

$$\leq 2 \left( \frac{\|X^T X\|}{\sigma} \right)^2 \|\Delta \alpha^t\|^2 + 2\|B_t\|^2 \|\Delta \alpha^t\|^2, \quad (35)$$

12
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 Lipschitz continuity of \(\nabla G^*\). We therefore get the following by combining (34), (35), and (32).

\[
\begin{align*}
 f(\alpha^{t+1}) - f^* &\leq \frac{\eta_t}{\mu} \left( \left( \frac{\|X^T X\|}{\sigma} \right)^2 + c_3^2 \right) \|\Delta \alpha^t\|^2 + (1 - \eta_t) \left( f(\alpha^t) - f^* \right) \\
 &\leq \left( \left( \frac{\|X^T X\|}{\sigma} \right)^2 + c_3^2 \right) \frac{1}{2\mu(c_1 + c_2)\tau} \left( f(\alpha^t) - f(\alpha^{t+1}) \right) + (1 - \eta_t) \left( f(\alpha^t) - f^* \right).
\end{align*}
\]

Let us define
\[
c_4 := \left( \left( \frac{\|X^T X\|}{\sigma} \right)^2 + c_3^2 \right) \frac{1}{2\mu(c_1 + c_2)\tau},
\]
then rearranging (36) gives
\[
(f(\alpha^{t+1}) - f^*) \leq \frac{1 - \eta_t + c_4}{1 + c_4} (f(\alpha^t) - f^*).
\]

Combine the above result with the lower bound of \(\eta_t\) from (24) proves the global \(Q\)-linear convergence, and the number of iterations taken to get an \(\epsilon\)-accurate solution can be directly deduced from (37).

From Lemma 4.4, both Assumptions 4.2 and 4.3 satisfy the conditions required in Theorem 4.6. We note that in the final expression (37), a larger step size suggests a faster convergence for a fixed choice of \(B_t\). Therefore, it makes sense to conduct line search to try to find a larger \(\eta_t\) instead of using a fixed and conservative one that ensures (9) is satisfied, while \(c_4\) can be tuned through the choice of \(B_t\). However, we note that these are just the worst case iteration complexity, and empirically we often observe much faster convergence.

The next two results link the above linear convergence result for the dual problem to the convergence rate of the primal problem (1).

**Theorem 4.7** If Assumption 4.2 holds, then for any \(\epsilon > 0\) and any \(\epsilon\)-accurate solution \(\alpha\) for (2), the corresponding \(w\) obtained through (6) is \((\epsilon(\sigma + \rho\|X\|^2)/\sigma)\)-accurate.

If Assumption 4.3 holds, then for any \(\epsilon > 0\) and any \(\epsilon\)-accurate solution \(\alpha\) for (2), the corresponding \(w\) obtained through (6) is \((\min\{\rho\|X^2L^2/\sigma, \sqrt{8\epsilon\|X\|^2L^2}/\sigma\})\)-accurate.

**Proof** Our proof consists of using \(\alpha\) 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 bound the bound. Finally, we notice that the decrease of dual objective in one iteration of any algorithm is upper bounded by the distance to the optimum from the current objective, and that the primal sub-optimality is upper-bounded by the duality gap. Therefore we will obtain an algorithm-independent result from some algorithm-specific results.

When Assumption 4.2 holds, the primal problem is in the type of problems considered in Shalev-Shwartz and Zhang (2012), and we have that \(\xi^*\) is \((1/\rho)\)-strongly convex. If we take \(\alpha\) as the initial point, and apply one step of their method to obtain the next iterate \(\alpha^+\), from (Shalev-Shwartz and Zhang, 2012, Lemma 1), we get

\[
\epsilon = f(\alpha) - f(\alpha^+), \\
\geq f(\alpha) - f(\alpha^*) \\
\geq s \left[ f^p(w(\alpha)) + f(\alpha) \right] - \frac{s^2 G_x}{2\sigma} \\
\geq s \left[ f^p(w(\alpha)) - f^p(w^*) \right] - \frac{s^2 G_x}{2\sigma}, \forall s \in [0, 1],
\]

(38)
where $w^*$ is the optimal solution of (1),

$$G_s = (\|X\|^2 - \sigma(1-s)/s\rho)\|u - \alpha\|^2,$$

and

$$u_i \in -\partial\xi_i(X^T_i w(\alpha)).$$

To remove the second term in (38), we set

$$\|X\|^2 - \sigma(1-s)/s\rho = 0 \Rightarrow s = \frac{\sigma}{\sigma + \rho\|X\|^2} \in [0,1].$$

This then gives

$$\frac{\sigma + \rho\|X\|^2}{\sigma} \epsilon \geq f^P(w(\alpha)) - f^P(w^*).$$

Note that in (Shalev-Shwartz and Zhang, 2012, Lemma 1), the result is for the expected value of dual objective value decrease at the current iteration and the expected duality gap of the previous iterations. However, for the initial point, the expected duality gap is a constant, and the expected function decrease cannot exceed the distance from the current objective to the optimum.

When Assumption 4.3 holds, the primal problem falls in the type of problem discussed in Bach (2015). If we take $\alpha$ as the initial point, and apply one step of their method to obtain the next iterate $\alpha + \hat{s}$, from the last inequality in the proof of Proposition 4.2 in Bach (2015) and weak duality,

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

where

$$R^2 = \max_{\alpha, \hat{\alpha} \in \Omega} \|X(\alpha - \hat{\alpha})\|^2 \leq \|X\|^2 \max_{\alpha, \hat{\alpha} \in \Omega} \|(\alpha - \hat{\alpha})\|^2 = 4\|X\|^2L^2. \tag{40}$$

In the last equality we used (Rockafellar, 1970, Corollary 13.3.3) such that if $\phi(\cdot)$ is $L$-Lipschitz continuous, then the radius of $\text{dom}(\phi^*)$ is no larger than $L$. The right-hand side of (39) is concave with respect to $s$, hence we can obtain the maximum of it by setting the partial derivative with respect to $s$ to zero. By defining the maximizer as $\hat{s}$, this gives

$$\hat{s} = \arg \max_{s \in [0,1]} s (f^P(w) - f^P(w^*)) - \frac{(sR)^2}{2\sigma} = \min\{1, \frac{\sigma(f^P(w) - f^P(w^*))}{R^2}\}. \tag{39}$$

If $\hat{s} = 1$, we have

$$R^2 \leq \sigma(f^P(w) - f^P(w^*)),$$

and thus

$$f^P(w) - f^P(w^*) \leq \frac{R^2}{\sigma}.$$

On the other hand, if $\hat{s} \neq 1$, we get

$$\frac{2R^2\epsilon}{\sigma} \geq (f^P(w) - f^P(w^*))^2.$$

These conditions and (40) indicate that

$$f^P(w) - f^P(w^*) \leq \min\{\frac{R^2}{\sigma}, \sqrt{\frac{2\epsilon R^2}{\sigma}}\} = \min\{\frac{4\|X\|^2L^2}{\sigma}, \sqrt{\frac{8\epsilon\|X\|^2L^2}{\sigma}}\}. \blacksquare$$
Corollary 4.8 If we apply Algorithm 2 to solve a regularized ERM problem that satisfies either Assumption 4.2 or Assumption 4.3, then the primal iterates $\mathbf{w}^t$ obtained from the dual iterates $\alpha^t$ via (6) converges R-linearly.

Proof If Assumption 4.2 holds, Theorem 4.6 shows that the dual objective converges Q-linearly. Therefore, given any $\epsilon > 0$, from Theorem 4.7, it suffices to have an $\left(\epsilon/\left(\sigma + \rho \|X\|^2\right)\right)$-accurate dual solution to transform to an $\epsilon$-accurate primal solution, which takes

$$O\left(\log \left(\frac{1}{\epsilon} \|X\|^2\right)\right) = O\left(\log \frac{1}{\epsilon}\right)$$

iterations, showing R-linear convergence of the primal iterate. Note that here we omit all factors independent of $\epsilon$ to show the linear rate with respect to $\epsilon$, while the number of iterations needed can be calculated by combining Theorems 4.6 and 4.7.

On the other hand, if Assumption 4.3 holds, we then assume $\epsilon < 4\|X\|^2 L/\sigma$, for otherwise it does not take any iteration to get an $\epsilon$-accurate solution for the primal problem. Therefore, we need an $O(\epsilon^2)$-accurate dual solution to make the corresponding primal solution $\epsilon$ accurate, thus by Theorem 4.6, we need

$$O\left(\log \left(\frac{1}{\epsilon^2}\right)\right) = O\left(2 \log \frac{1}{\epsilon}\right) = O\left(\log \frac{1}{\epsilon}\right)$$

iterations. 

We note that the results in Theorem 4.7 and Corollary 4.8 are implied by existing works, and the calculations take very little effort. It is also not too difficult to obtain sublinear rates following similar techniques for problems that do not satisfy (29), but we omit them to simplify and focus our description.

5. Related Works

Our algorithm can be viewed from two different aspects. If we simply consider solving (8), then our method is similar to proximal quasi-Newton methods, with some specific pick of the second-order approximation. A generalization of it appears as the block coordinate descent method (Tseng and Yun, 2009), where the proximal quasi-Newton method is the special case that there is only one block of the variables. One thing worth noticing is that Tseng and Yun (2009) requires the matrix in (8) to be positive definite with a lower bound of the smallest eigenvalue over all iterations. We relaxed this condition to allow $B$ be indefinite or positive semidefinite when the $\xi^*$ term is strongly convex. This condition is used when Assumption 4.2 holds and in this case we do not need to add a damping term in our second order approximation. Namely, we can set $a_2^t \equiv 0$ in (15).

On the other hand, our focus is on how to devise a good approximation of the Hessian matrix of the smooth term to work efficiently for distributed optimization. Works focusing on this direction for dual ERM problems include Pechyony et al. (2011); Yang (2013); Ma et al. (2017). The work Pechyony et al. (2011) discusses how to solve the SVM dual problem distributedly. 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 (8) with $B_t$ defined in (15) with $a_1^t \equiv 1, a_2^t \equiv 0$ to obtain the update direction, while the step size $\eta_t$ is fixed to $1/K$. Though they did not provide theoretical convergence guarantee in Pechyony et al. (2011), we can see the reasoning of this choice from (25). First, one can easily see that

$$\|X \mathbf{d}\|^2 \leq K \mathbf{d}^T B_t \mathbf{d}, \quad \forall \mathbf{d},$$

with the equality holds when all $X_i$ are identical and $\mathbf{d}$ is a multiple of the vector of ones. Therefore, taking $\lambda = 1/K$ in (25) and plug in the bound in (41), we can see that minimizing (8) with a step size of $1/K$ leads to decrease of the objective value.
In Yang (2013), the algorithm DisDCA is proposed to solve (2) under the same 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 $c_i > 1$ easily. This method uses a specific algorithm, the stochastic dual coordinate descent (SDCA) method (Shalev-Shwartz and Zhang, 2016), 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 $\alpha_{i_k}$. If each time machine $k$ samples $m_k$ entries, and we denote

$$m := \sum_{k=1}^{K} m_k,$$

then the first variant of DisDCA, called the basic variant in Yang (2013), picks $B_t$ in (8) as

$$(B_t)_{i,j} = \begin{cases} m x_i^T x_j & \text{if } x_i, x_j \text{ are from the same } X_k \text{ for some } k \text{ that is sampled}, \\ 0 & \text{else}, \end{cases}$$

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 sampled. If we let $I$ be the indices not sampled, then following the same reasoning of (41), we have

$$\|X d\|^2 \leq d^T B_t d, \quad \forall d \text{ such that } d_i = 0,$$

where the equality holds when all $X$ are identical and $|I| = l - m$. Therefore, by combining (43) and (25), it is not hard to see that directly in this case minimizing $Q^\alpha_{B_t}$ directly results in a certain amount of function value decrease. The analysis in Yang (2013) then shows that the primal iterates $\{w_t\}$ obtained by using substituting the dual iterates $\{\alpha_t\}$ into (6) converges linearly to the optimum when all $\xi_i$ have Lipschitz continuous gradient and converges with a rate of $O(1/\epsilon)$ when all $\xi_i$ are Lipschitz continuous by using some proof techniques similar to that in Shalev-Shwartz and Zhang (2012), and as we noted in Section 4, this is actually the same as showing the decrease of the dual objective and then relate it to the primal objective. The key difference to our analysis is that they do not assume the further structure (29) of the dual problem when $\xi^*$ is not strongly convex, ergo the sublinear rate.

The second choice of $B_t$ in Yang (2013), called the practical variant, considers

$$(B_t)_{i,j} = \begin{cases} K x_i^T x_j & \text{if } \pi(i) = \pi(j), \\ 0 & \text{else}, \end{cases}$$

and takes unit step sizes. Similar to our discussion above for their basic variant, $Q^\alpha_{B_t}$ in this case is also an upper bound of the function value decrease if the step size is fixed to be 1, and we can expect this method to work better than the basic variant as the approximation is closer to the real Hessian while 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 of this variant in Yang (2013).

Both DSVM-AVE and the practical variant of DisDCA are generalized to a framework proposed in Ma et al. (2017) that discusses the relation between the second-order approximation and the step size. In particular, their theoretical analysis for a fixed step size starts from (25) and (41). They considered solving (8) with $B_t$ defined as

$$(B_t)_{i,j} = \begin{cases} a x_i^T x_j & \text{if } \pi(i) = \pi(j), \\ 0 & \text{else}, \end{cases}$$

and showed that for $a \in [1, K]$, a step size of $a/K$ is enough to ensure convergence of the dual objective. As we discussed above for Pechyony et al. (2011) and Yang (2013), this choice can be
proven to ensure objective value decrease from (25). Unlike DisDCA which is tied to SDCA, their framework allows arbitrary solver for the local sub-problems, and relates how precise the local sub-problems are solved with the convergence rate. If we ignore the part of local precision, which can be derived from the solution precision to the right-hand side of (25), the convergence rates of their framework shown in Ma et al. (2017) is similar to that of Yang (2013) for the basic variant of DisDCA. This work therefore provides theoretical convergence rates for both DSVM-AVE and the practical variant of DisDCA that are similar to the basic variant of DisDCA, and their experimental results on the local solver and the step size choices suggest that the practical variant of DisDCA is indeed the most efficient one.

When we take the same $B_t$ as these methods in (8), the major difference is that we do not take a pre-specified safe step size. Instead, we dynamically find a possibly larger step size that still guarantees function decrease. We can see that the choice of $a = 1$ in (44) gives too conservative the step size, while the choice of $a = K$ might make the quadratic approximation in (8) deviate from the real Hessian too far. In particular, 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 an assumption that holds true for most ERM losses, we are able to show linear convergence of a broader class of problems.

Most other practical 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 because the main computations are matrix-vector products like $Xw$, and it mostly takes some implementation modification to make these approaches efficient in distributed environments, so the main algorithmic framework remains the same. Among them, it has been shown that distributed truncated-Newton (Zhuang et al., 2015; Lin et al., 2014), distributed limited-memory BFGS (Chen et al., 2014), and the limited-memory common-directions method (Lee et al., 2017) are the most efficient ones in practice. These methods have the advantage that their convergences are invariant of the data partition, but they require the additional assumption that the primal objective is differentiable or even twice-differentiable. However, there are important cases of ERM problems that do not possess differentiable primal objective function. In these cases, one still needs to consider the dual approaches, for other wise the convergence might be slower. Another popular algorithm for distributedly solving (1) without requiring differentiability is the alternating direction method of multipliers (ADMM) (Boyd et al., 2011) which is widely used in consensus problems. However, ADMM is known to converge slowly in practice and the convergence also relies on the parameter of weighing the penalty term in the Augmented Lagrangian, and there is no known way to tune this parameter easily. It has been shown in Yang (2013) that DisDCA outperforms ADMM on SVM problems, while Zhuang et al. (2015) showed that distributed truncated-Newton is faster than ADMM on logistic regression problems.

Recently there are also many methods that focus on the communication efficiency. By adding more computation per iteration, they are able to use fewer communication rounds to achieve the same level of objective value. However, these approaches either rely on stronger assumption that data points across machines are independent and identically distributed, which may not hold true in practice because it is possible that each server gathers data from a specific region, or has higher computational dependency on the dimensionality of the problem, resulting in communication-efficient but computational-inefficient and thus impractical methods. We therefore exclude discussion of these methods.
6. Applications

In this section, we apply the proposed algorithm in Section 2 to solve various regularized ERM problems, and discuss some techniques that can be used to improve the efficiency by utilizing the problem structures. We will show the empirical performance of these algorithms in Section 7.

For what follows, we first show that a class of problems satisfy the condition (29).

Lemma 6.1 Given any initial point $\alpha^0$, a problem of the following form

$$\min_\alpha F(\alpha) := g(A\alpha) + b^T\alpha$$

subject to $C\alpha \leq d$, (45)

where $g$ is strongly convex, satisfies the condition (29) in the level set $\{\alpha \mid C\alpha \leq d, \ F(\alpha) \leq F(\alpha^0)\}$. If the constraint is a polytope, then the condition (29) holds for any feasible $\alpha$.

Proof From (Wang and Lin, 2014, Theorem 4.6), problem (45) satisfies a error bound condition with a parameter $\mu_{\alpha^0}$ that is continuous with respect to $\alpha^0$ within the level set, and (Bolte et al., 2015, Theorem 5) shows that this condition is equivalent to (29) with $\mu_{\alpha^0}$. This proves the first part of the theorem. When the constraint is a polytope, the feasible region is a compact set. Therefore, we can find the maximum of $\mu_{\alpha^0}$ within this compact set and letting this value be $\mu$, then (29) holds with the same $\mu$ for all feasible $\alpha$. ■

6.1 Binary Classification and Regression

The first case that we consider is the support vector machine (SVM) problem proposed by Boser et al. (1992); Vapnik (1995), such that $c_i \equiv 1$, and given $C > 0$,

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

where $y_i \in \{-1, 1\}, \forall i$. Obviously, Assumption 2.1 holds with $\sigma = 1$ in this case. After some simple calculation, we have that

$$g^*(X\alpha) = \frac{1}{2} \alpha^T X^T X \alpha, \ \xi_i^*(-\alpha_i) = I_{[0,C]}(\alpha_i y_i) - \alpha_i y_i,$$ (46)

where $I$ is the characteristic function of convex sets. It is clear that $\xi_i$ are C-Lipschitz continuous. By using results in Wang and Lin (2014), one can show that (29) is satisfied.

We can also replace the hinge loss in SVM with squared-hinge loss:

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

and then $\xi_i$ becomes differentiable, with the gradient being Lipschitz continuous. Therefore, Assumption 4.2 is satisfied. We have that

$$\xi_i^*(-\alpha_i) = I_{[0,\infty]}(\alpha_i y_i) - \alpha_i y_i + \frac{\alpha_i^2}{4C}.$$ (47)

One can observe that the dual objectives of both the hinge loss and squared-hinge loss SVM are quadratic functions, hence we can apply the exact line search approach in (21) with very low cost by utilizing the $\Delta v$ and $v$ vectors. Moreover, since the dual problems are box-constrained, it is clear that both cases are of the form described in (45) so our analysis of linear convergence applies.

Another widely used classification model is logistic regression, where

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

It can then be shown that the logistic loss is infinitely differentiable, and its gradient is Lipschitz continuous. Thus Assumption 4.2 is satisfied.

An analogy of SVM to regression is support vector regression (SVR) by Boser et al. (1992); Vapnik (1995) such that given $C > 0$ and $\epsilon \geq 0$,

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

with $y_i \in \mathbb{R}$ for all $i$. Similar to the case of SVM, the first case satisfies Assumption 4.3 and the latter satisfies Assumption 4.2. Note that the degenerate case of $\epsilon = 0$ corresponds to absolute deviation loss and least square loss. In the case of 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 its possible that (29) 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 $\xi_i$ we discussed in this section is summarized 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 $y_i \in \{1, \ldots, T\}$ for all $i$. The first model we consider is the multi-class SVM model proposed by Crammer and Singer (2002). Given an original feature vector $x_i \in \mathbb{R}^{\tilde{n}}$, the data matrix $X_i$ is defined as $(I_T - e_i 1^T) \otimes x_i$, where $I_T$ is the $T$ by $T$ identity matrix, $e_i$ is the unit vector of the $i$-th coordinate, $1$ 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

$$g(w) := \frac{1}{2} w^T w,$$

$$\xi_i(z) := C \max_{1 \leq j \leq T} \left( \max_{1 \leq j \leq T} 1 - z_i, 0 \right).$$

2. Up to a reformulation of the dual problem by setting $\alpha = \alpha^+ - \alpha^-$ and $\alpha^+, \alpha^- \geq 0$. 

19
From the first glance, this $\xi$ seems to be not even Lipschitz continuous. However, its dual formulation is

$$\min_{\alpha} \frac{1}{2} \alpha^T X^T X \alpha + \sum_{i=1}^{l} \sum_{j \neq y_i} (\alpha_i)_j$$

subject to $\alpha^T 1 = 0, i = 1, \ldots, l,$

$$(\alpha_i)_j \leq 0, \forall j \neq y_i, i = 1, \ldots, l,$$

$$(\alpha_i)_{y_i} \leq C, i = 1, \ldots, l,$$

(48)

showing the boundedness of the dual variables $\alpha$. Thus the primal variable $w(\alpha) = X \alpha$ also lies in a bounded area. Therefore, $\xi_i(X^T w(\alpha))$ also has a bounded domain, indicating that we can find $L \geq 0$ such that this continuous function is Lipschitz continuous within this domain. Moreover, the formulation (48) satisfies the form in (45), so (29) holds. Therefore, Assumption 4.3 is satisfied.

Note that in this case the objective of (48) is a quadratic function so once again we can apply the exact line search method for this problem.

Being an analogy of SVM, one can also consider the squared-hinge loss for multi-class SVM (Lee and Lin, 2013).

$$\xi_i(z) := C \max \left( \max_{1 \leq j \leq T} 1 - z_i, 0 \right)^2.$$  (49)

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 $\xi$. The dual formulation from the derivation of Lee and Lin (2013) is

$$\min_{\alpha} \frac{1}{2} \alpha^T X^T X \alpha + \sum_{i=1}^{l} \sum_{j \neq y_i} (\alpha_i)_j + \sum_{i=1}^{l} \frac{(\alpha_i)_{y_i}^2}{4C}$$

subject to $\alpha^T 1 = 0, i = 1, \ldots, l,$

$$(\alpha_i)_j \leq 0, \forall j \neq y_i, i = 1, \ldots, l,$$

suggested that each coordinate of $\alpha$ is only one-side-bounded, so this is not the case that $\alpha$ lies explicitly in a compact set. However, from the constraint and the objective, we can see that given any initial point $\alpha^0$, the level set of $\{ \alpha \mid f(\alpha) \leq f(\alpha^0) \}$ is compact. Because our algorithm is a descent method, throughout the optimization process, all $\alpha$ implicitly lies within this compact set. This again indicates that $w(\alpha)$ and $X^T w(\alpha)$ are within a bounded set, proving the Lipschitzness of $\xi$ when our algorithm is applied. The condition (29) is also satisfied following the same argument for the hinge-loss case. Therefore Assumption 4.3 still holds, and it is obvious that we can use the exact line search method for this problem 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

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

It is not hard to see that Assumption 4.2 holds for this problem. For more details of its dual problem and an efficient local sub-problem solver, interested readers are referred to Yu et al. (2011).

### 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
that the value 1 in the multi-class losses are replaced by the value of $\Delta(\cdot)$ (see the similarity between (51) and the multi-class losses (47)-(49), where the major difference is the constraints are not satisfied, then a penalty term $\psi_i$ is introduced to the objective function, where $\ell(\psi)$ defines the loss term. Similar to the binary classification and multi-class classification cases, common choices of the loss functions are the L2 loss, $\ell_2$, and the L1 loss, $\ell_1$. The SSVM problem in (50) fits into our framework, depending on the definition of the features, one can define $X_i$ to encode the output $y$. One example is to set every column of $X_i$ as a vector of the form $\phi(y, y_i, x_i)$ with different $y \in Y_i$ such that $c_i = |Y_i|$, and let $\xi_i(X_i^T w)$ in problem (1) be

$$\xi_i(z) = C \max_y \ell(\Delta(y, y_i) - z_y).$$

Here, we use the order of $y$ appeared in the columns of $X_i$ as the enumeration order for the coordinates of $z$.

We consider solving problem (50) in its dual form (Tsochantaridis et al., 2005). One can clearly see the similarity between (51) and the multi-class losses (47)-(49), where the major difference is the value 1 in the multi-class losses are replaced by the value of $\Delta(y, y_i)$. 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 (50) can be written as,

$$\min_{\alpha} \frac{1}{2} \alpha^T X^T X \alpha - \sum_{i=1}^{l} \sum_{y \neq y_i} \Delta(y_i, y)(\alpha_i) y_i$$

subject to

$$\alpha_i^T 1 = 0, i = 1, \ldots, l,$$

$$\alpha_i y \leq 0, \forall y \neq y_i, i = 1, \ldots, l,$$

$$\alpha_i y_{y_i} \leq C, i = 1, \ldots, l.$$

$$\min_{\psi} \frac{1}{2} w^T w + C \sum_{i=1}^{l} \ell(\psi_i)$$

subject to $w^T \phi(y, y_i, x_i) \geq \Delta(y_i, y) - \psi_i, \forall y \in Y_i, i = 1, \ldots, l,$

where $C > 0$ is a predefined parameter. $\phi(y, y_i, x_i) = \Phi(x_i, y_i) - \Phi(x_i, y)$, and $\Phi(x, y)$ is a generated feature vector depending on both the input $x$ and the output structure $y$. 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 (50) specify that the difference between the score assigned to the correct output structure should be higher than a predefined scoring metric $\Delta(y, y_i) \geq 0$ that represents the distance between output structures. If the constraints are not satisfied, then a penalty term $\psi_i$ is introduced to the objective function, where $\ell(\psi)$ defines the loss term. Similar to the binary classification and multi-class classification cases, common choices of the loss functions are the L2 loss, $\ell_2$, and the L1 loss, $\ell_1$. The SSVM problem in (50) fits into our framework, depending on the definition of the features, one can define $X_i$ to encode the output $y$. One example is to set every column of $X_i$ as a vector of the form $\phi(y, y_i, x_i)$ with different $y \in Y_i$ such that $c_i = |Y_i|$, and let $\xi_i(X_i^T w)$ in problem (1) be

$$\xi_i(z) = C \max_y \ell(\Delta(y, y_i) - z_y).$$

Here, we use the order of $y$ appeared in the columns of $X_i$ as the enumeration order for the coordinates of $z$.

We consider solving problem (50) in its dual form (Tsochantaridis et al., 2005). One can clearly see the similarity between (51) and the multi-class losses (47)-(49), where the major difference is the value 1 in the multi-class losses are replaced by the value of $\Delta(y, y_i)$. 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 (50) can be written as,
And, with the L2 loss, the dual of (50) is

$$\min_{\alpha} \frac{1}{2} \alpha^T X^T X \alpha - \sum_{i=1}^{l} \sum_{y \neq y_i} \Delta(y_i, y)(\alpha_i)_{y_i} + \sum_{i=1}^{l} \frac{((\alpha_i)_{y_i})^2}{4C}$$

subject to  
$$\alpha^T 1 = 0, i = 1, \ldots, l,$$
$$\alpha_i \leq 0, \forall y \neq y_i, i = 1, \ldots, l$$

(53)

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

The key challenge of solving problems (52) and (53) is that for most applications, the size of $\mathcal{Y}_i$ and thus the dimension of $\alpha$ is exponentially large (with respect to the length of $x_i$), so optimizing over all variables is unrealistic. Efficient dual methods (Tsochantaridis et al., 2005; Lacoste-Julien et al., 2013; Chang and Yih, 2013) 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. 3 The working set is selected using the sub-gradient of (50) with respect to the current iterate. Specifically, for each training instance $x_i$, we add the dual variable $\alpha_i, \hat{y}$ corresponding to the structure $\hat{y}$ into the working set, where

$$\hat{y} = \arg \max_{y \in \mathcal{Y}_i} w^T \phi(y, y_i, x_i) - \Delta(y_i, y).$$

(54)

Once $\alpha$ is updated, we update $w$ accordingly. We call the step of computing eq. (54) “inference”, and call the part of optimizing Eq. (52) or (53) over a fixed working set “learning”. When training SSVM distributedly, the learning step involves communication across machines. Therefore, inference and learning steps are both expensive.

SSVM is an extension of multi-class SVM for the structured prediction cases. Similar, conditional random fields (CRF) (Lafferty et al., 2001) extends multinomial logistic regression. The loss function in CRF is defined as the negative log-likelihood:

$$\xi_i(z) := - \log \left( \frac{\exp(z_{y_i})}{\sum_{y \in \mathcal{Y}_i} \exp(z_y)} \right).$$

Similar to multinomial logistic regression, Assumption 4.2 holds for this loss function.

7. Experiments

We conduct experiments on different ERM problems to examine the efficiency of variant realizations of our framework. The problems range from simple binary and regression problem (i.e., $c_i \equiv 1$) to problems with complex output structures (i.e., each $c_i$ are different), and from that exact line search can be conducted to that backtracking using Algorithm 1 is required. For each problem, we compare our method with state of the art for that specific problem.

For the case $c_i \equiv 1$, we consider two linear classification tasks. Regarding the situation of larger $c_i$, we take L2-regularized SSVM as the exemplifying application.

7.1 Binary Linear Classification

For binary linear classification, From existing study for the single machine case, solving the dual problem is more favorable when the feature dimension is larger than the number of instances because the number of variables is smaller. We therefore consider data sets with $l < n$, shown in Table 2, in

3. This approach is related to applying the cutting-plane methods to solve the primal problem (50) (Tsochantaridis et al., 2005; Joachims et al., 2009).
this task. We consider both linear SVM and L2-regularized logistic regression discussed in Section 6.1. The comparison criterion is the relative primal or dual objective distance to the optimum, respectively defined as

\[ \frac{|f(\alpha^t) - f^*|}{f^*}, \quad \frac{|f^P(w(\alpha^t)) - f^*|}{f^*}, \]

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, according to strong duality. We examine the relation between these values and the running time. We fix \( C = 1 \) in this experiment. The distributed environment is a local cluster with 16 nodes running MPI.

We compare the methods below whenever they are applicable for the problem.

- **DisDCA** (Yang, 2013): we consider the practical variant because it is shown to be faster than the basic variant. Moreover, experimental result in Ma et al. (2017) showed that this algorithm is faster than DSVM-AVE, and the best local solver for the per-machine sub-problem is indeed the coordinate descent method used in Yang (2013).

- **BDA**: the Block-Diagonal Approximation method proposed in this paper. For the dual SVM problem, we utilize the fact that it has a quadratic objective to conduct exact line search efficiently. For logistic regression, backtracking line search is used.

- **L-CommDir** (Lee et al., 2017): this is a state-of-the-art distributed primal ERM solver that has been shown to empirically outperforms existing distributed primal approaches. This algorithm requires differentiability of the primal objective, so we only apply it on the squared-hinge loss SVM problem and logistic regression. We use the implementation in the package MPI-LIBLINEAR 2.1.4

- **TRON** (Zhuang et al., 2015): this is a distributed implementation of the trust-region truncated Newton method for unconstrained optimization proposed by Steihaug (1983). This algorithm also requires differentiability of the primal objective, so it is applied on only squared-hinge loss SVM and logistic regression problems. The implementation of this method is also from MPI-LIBLINEAR 2.1.

For both DisDCA and BDA, we implement them in C++ and the local solver being considered is random permutation cyclic coordinate descent (RPCD) for dual SVM (Hsieh et al., 2008) and dual logistic regression (Yu et al., 2011). For the choice of \( B_t \) in (15) for our method, we use \( a_t^1 = 1 \) for all problems, as it is the closest approximation of the real Hessian, while regarding \( a_t^2 \), it is set to \( 10^{-3} \) for the hinge-loss SVM problem and 0 for the other two. The other two algorithms are also implemented in C++. At every iteration, we run one epoch of RPCD on each machine, namely we pass through the whole data set once, before communication. For L-CommDir, we take the experimental setting in Lee et al. (2017) to use historical information from the previous five iterations.

The comparison of the dual objective is shown in Figure 1. 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 exact line search has its advantage over backtracking, especially because its cost is low, 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 modeling of the objective difference such that the local updates can be directly applied to ensure the objective decrease, this upper bound might be too conservative, and more aggressive upper bound modelings might be computationally impractical to obtain. On the other hand, our approach provides an efficient way to estimate how aggressive the current update can be, according to the current iterate. Therefore the objective can decrease faster as the update is more aggressive but still safe to guarantee sufficient

4. http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/distributed-liblinear/.
Table 2: Data statistics.

| Data set     | #instances | #features | #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     |

objective value decrease. Investigation on the step sizes at different iterations of the line search approaches shows that the step size is not fixed throughout the optimization procedure, indicating that a fixed step size scenario might not be an ideal choice.

The comparison of the primal objective is provided in Figure 2. Note that the step-like behavior of BDA is from that we use the best primal objective so far as we discussed in Section 3.4. For the case of hinge-loss SVM, our method is 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, our method outperforms all approaches regardless of whether the primal or the dual problem is solved. The dual problem of logistic regression is not a simple quadratic one, hence we cannot easily implement exact line search and need to resort to the backtracking approach. We can see that for this problem, L-CommDir has advantages in the later stage of optimization, while our method and DisDCA are competitive in the early stage, 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 (Marcus et al., 1993) 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 (Viterbi, 1967). 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 McDonald et al. (2005), where we find the highest scoring parse using the Chu-Liu-Edmonds algorithm (Chu and Liu, 1965; Edmonds, 1967). 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. Data sets are uniformly randomly split into eight partitions.

- **BDA**: the proposed algorithm. We take $a_1^t = K$ and $a_2^t = 10^{-3}$ as $a_1^t = 1$ is less stable in the primal objectives, which is essential for the sub-problem solver in this application.

- **ADMM**: distributed alternating directions method of multiplier discussed in Boyd et al. (2011).

- **Perceptron**: a parallel structured perceptron algorithm described in McDonald et al. (2010).

- **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, the problem considered is SSVM in (50) with L2 loss. Perceptron, on the other hand, solves a similar but different problem such that no regularization is involved. We set $C = 0.1$.
Figure 1: Comparison of different algorithms for optimizing the dual ERM problem with $C = 1$. We show training time v.s. relative difference of the dual objective to the optimal function value. Running time is in log scale.
Figure 2: Comparison of different algorithms for optimizing the ERM problem with $C = 1$. We show training time v.s. relative difference of the primal objective to the optimal function value. The step-like behavior of BDA is resulted from the pocket approach discussed in Section 3.4.
for SSVM. Empirical experience suggest that structured SVM is not sensitive to $C$, and the model trained with $C = 0.1$ often attains reasonable test performance.

Both ADMM and BDA decompose the original optimization problem into sub-problems and solve the sub-problems by the dual coordinate descent solver proposed in Chang and Yih (2013), 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 and BDA. For all algorithms, we fix the number of inferences through all instances between any two rounds of communication 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 to obtain a better convergence result, fine-tuning this parameter is not realistic for users. For BDA and ADMM, each time in 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, the weight for the penalty term in the augmented Lagrangian also affects the convergence speed. Instead of fine-tuning it, a fixed value of 1.0 is used. Note that since Perceptron and BDA/ADMM consider different problems, instead of showing objective function values, we compare the test performance along training time of these methods.

Figure 3 shows the results. The x-axis is in log-scale. Although averaging local classifiers achieves reasonable performance, all 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 fast convergence rate.

8. Discussion

Our analysis discusses the situation of exactly minimizing (8) at each round, while the setting in the experiment seems not satisfying so. However, by linear convergence of cyclic coordinate descent on composite problems (see, for example, Wang and Lin (2014); Hong et al. (2013)), we can ensure

---

5. See, for example, Boyd et al. (2011) for details.
that the problem is solved at least to some precision. Therefore, one can ensure that in this case $\Delta_t$ is still bounded by some amount related to the norm of the obtained update direction. Hence, we can find another matrix $\tilde{B}_t$ such that the update direction is the solution of (8) with this matrix, and the conditions in Lemma 4.1 and Theorem 4.6 are still satisfied because the degree of freedom of the matrix is large enough. The key requirement here is that we need to control how exact or how loose the solutions of (8) at different iterations are. As long as we require that at each iteration it is solved at least to a fixed multiplicative precision, the values of $c_1, c_2, c_3$ from $\tilde{B}_t$ will be bounded over iterations. We omit the proof details as the convergence theorems are just to provide a guarantee in the worst case while we observe from the experiments that the empirical convergence speed is much faster than that in the worst-case analysis.

As aforementioned, if the block-diagonal $B_t$ is closer to the Hessian of $f$, then empirical convergence is expected to be faster. A way to achieve this is to have a better data partition such that those off diagonal-block entries are as small as possible. However, repartitioning the data across machines involves a significant amount of data transmission, and thus is not a practical approach in general. A feasible scenario is when the data points are streamed in, then when a new data point arrives, we can conduct some approach to decide which machine to put this data point to achieve this goal approximately. However, this may make the partition imbalanced, so that the computational burden is not well parallelized. How to achieve a data partition that is simultaneously balanced and has smaller off diagonal-block entries is an interesting future research direction.

As we noted in (37), one always want to achieve a larger step size simultaneously with a smaller $c_4$, and balancing these two factors is not an easy task in practice as the step size is not fixed. One possible heuristic is to dynamically adjust $a_{t1}$ and $a_{t2}$ according to the step size of the previous iteration. When the step size is smaller than a threshold, we can enlarge $a_{t1}$ and $a_{t2}$ by a factor, while when it is large enough, one can try to make $(a_{t1}, a_{t2})$ closer to $(1, 0)$ than the previous iteration.

One limitation of our method is its strong scaling with number of machines is not promising. If we fix the data set but increase the number of machines we use, then one can see that $B_t$ will contain more zero entries. Therefore, one can expect our algorithm gets closer to the proximal gradient method and converges slower in practice. This is inevitable for all distributed optimization dual methods we discussed in Section 5. However, in practice, one apply distributed optimization for (1) or (2) not for the purpose of accelerating the training process, but because of the data storage nature. Usually the data set is originally stored distributedly, and therefore as just discussed, repartitioning it is costly in communication. Therefore, the strong scaling scenario is not the normal case in distributed optimization. Practitioners concern more about how to make the optimization procedure more efficient given a fixed number of machines and a fixed data partition.

9. Conclusions
In this work, we propose a framework for distributedly optimizing 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 problems under a condition weaker than strong convexity for the dual problem. 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 dual approaches for regularized empirical risk minimization, and is competitive to cutting-edge distributed primal methods.

References
Francis Bach. Duality between subgradient and conditional gradient methods. *SIAM Journal on Optimization*, 25(1):115–129, 2015.
Jérôme Bolte, Trong Phong Nguyen, Juan Peypouquet, and Bruce W Suter. From error bounds to the complexity of first-order descent methods for convex functions. *Mathematical Programming*, pages 1–37, 2015.

Bernhard E. Boser, Isabelle Guyon, and Vladimir Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, pages 144–152. ACM Press, 1992.

Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine Learning*, 3(1):1–122, 2011.

Ming-Wei Chang and Wen-tau Yih. Dual coordinate descent algorithms for efficient large margin structural learning. *Transactions of the Association for Computational Linguistics*, 2013.

Weizhu Chen, Zhenghao Wang, and Jingren Zhou. Large-scale L-BFGS using MapReduce. In *Advances in Neural Information Processing Systems*, pages 1332–1340, 2014.

Yoeng-Jin Chu and Tseng-Hong Liu. On shortest arborescence of a directed graph. *Scientia Sinica*, 14(10):1396, 1965.

Koby Crammer and Yoram Singer. On the learnability and design of output codes for multiclass problems. *Machine Learning*, (2–3):201–233, 2002.

Jack Edmonds. Optimum branchings. *Journal of Research of the national Bureau of Standards B*, 71(4):233–240, 1967.

Stephen I. Gallant. Perceptron-based learning algorithms. *Neural Networks, IEEE Transactions on*, 1(2):179–191, 1990.

Jean-Baptiste Hiriart-Urruty and Claude Lemaréchal. *Fundamentals of convex analysis*. Springer Science & Business Media, 2001.

Mingyi Hong, Xiangfeng Wang, Meisam Razaviyayn, and Zhi-Quan Luo. Iteration complexity analysis of block coordinate descent methods. *Mathematical Programming*, pages 1–30, 2013.

Cho-Jui Hsieh, Kai-Wei Chang, Chih-Jen Lin, S. Sathiya Keerthi, and Sellamanickam Sundararajan. A dual coordinate descent method for large-scale linear SVM. In *Proceedings of the Twenty Fifth International Conference on Machine Learning (ICML)*, 2008.

Thorsten Joachims, Thomas Finley, and Chun-Nam John Yu. Cutting-plane training of structural SVMs. *Machine Learning Journal*, 77(1):27–59, 2009.

Krzysztof Kurdyka. On gradients of functions definable in o-minimal structures. In *Annales de l’institut Fourier*, volume 48, pages 769–783, 1998.

Simon Lacoste-Julien, Martin Jaggi, Mark Schmidt, and Patrick Pletscher. Block-coordinate Frank-Wolfe optimization for structural svms. In *Proceedings of the Twenty Fifth International Conference on Machine Learning (ICML)*, 2013.

John Lafferty, Andrew McCallum, and Fernando Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*, 2001.

Ching-pei Lee and Chih-Jen Lin. A study on L2-loss (squared hinge-loss) multi-class SVM. *Neural Computation*, 25(5):1302–1323, 2013.
Ching-pei Lee and Dan Roth. Distributed box-constrained quadratic optimization for dual linear SVM. In \textit{Proceedings of the 32nd International Conference on Machine Learning (ICML)}, 2015.

Ching-pei Lee, Kai-Wei Chang, Shyam Upadhyay, and Dan Roth. Distributed training of structured SVM. In \textit{NIPS Workshop on Optimization for Machine Learning}, 2015.

Ching-pei Lee, Po-Wei Wang, Weizhu Chen, and Chih-Jen Lin. Limited-memory common-directions method for distributed optimization and its application on empirical risk minimization. In \textit{Proceedings of SIAM International Conference on Data Mining}, 2017.

Jason D. Lee, Yuekai Sun, and Michael A. Saunders. Proximal newton-type methods for minimizing composite functions. \textit{SIAM Journal on Optimization}, 24(3):1420–1443, 2014.

Chieh-Yen Lin, Cheng-Hao Tsai, Ching-pei Lee, and Chih-Jen Lin. Large-scale logistic regression and linear support vector machines using Spark. In \textit{Proceedings of the IEEE International Conference on Big Data}, pages 519–528, 2014.

Hongzhou Lin, Julien Mairal, and Zaid Harchaoui. A universal catalyst for first-order optimization. In \textit{Advances in Neural Information Processing Systems}, pages 3384–3392, 2015.

Stanislas Lojasiewicz. Sur la géométrie semi-et sous-analytique. In \textit{Annales de l'institut Fourier}, volume 43, pages 1575–1595, 1993.

Stanislaw Lojasiewicz. Une propriété topologique des sous-ensembles analytiques réels. In \textit{Les Équations aux Dérivées Partielles}. Éditions du centre National de la Recherche Scientifique, 1963.

Chenxin Ma, Jakub Konečný, Martin Jaggi, Virginia Smith, Michael I Jordan, Peter Richtárik, and Martin Takáč. Distributed optimization with arbitrary local solvers. \textit{Optimization Methods and Software}, pages 1–36, 2017.

Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. Building a large annotated corpus of English: The Penn Treebank. \textit{Computational linguistics}, 19(2):313–330, 1993.

Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajič. Non-projective dependency parsing using spanning tree algorithms. In \textit{Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing}, 2005.

Ryan McDonald, Keith Hall, and Gideon Mann. Distributed training strategies for the structured Perceptron. In \textit{Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics}, 2010.

Dmitry Pechyony, Libin Shen, and Rosie Jones. Solving large scale linear svm with distributed block minimization. In \textit{NIPS 2011 Workshop on Big Learning: Algorithms, Systems, and Tools for Learning at Scale}. 2011.

R. Tyrrell Rockafellar. \textit{Convex Analysis}. Princeton University Press, Princeton, NJ, 1970.

Shai Shalev-Shwartz and Tong Zhang. Proximal stochastic dual coordinate ascent. Technical report, 2012. arXiv preprint arXiv:1211.2717.

Shai Shalev-Shwartz and Tong Zhang. Accelerated proximal stochastic dual coordinate ascent for regularized loss minimization. \textit{Mathematical Programming}, 155(1-2):105–145, 2016.

Trond Steihaug. The conjugate gradient method and trust regions in large scale optimization. \textit{SIAM Journal on Numerical Analysis}, 20:626–637, 1983.
Ruoyu Sun and Yinyu Ye. Worst-case complexity of cyclic coordinate descent: $O(n^2)$ gap with randomized version. Technical Report arXiv:1604.07130, Department of Management Science and Engineering, Stanford University, Stanford, California, April 2016.

Ben Taskar, Carlos Guestrin, and Daphne Koller. Max-margin Markov networks. In Advances in Neural Information Processing Systems 16. MIT Press, Cambridge, MA, 2004.

Paul Tseng and Sangwoon Yun. A coordinate gradient descent method for nonsmooth separable minimization. Mathematical Programming, 117:387–423, 2009.

Ioannis Tsochantaridis, Thorsten Joachims, Thomas Hofmann, and Yasemin Altun. Large margin methods for structured and interdependent output variables. Journal of Machine Learning Research, 6:1453–1484, 2005.

Vladimir Vapnik. The Nature of Statistical Learning Theory. Springer-Verlag, New York, NY, 1995.

Andrew Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. IEEE transactions on Information Theory, 13(2):260–269, 1967.

Po-Wei Wang and Chih-Jen Lin. Iteration complexity of feasible descent methods for convex optimization. Journal of Machine Learning Research, 15:1523–1548, 2014.

Tianbao Yang. Trading computation for communication: Distributed stochastic dual coordinate ascent. In Advances in Neural Information Processing Systems 26, pages 629–637. 2013.

Hsiang-Fu Yu, Fang-Lan Huang, and Chih-Jen Lin. Dual coordinate descent methods for logistic regression and maximum entropy models. Machine Learning, 85(1-2):41–75, October 2011.

Guo-Xun Yuan, Chia-Hua Ho, and Chih-Jen Lin. Recent advances of large-scale linear classification. Proceedings of the IEEE, 100(9):2584–2603, 2012.

Yuchen Zhang and Xiao Lin. DiSCO: Distributed optimization for self-concordant empirical loss. In Proceedings of the Twenty Fifth International Conference on Machine Learning (ICML), 2015.

Shun Zheng, Fen Xia, Wei Xu, and Tong Zhang. A general distributed dual coordinate optimization framework for regularized loss minimization. Technical report, 2017. arXiv preprint arXiv:1604.03763.

Yong Zhuang, Wei-Sheng Chin, Yu-Chin Juan, and Chih-Jen Lin. Distributed Newton method for regularized logistic regression. In Proceedings of the Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD), 2015.

Hui Zou and Trevor Hastie. Regularization and variable selection via the elastic net. Journal of the Royal Statistical Society: Series B (Statistical Methodology), 67(2):301–320, 2005.