Communication-Efficient Distributed Primal-Dual Algorithm for Saddle Point Problems

Yaodong Yu[*]
Nanyang Technological University
ydyu@ntu.edu.sg

Sulin Liu[*]
Nanyang Technological University
liusl@ntu.edu.sg

Sinno Jialin Pan
Nanyang Technological University
sinnopan@ntu.edu.sg

Abstract

Primal-dual algorithms, which are proposed to solve reformulated convex-concave saddle point problems, have been proven to be effective for solving a generic class of convex optimization problems, especially when the problems are ill-conditioned. However, the saddle point problem still lacks a distributed optimization framework where primal-dual algorithms can be employed. In this paper, we propose a novel communication-efficient distributed optimization framework to solve the convex-concave saddle point problem based on primal-dual methods. We carefully design local subproblems and a central problem such that our proposed distributed optimization framework is communication-efficient. We provide a convergence analysis of our proposed algorithm, and extend it to address non-smooth and non-strongly convex loss functions. We conduct extensive experiments on several real-world datasets to demonstrate competitive performance of the proposed method, especially on ill-conditioned problems.

1 INTRODUCTION

In the era of big data, developing distributed machine learning algorithms has become increasingly important yet challenging. In this work, we focus on developing a new distributed optimization algorithm for regularized *empirical risk minimization* (ERM), which is a generic class of convex optimization problems that arises often from machine learning. Specifically, our goal is to minimize the empirical loss defined over n data samples:

$$\min_{\mathbf{x}\in\mathbb{R}^d} \mathcal{P}(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n} \phi_i(a_i^\top \mathbf{x}) + g(\mathbf{x}), \qquad (1)$$

where $a_1, ..., a_n \in \mathbb{R}^d$ are feature vectors of n data points, $\phi_i : \mathbb{R} \to \mathbb{R}$ is a convex loss function with the linear predictor $a_i^\top \mathbf{x}$, for $i = 1, ..., n$, and $g : \mathbb{R}^d \to \mathbb{R}$ is a convex regularization function for the predictor $\mathbf{x} \in \mathbb{R}^d$. Suppose that a distributed system consists of K machines, and each machine has access only a subset $\mathcal{P}_k$ of the data $[n] := \{1, ..., n\}$, where $\{\mathcal{P}_k\}_{k=1}^K$ is a given partition of the dataset $[n]$, and we denote by $n_k = |\mathcal{P}_k|$.

1.1 COMMUNICATION-EFFICIENT DISTRIBUTED OPTIMIZATION

One of the most important issues in distributed optimization is the communication efficiency because communication between machines is much more expensive than reading data from the memory on local machines. Therefore, more and more efforts have been made in proposing communication-efficient methods in distributed optimization (Jaggi et al., 2014, Ma et al., 2015, Reddi et al., 2016, Shamir et al., 2014, Smith et al., 2015, Yang, 2013). The basic idea behind these methods is to carefully design local computation and communication in a distributed system. To achieve this goal, existing methods usually decompose the optimization problem into K local subproblems, denoted by $\mathcal{L}_k$, with respect to local data of each machine. After each local machine performs an arbitrary optimization method on $\mathcal{L}_k$ and solves it approximately, updated information from local machines is sent to a central node to carry out a central update. This allows one to control the trade-off between communication and local computation, which is more flexible in distributed setting.

In general, existing communication-efficient methods can be classified into two categories. A first category

[*]Indicates equal contributions.

is referred to as *gradient*-type, which aims to decompose the problem (1) into K local subproblems and solve each subproblem using gradient descent methods, such as stochastic variance reduced gradient (SVRG) (Johnson and Zhang, 2013), on each local machine independently, and then update the optimization variables by doing an average of local variables of each machine (Reddi et al., 2016, Shamir et al., 2014).

The second one is referred to as *coordinate*-type, which usually focuses on solving the dual problem of (1),

$$\max_{\boldsymbol{\alpha}\in\mathbb{R}^n} \mathcal{D}(\boldsymbol{\alpha}) = \frac{1}{n}\sum_{i=1}^{n} -\phi_i^*(\alpha_i) - g^*(-\frac{1}{n}\sum_{i=1}^{n}\alpha_i a_i), \quad (2)$$

where ϕ_i^* is the conjugate function of ϕ_i, $\boldsymbol{\alpha}$ is the dual variable vector with the i-th element being α_i. This category of methods decomposes the dual problem (2) into K local subproblems with regard to each machine, and employs a coordinate-type method, such as stochastic dual coordinate ascent method (SDCA) (Shalev-Shwartz and Zhang, 2013), to solve each local subproblem. After each local subproblem is approximately solved, the central node takes an average/add step according to the local update from each machine (Jaggi et al., 2014, Ma et al., 2015, Yang, 2013). Our proposed method falls into this category.

There are two limitations of the *coordinate*-type approach. One limitation is that most existing coordinate-type methods fail to match the communication complexity lower bounds proved in (Arjevani and Shamir, 2015). For example, when ϕ_i is $(1/\gamma)$-smooth, g is λ-strongly convex, and $R = \max_i \|a_i\|_2$ in problem (1), then the *condition number* is defined as

$$\kappa = R^2/(\lambda\gamma), \quad (3)$$

and the lower bound of communication complexity obtained in (Arjevani and Shamir, 2015) is $\tilde{\mathcal{O}}\left(\sqrt{\kappa}\log(1/\epsilon)\right)$ in order to achieve an ϵ-suboptimal solution (Nesterov, 2013). Since these *coordinate*-type methods (Jaggi et al., 2014, Ma et al., 2015, Yang, 2013) could be seen as a distributed version of SDCA, and the iteration complexity of SDCA is $\tilde{\mathcal{O}}\left(\kappa\log(1/\epsilon)\right)$, the communication complexity of these methods (Jaggi et al., 2014, Ma et al., 2015, Yang, 2013) is $\tilde{\mathcal{O}}\left(\kappa\log(1/\epsilon)\right)$, which does not match the optimal communication complexity. The problem could get worse when problem (1) is ill-conditioned, i.e., $\kappa \gg 1$. The other limitation is that it is not straight-forward to extend existing coordinate-type approaches (Jaggi et al., 2014, Ma et al., 2015, Yang, 2013) to deal with different regularization forms of the regularization term $g(\cdot)$ other than ℓ_2 regularization.

1.2 OUR APPROACH

To overcome the limitations mentioned above, we propose a novel communication-efficient distributed framework with a primal-dual algorithm for saddle point problems (Chambolle and Pock, 2011, Zhang and Xiao, 2015b). The reasons that we focus on developing distributed framework for saddle point problems are three folds:

1. It has been shown that saddle point algorithms are able to obtain comparable and even more stable performance than other state-of-the-art techniques for convex optimization (Chambolle and Pock, 2011, Yu et al., 2015, Zhang and Xiao, 2015b).

2. Primal-dual coordinate algorithms for saddle point problems are able to reach the optimal iteration complexity as proved by Zhang and Xiao (2015b).

3. Since primal-dual algorithms keep both primal and dual variables in optimization, they are able to deal with different kinds of regularization terms $g(\cdot)$, such as ℓ_1-regularization, naturally.

Therefore, we aim to develop a distributed primal-dual algorithm which can inherit the above three properties in a distributed environment. To be specific, we first reformulate problem (1) as a convex-concave saddle point problem through convex conjugation, which keeps both primal and dual variables in optimization. We then decompose the saddle point problem into carefully designed local subproblems, which can be solved independently by each machine with respect to its local data. After the local subproblems are solved approximately, the updated local dual variables of each machine are sent to a central node. Based on all the updates of local dual variables, we construct a central problem and get the central update of the primal variables on the central node. Finally, the central node sends the updated primal variables and aggregated dual variables to each local machine for next local optimization iteration. Details will be described in Section 3.

As will be discussed in Section 4 and Section 5, by carefully designing the local subproblems and the central problem, our algorithm is able to reach the communication complexity lower bounds, and deal with different kinds of regularization terms. As the parameter λ of the regularization function $g(\cdot)$ is usually on the order of $1/\sqrt{n}$ or $1/n$ for many machine learning problems, we are especially interested in solving problem (1) under the distributed setting when it is ill-conditioned. With an extremely large-scale dataset, the *condition number* in (3) could be relatively large ($\kappa \gg 1$). We will

show through experiments in Section 6 that our proposed algorithm can obtain better and more stable performance on ill-conditioned problems compared to other communication-efficient distributed optimization methods. Moreover, we provide a solution to extend our algorithm to deal with non-smooth and non-strongly convex optimization problems in Section 5.

1.3 OTHER RELATED WORK

Besides the communication-efficient distributed optimization methods reviewed above, there exist other parallel and distributed optimization techniques. For example, a well-known approach for solving problem (1) is to perform a gradient descent method implemented in a distributed system. Each local machine computes its local gradient and sends it to the central node. The central node aggregates the local gradient to take a gradient step and updates $\mathbf{x}$, and then broadcasts it back to each local machine for the next iteration updates. If the accelerated gradient descent method (Nesterov, 2013) is used, one can obtain the iteration complexity $\tilde{\mathcal{O}}\left(\sqrt{\kappa}\log(1/\epsilon)\right)$. Another popular technique is distributed alternating direction method of multipliers (ADMM) (Boyd et al., 2011, Shi et al., 2014), whose complexity is $\tilde{\mathcal{O}}\left(\sqrt{\kappa}\log(1/\epsilon)\right)$ under certain conditions. Recently, Zhang and Xiao (2015a) proposed a distributed algorithm based on the inexact damped Newton method, which matches the communication complexity lower bound $\tilde{\mathcal{O}}\left(\sqrt{\kappa}\log(1/\epsilon)\right)$. For more related distributed and parallel optimization methods, we refer the readers to (Arjevani and Shamir, 2015, Chang et al., 2014, Duchi et al., 2013, Zhang et al., 2012) for a more comprehensive literature of the related work.

2 PRELIMINARIES

In this section, we introduce first-order primal-dual algorithms for convex-concave saddle point problems in a single machine. To begin with, we reformulate the primal problem (1) as a convex-concave saddle point problem through convex conjugation, and the saddle point reformulation is widely applied in machine learning (Dai et al., 2016, Zhang and Xiao, 2015b). Based on the definition of convex conjugate, we replace each component function $\phi_i(a_i^\top \mathbf{x})$ by

$$\phi_i(a_i^\top \mathbf{x}) = \sup_{y_i \in \mathbb{R}} \left\{ y_i \langle a_i, \mathbf{x}\rangle - \phi_i^*(y_i) \right\},$$

where $\phi_i^*(y_i) = \sup_{\alpha\in\mathbb{R}}\{\alpha y_i - \phi_i(\alpha)\}$ is the convex conjugate of ϕ_i, then we arrive at a convex-concave saddle point problem

$$\min_{\mathbf{x}\in\mathbb{R}^d} \max_{\mathbf{y}\in\mathbb{R}^n} f(\mathbf{x},\mathbf{y}), \tag{4}$$

where

$$f(\mathbf{x},\mathbf{y}) \stackrel{\text{def}}{=} \frac{1}{n}\sum_{i=1}^n \left(y_i\langle a_i,\mathbf{x}\rangle - \phi_i^*(y_i)\right) + g(\mathbf{x}),$$

where $\mathbf{y}\in\mathbb{R}^n$ is referred to as a vector of dual variables with its i-th element denoted by y_i, and each y_i is associated with a data point a_i. Under the assumption that ϕ_i is $(1/\gamma)$-smooth, g is λ-strongly convex, and each ϕ_i^* is γ-strongly convex, the saddle point problem (4) has a unique solution, which is denoted by $(\mathbf{x}^\star,\mathbf{y}^\star)$.

Based on the reformulated problem (4), we could rewrite the optimization problem as

$$\min_{\mathbf{x}\in\mathbb{R}^d}\max_{\mathbf{y}\in\mathbb{R}^n} f(\mathbf{x},\mathbf{y}) = \frac{1}{n}\langle A\mathbf{y},\mathbf{x}\rangle - \Phi^*(\mathbf{y}) + g(\mathbf{x}),$$

where $A = [a_1,...,a_n]$ and $\Phi^*(\mathbf{y}) = \frac{1}{n}\sum_{i=1}^n \phi_i^*(y_i)$. This is the generic saddle point problem (Chambolle and Pock, 2011). To solve this optimization problem, the basic idea is to alternatingly maximize f with repect to $\mathbf{y}$, and minimize f with respect to $\mathbf{x}$, which is

$$\mathbf{y}^{(t+1)} = \arg\max_{\mathbf{y}\in\mathbb{R}^n} \frac{1}{n}\langle A\mathbf{y},\bar{\mathbf{x}}^{(t)}\rangle - \Phi^*(\mathbf{y}) - \frac{\|\mathbf{y}-\mathbf{y}^{(t)}\|_2^2}{2\sigma n},$$

$$\mathbf{x}^{(t+1)} = \arg\min_{\mathbf{x}\in\mathbb{R}^d} \frac{1}{n}\langle A\mathbf{y}^{(t+1)},\mathbf{x}\rangle + g(\mathbf{x}) + \frac{\|\mathbf{x}-\mathbf{x}^{(t)}\|_2^2}{2\tau},$$

$$\bar{\mathbf{x}}^{(t+1)} = \mathbf{x}^{(t+1)} + \theta\left(\mathbf{x}^{(t+1)} - \mathbf{x}^{(t)}\right),$$

where the parameters τ and σ control the quadratic regularization terms with respect to $\mathbf{x}$ and $\mathbf{y}$, respectively, which is similar to the use of step size in primal methods. And $\bar{\mathbf{x}}^{(t+1)}$ is the extrapolation from $\mathbf{x}^{(t)}$ and $\mathbf{x}^{(t+1)}$ with parameter $\theta\in[0,1]$. Here, $\theta\left(\mathbf{x}^{(t+1)}-\mathbf{x}^{(t)}\right)$ is similar to the momentum term in Nesterov acceleration (Nesterov, 2013). There have been various primal-dual coordinate algorithms designed based on the above scheme, such as (Chambolle and Pock, 2011, Yu et al., 2015, Zhang and Xiao, 2015b). Among them, Yu et al. (2015), Zhang and Xiao (2015b) developed stochastic versions of the primal-dual algorithm for the saddle point problem. Suppose ϕ_i is $(1/\gamma)$-smooth and g is λ-strongly convex, most existing algorithms (Chambolle and Pock, 2011, Yu et al., 2015, Zhang and Xiao, 2015b) can achieve linear convergence rate to obtain an ϵ-accurate solution. The (expected) iteration complexity of these methods is $\tilde{\mathcal{O}}\left(\sqrt{\kappa}\log(1/\epsilon)\right)$, which is desired on ill-conditioned problems.

3 DISTRIBUTED SADDLE-POINT FRAMEWORK

In this section, we present our proposed communication-efficient **Di**stributed **S**addle **P**oint **A**lgorithm (DiSPA) in

detail. Assume that the dataset $\{a_i\}_{i=1}^n$ is distributed over K local machines, each machine k has access to a subset $\mathcal{P}_k$ of the data with size of n_k. Here we define K local vectors of $\mathbf{y}$ by using the notation $\mathbf{y}_{[k]} \in \mathbb{R}^n$ for $k = 1, ..., K$:

$$(\mathbf{y}_{[k]})_i = \begin{cases} y_i, & \text{if } i \in \mathcal{P}_k, \\ 0, & \text{otherwise,} \end{cases}$$

and K local data matrix of A by using the notation $A_{[k]} \in \mathbb{R}^{d\times n}$ for $k = 1, ..., K$:

$$(A_{[k]})_i = \begin{cases} a_i, & \text{if } i \in \mathcal{P}_k, \\ \mathbf{0}, & \text{otherwise,} \end{cases}$$

where $(A_{[k]})_i$ denotes the i-th column of $A_{[k]}$, and $\mathbf{0}$ is a d-dimensional vector of all zeros. According to the data partition, we decompose the problem (4) into K local subproblems, and define the associated central problem. The core idea is to solve the defined **local subproblems** on each local machine independently, and then centralize the updated information of each machine to solve a easy **central problem**.

3.1 LOCAL SUBPROBLEMS

We define a local subproblem of the original saddle point problem (4) for machine k, which only requires accessing data that is available locally, i.e., $\mathcal{P}_k$. Specifically, each local subproblem of machine k at t-iteration is defined as

$$\min_{\mathbf{x}\in\mathbb{R}^d} \max_{\mathbf{y}_{[k]}\in\mathbb{R}^n} \mathcal{L}_k^{(t)}(\mathbf{x}, \mathbf{y}_{[k]}), \tag{5}$$

where

$$\begin{aligned}\mathcal{L}_k^{(t)}(\mathbf{x}, \mathbf{y}_{[k]}) :=& \frac{1}{n}\sum_{j\in\mathcal{P}_k} \left(y_j\langle a_j, \mathbf{x}\rangle - \phi_j^*(y_j)\right) + g(\mathbf{x}) \\ &+ \frac{1}{n}\sum_{i\notin\mathcal{P}_k} y_i^{(t-1)}\langle a_i, \mathbf{x}\rangle + r_k^{(t)}(\mathbf{x}, \mathbf{y}_{[k]}),\end{aligned}$$

and

$$r_k^{(t)}(\mathbf{x}, \mathbf{y}_{[k]}) = \frac{\|\mathbf{x} - \mathbf{x}^{(t-1)}\|_2^2}{2\tau} - \frac{K\|\mathbf{y}_{[k]} - \mathbf{y}_{[k]}^{(t-1)}\|_2^2}{2\sigma n}.$$

Each local subproblem is related to the solution obtained in the previous iteration $\left(\mathbf{x}^{(t-1)}, \mathbf{y}^{(t-1)}\right)$, and we will define the parameters τ, σ later. The first two terms can be regarded as a local version of the original saddle point problem (4). The third term $\frac{1}{n}\sum_{i\notin\mathcal{P}_k} y_i^{(t-1)}\langle a_i, \mathbf{x}\rangle = \frac{1}{n}\langle\sum_{l\neq k} A_{[l]}\mathbf{y}_{[l]}^{(t-1)}, \mathbf{x}\rangle$ in (5) can be interpreted as the interaction with other local subproblems. And the quadratic regularization term $r_k^{(t)}(\mathbf{x}, \mathbf{y}_{[k]})$ is used to enforce $\mathbf{x}$ and $\mathbf{y}_{[k]}$ not to move too far away from solution obtained at the previous iteration.

To solve the local subproblem $\mathcal{L}_k^{(t)}(\mathbf{x}, \mathbf{y}_{[k]})$, we could apply primal-dual coordinate type of methods (Yu et al., 2015, Zhang and Xiao, 2015b). The input needed for solving the local subproblem would be $\mathbf{x}^{(t-1)}$ and $A\mathbf{y}^{(t-1)}$, as the term $\sum_{l\neq k} A_{[l]}\mathbf{y}_{[l]}^{(t-1)}$ can be expressed as $\left(A\mathbf{y}^{(t-1)} - A_{[k]}\mathbf{y}_{[k]}^{(t-1)}\right)$ and machine k has access to $\left(A_{[k]}, \mathbf{y}_{[k]}^{(t-1)}\right)$. By denoting our local optimization algorithm used as 'Local-DiSPA', we could denote the procedure of solving the local subproblem as $\left(\mathbf{x}_k^{(t)}, \mathbf{y}_{[k]}^{(t)}\right) \leftarrow$ Local-DiSPA$(\mathbf{x}^{(t-1)}, A\mathbf{y}^{(t-1)})$. Note that in our framework, each local subproblem is only required to be solved approximately.

3.2 CENTRAL PROBLEM

Let $\left(\mathbf{x}_k^{(t)}, \mathbf{y}_{[k]}^{(t)}\right)$ denote the approximate solution obtained by solving $\mathcal{L}_k^{(t)}(\mathbf{x}, \mathbf{y}_{[k]})$ on machine k. Let $\mathbf{y}^{(t)} = \sum_{k=1}^K \mathbf{y}_{[k]}^{(t)}$, we define a central problem at t-iteration as

$$\mathbf{x}^{(t)} = \arg\min_{\mathbf{x}\in\mathbb{R}^d} \mathcal{C}^{(t)}(\mathbf{x}), \tag{6}$$

where

$$\mathcal{C}^{(t)}(\mathbf{x}) := \frac{1}{n}\sum_{i=1}^n y_i^{(t)}\langle a_i, \mathbf{x}\rangle + g(\mathbf{x}) + \frac{\|\mathbf{x} - \mathbf{x}^{(t-1)}\|_2^2}{2\tau}.$$

The central update involves two steps. Firstly, updated local $A_{[k]}\mathbf{y}_{[k]}^{(t)} \in \mathbb{R}^d$ are sent to the central node. After that, we solve (6) on the central node, and then send the updated $\mathbf{x}^{(t)} \in \mathbb{R}^d$ and $A\mathbf{y}^{(t)} \in \mathbb{R}^d$ of the t-iteration back to each local machine. The central update could be viewed as minimizing the original saddle point problem (4) over $\mathbf{x}$ with updated $\mathbf{y}^{(t)}$. Also, we should observe that the parameter τ defined in (5) and (6) is the same parameter, which is one of the key points of our framework.

3.3 OVERALL ALGORITHM DESCRIPTION

The overall algorithm of DiSPA is presented in Algorithm 1. Firstly, we distribute the initial dual variable $\mathbf{y}^{(0)}$ to K local machines, and aggregate $A_{[k]}\mathbf{y}_{[k]}^{(0)}$ from all machines to compute $A\mathbf{y}^{(0)} \leftarrow \sum_{k=1}^K A_{[k]}\mathbf{y}_{[k]}^{(0)}$. After that, the initial primal variables $\mathbf{x}^{(0)}$ and the transformed dual variables $A\mathbf{y}^{(0)}$ are sent to each machine (Steps 2-4). Secondly, each machine performs optimization on its own local subproblem $\mathcal{L}_k^{(t)}(\mathbf{x}, \mathbf{y}_{[k]})$, using a primal-dual algorithm, such as SPDC (Zhang and Xiao, 2015b), and sends its local update $A_{[k]}\mathbf{y}_{[k]}^{(t)} \in \mathbb{R}^d$ to the central

Algorithm 1 DiSPA(f, $\mathbf{x}^{(0)}$, $\mathbf{y}^{(0)}$, τ, σ, T)

1: **Input:** Data points $\{a_i\}_{i=1}^n$ distributed across K machines $\{\mathcal{P}_k\}_{k=1}^K$, parameters $\tau, \sigma \in \mathbb{R}$, initial primal variables $\mathbf{x}^{(0)} \in \mathbb{R}^d$ and initial dual variables $\mathbf{y}^{(0)} \in \mathbb{R}^n$.
2: Distribute $\mathbf{y}^{(0)}$ to K machines as $\mathbf{y}_{[k]}^{(0)}$ for each.
3: Each machine compute $A_{[k]}\mathbf{y}_{[k]}^{(0)}$, and send it back to the central node.
4: Perform $A\mathbf{y}^{(0)} \leftarrow \sum_{k=1}^K A_{[k]}\mathbf{y}_{[k]}^{(0)}$ on the central node, and send $\mathbf{x}^{(0)}$ and $A\mathbf{y}^{(0)}$ to each machine
5: **for** $t = 1, 2, ..., T$ **do**
6: **for** $k \in [K]$ in parallel over all machines **do**
7: $\left(\mathbf{x}_k^{(t)}, \mathbf{y}_{[k]}^{(t)}\right) \leftarrow$ Local-DiSPA$(\mathbf{x}^{(t-1)}, A\mathbf{y}^{(t-1)})$
8: Send $A_{[k]}\mathbf{y}_{[k]}^{(t)}$ to the central node
9: **end for**
10: $A\mathbf{y}^{(t)} \leftarrow \sum_{k=1}^K A_{[k]}\mathbf{y}_{[k]}^{(t)}$ (on the central node)
11: $\mathbf{x}^{(t)} \leftarrow \arg\min_{\mathbf{x}\in\mathbb{R}^d} \mathcal{C}^{(t)}(\mathbf{x})$ (on the central node)
12: Send $\mathbf{x}^{(t)}$ and $A\mathbf{y}^{(t)}$ back to each machine
13: **end for**
14: **Output:** $(\mathbf{x}^{(T)}, \mathbf{y}^{(T)})$

node. The goal of this internal procedure is to approximately solve each local subproblem. We refer this procedure as the inner loop, which is stated in Steps 6-9 of Algorithm 1. Thirdly, with the local updates $A_{[k]}\mathbf{y}_{[k]}^{(t)}$'s, the central node aggregates them to compute $A\mathbf{y}^{(t)}$, and solves the defined central problem $\mathcal{C}^{(t)}(\mathbf{x})$ to get the update of $\mathbf{x}^{(t)}$ (Steps 10-11 of Algorithm 1). Finally, the central node sends ($\mathbf{x}^{(t)}$ and $A\mathbf{y}^{(t)}$) back to each machine in order to start the next round of local optimization (Step 12), we refer Steps 5-13 as the outer loop of our algorithm.

Note that, on one hand, the essential information that the central problem requires can be represented by the vectors $\left\{A_{[k]}\mathbf{y}_{[k]}^{(t)}\right\}$'s of each local machine, which are of dimension of d. On the other hand, each local machine only requires $\left(\mathbf{x}^{(t)}, A\mathbf{y}^{(t)}\right)$, which is of $2d$ dimensions, from the central node.

4 CONVERGENCE ANALYSIS

Before we present our main convergence results, we make some assumptions on the objective functions and the local suboptimality on each local subproblem.

Assumption 1. *Each $\phi_i(\cdot)$ is convex and differentiable, and $\phi_i(\cdot)$ is $(1/\gamma)$-smooth, i.e.,*

$$|\phi_i'(a) - \phi_i'(b)| \leq (1/\gamma)|a - b|, \quad \forall a, b \in \mathbb{R}.$$

Assumption 2. *$g(\cdot)$ is λ-strongly convex i.e. for $\forall x, y \in \mathbb{R}^d$ and $g'(y) \in \partial g(y)$,*

$$g(x) \geq g(y) + g'(y)(x - y) + \frac{\lambda}{2}\|x - y\|_2^2.$$

Assumption 3. *There exists constants $\Omega_{\mathbf{x}}, \Omega_{\mathbf{y}} > 0$ such that*

$$\max_t \|\mathbf{x}^{(t)} - \mathbf{x}^\star\|_2 \leq \Omega_{\mathbf{x}},$$
$$\max_t \|\mathbf{y}^{(t)} - \mathbf{y}^\star\|_2 \leq \Omega_{\mathbf{y}},$$

where $(\mathbf{x}^{(t)}, \mathbf{y}^{(t)})$ is t-iteration update of Algorithm 1, and $(\mathbf{x}^\star, \mathbf{y}^\star)$ is the optimal solution of (4).

Assumption 4. *Each local machine k produces an approximate solution $\left(\mathbf{x}_k^{(t)}, \mathbf{y}_{[k]}^{(t)}\right)$ that satisfies*

$$\mathbb{E}\left[\left\|\mathbf{y}_{[k]}^{(t)} - \hat{\mathbf{y}}_{[k]}^{(t)}\right\|_2\right] \leq \hat{\Theta}\left\|\mathbf{y}_{[k]}^{(t-1)} - \hat{\mathbf{y}}_{[k]}^{(t)}\right\|_2,$$

and

$$\mathbb{E}[(f(\hat{\mathbf{x}}_k^{(t)}, \hat{\mathbf{y}}_{[k]}^{(t)}) - f(\hat{\mathbf{x}}_k^{(t)}, \mathbf{y}_{[k]}^{(t)}))] \leq \tilde{\Theta}\left(f(\hat{\mathbf{x}}_k^{(t)}, \hat{\mathbf{y}}_{[k]}^{(t)}) - f(\hat{\mathbf{x}}_k^{(t)}, \mathbf{y}_{[k]}^{(t-1)})\right),$$

where $\hat{\Theta}, \tilde{\Theta} < 1$, and $\left(\hat{\mathbf{x}}_k^{(t)}, \hat{\mathbf{y}}_{[k]}^{(t)}\right)$ is the optimal solution of local subproblem $\mathcal{L}_k^{(t)}\left(\mathbf{x}, \mathbf{y}_{[k]}\right)$.

Assumption 5. *There exist constants c', $c'' > 0$, and the approximate solution $\mathbf{y}^{(t)} = \sum_{k=1}^K \mathbf{y}_{[k]}^{(t)}$ satisfies*

$$\mathbb{E}[\blacktriangle] \leq \Theta(f(\mathbf{x}^{(t-1)}, \mathbf{y}^\star) - f(\mathbf{x}^\star, \mathbf{y}^{(t-1)})),$$

and $\blacktriangle$ is defined as

$$\blacktriangle = \sum_{k=1}^K (f(\hat{\mathbf{x}}_k^{(t)}, \hat{\mathbf{y}}_{[k]}^{(t)}) - f(\hat{\mathbf{x}}_k^{(t)}, \mathbf{y}_{[k]}^{(t)})) + \frac{M}{n}\|\mathbf{y}^{(t)} - \hat{\mathbf{y}}^{(t)}\|_2,$$

where $M = c'\Omega_{\mathbf{x}} + c''\Omega_{\mathbf{y}}$, and $\Theta \in (0, 1)$ is a pre-defined constant.

Note that Assumptions 4 and 5 imply that the local subproblems need to be solved approximately to some extent. As the local subproblems are saddle point problems, we can apply different kinds of algorithms for saddle point problems to solve them, such as (Chambolle and Pock, 2011, Yu et al., 2015, Zhang and Xiao, 2015b), which include both stochastic and non-stochastic optimization methods.

To analyse the convergence behavior of our distributed algorithm, we need to characterize the connection between local subproblems and central update on the central node. Based on the relationship between the central update $\mathbf{x}^{(t)}$ and the local optimal solution $\left(\hat{\mathbf{x}}_k^{(t)}, \hat{\mathbf{y}}_{[k]}^{(t)}\right)$, we can get the convergence guarantee of Algorithm 1. All proofs can be found in Appendix.

Lemma 1. *Suppose that Assumptions 1-5 hold. Let* $(\mathbf{x}^\star, \mathbf{y}^\star)$ *be the unique saddle point of* $f(\mathbf{x}, \mathbf{y})$ *defined in (4), and define*

$$\Delta^{(t)} = f(\mathbf{x}^{(t)}, \mathbf{y}^\star) - f(\mathbf{x}^\star, \mathbf{y}^{(t)}) + \frac{1+\tau\lambda}{2\tau}\|\mathbf{x}^{(t)} - \mathbf{x}^\star\|_2^2 + \frac{2K^2 + 2K\sigma\gamma - 1}{4\sigma n K}\|\mathbf{y}^{(t)} - \mathbf{y}^\star\|_2^2.$$

If the parameters τ *and* σ *are chosen such that* $\tau\sigma = \frac{1}{4R^2}$ *and* $\sigma > \frac{1}{2K\gamma}$*, then for* $t \geq 1$*, the proposed DiSPA (Algorithm 1) achieves*

$$\mathbb{E}[\Delta^{(t)}] \leq \Theta\mathbb{E}[\Delta^{(t-1)}], \tag{7}$$

where $\Theta = \max\{\frac{1}{1+(\frac{\sigma\gamma}{K} - \frac{1}{2K^2})}, \frac{1}{1+\tau\lambda}\}$.

Theorem 1. *Suppose that Assumptions 1-5 hold, and the parameters* $\tau = \gamma/(4R^2)$*,* $\sigma = 1/\gamma$ *and* $K \leq \kappa$*. In order for Algorithm 1 to obtain*

$$\mathbb{E}[\|\mathbf{x}^{(T)} - \mathbf{x}^\star\|_2^2] \leq \epsilon,$$

it suffices to have the number of communication iterations T *satisfy*

$$T \geq \left(1 + \frac{4R^2}{\lambda\gamma}\right)\log\left(\frac{C}{\epsilon}\right),$$

where $C = \frac{\Delta^{(0)}}{\frac{\lambda}{2}\left(1+\frac{4R^2}{\lambda\gamma}\right)}$.

Based on Lemma 1 and Theorem 1, we can get the communication complexity of Algorithm 1. By taking $\tau = \frac{\gamma}{4R^2}$ and $\sigma = \frac{1}{\gamma}$, the communication complexity of Algorithm 1 is $\tilde{\mathcal{O}}\left(\kappa\log(1/\epsilon)\right)$. This does not match the optimal communication complexity. In the next section, we derive an accelerated version of DiSPA that attains the optimal communication complexity.

5 EXTENSION OF DISPA

In this section, we derive two extensions of DiSPA. The first extension applies a generic acceleration scheme proposed by Lin et al. (2015) to obtain an accelerated version of DiSPA. The second one extends our algorithm to handle non-smooth and non-strongly convex loss functions with convergence guarantee.

5.1 ACCELERATION

The communication complexity of our algorithm derived in Theorem 1 is $\tilde{\mathcal{O}}\left(\kappa\log(1/\epsilon)\right)$ in order to achieve an ϵ-suboptimal solution, which does not match the lower bound proved in (Arjevani and Shamir, 2015). Based on the 'Catalyst' acceleration scheme proposed by Lin et al. (2015), we develop the accelerated DiSPA to achieve the optimal communication complexity. 'Catalyst' is a generic scheme for accelerating first-order optimization methods, similar to classical gradient descent schemes of Nesterov acceleration. It is a natural choice for distributed optimization algorithms that solve local subproblem approximately in each communication iteration. Based on Catalyst acceleration, we modify the original objective function by adding a quadratic term

$$f^{(t)}(\mathbf{x}, \mathbf{y}) = f(\mathbf{x}, \mathbf{y}) + \frac{\vartheta}{2}\|\mathbf{x} - \mathbf{z}^{(t-1)}\|_2^2,$$

where $\vartheta > 0$ is a parameter defined in Catalyst, $\mathbf{z}^{(t)}$ is obtained by an extrapolation step similar to the acceleration scheme in (Nesterov, 2013). With a carefully selected parameter ϑ, we can develop an accelerated version of DiSPA described in Algorithm 2, which achieves the optimal communication complexity. We refer to the accelerated distributed saddle point algorithm as A-DiSPA.

Theorem 2. *Suppose Assumption 1-5 hold and assume* $\vartheta = \frac{4R^2}{\gamma} - \lambda > 0$*, if the parameters in Catalyst are chosen as* $T_s = \tilde{O}(1)$*,* $q = \lambda/(\lambda + \vartheta)$*,* $\alpha_0 = \sqrt{q}$*. The total communication iterations of Algorithm 2 for achieving* $\mathcal{P}(\mathbf{x}^{(T)}) - \mathcal{P}(\mathbf{x}^\star) < \epsilon$ *is* $\tilde{O}\left(\sqrt{\kappa}\log\left(\frac{1}{\epsilon}\right)\right)$.

The 'Catalyst' acceleration scheme is applied widely to accelerate different kinds of optimization methods (Reddi et al., 2016). As the 'acceleration' quadratic term is related to the primal variable $\mathbf{x}$ and the saddle point problem keeps the primal variable, it is convenient to apply this acceleration scheme to DiSPA and achieve the optimal communication complexity.

5.2 NON-SMOOTH OR/AND NON-STRONGLY CONVEX FUNCTIONS

The communication complexity bounds in Theorem 1 and Theorem 2 are developed under the Assumptions 1 and 2, which means that the derivative of $\phi_i(\cdot)$ needs to be $(1/\gamma)$-Lipschitz continuous and $g(\cdot)$ needs to be λ-strongly convex. For general loss functions in machine learning, both assumptions may fail, for example, when $\phi_i(\cdot)$ is hinge loss and $g(\cdot)$ is ℓ_1-regularization term. Therefore, we aim to develop an extension of Algorithm 2 to deal with non-smooth and non-strongly convex loss functions.

To be concise, we only consider the case when $\phi_i(\cdot)$ is non-smooth and $g(\cdot)$ is non-strongly convex. The key idea here is to apply a perturbation term to analyze the non-smooth and non-strongly convex setting, which is commonly used in many optimization methods (Reddi et al., 2016, Zhang and Xiao, 2015b). Here we assume that neither $\phi_i(\cdot)$ is smooth nor $g(\cdot)$ is strongly convex.

Algorithm 2 A-DiSPA

Input: Data points $\{a_i\}_{i=1}^n$ distributed across K machines $\{\mathcal{P}_k\}_{k=1}^K$, parameters $\tau, \sigma, \vartheta, \alpha_0 \in \mathbb{R}$, initial primal variable $\mathbf{x}^{(0)}$ generated on the central node, initial local dual variables $\mathbf{y}^{(0)}$ generated on local machines, optimization algorithm DiSPA, number of inner iterations T_s of DiSPA.
Initialization: $q = \lambda/(\lambda + \vartheta)$, $\mathbf{z}^{(0)} = \mathbf{x}^{(0)}$, $t = 1$
while stopping criterion is not satisfied **do**
  $(\mathbf{x}^{(t)}, \mathbf{y}^{(t)}) = \text{DiSPA}(f^{(t)}, \mathbf{x}^{(t-1)}, \mathbf{y}^{(t-1)}, \tau, \sigma, T_s)$
  Compute $\alpha_t \in (0, 1)$ from equation

$$\alpha_t^2 = (1 - \alpha_t)\alpha_{t-1}^2 + q\alpha_t$$

  Compute $\mathbf{z}^{(t)} = \mathbf{x}^{(t)} + \beta_t(\mathbf{x}^{(t)} - \mathbf{x}^{(t-1)})$, where $\beta_t = \frac{\alpha_{t-1}(1-\alpha_{t-1})}{\alpha_{k-1}^2+\alpha_t}$
  $t = t + 1$
end while
Output $\mathbf{x}^{(t)}$

In particular, we consider the following modified saddle point function

$$f_\epsilon(\mathbf{x}, \mathbf{y}) \stackrel{\text{def}}{=} \frac{1}{n}\sum_{i=1}^n \left(y_i\langle a_i, \mathbf{x}\rangle - \left(\phi_i^*(y_i) + \frac{\epsilon}{2}y_i^2\right)\right) + \left(g(\mathbf{x}) + \frac{\epsilon}{2}\|\mathbf{x}\|_2^2\right), \quad (8)$$

where $\epsilon > 0$ is a pre-defined scalar. $f_\epsilon(\mathbf{x}, \mathbf{y})$ can be regarded as an approximation of the original saddle point problem $f(\mathbf{x}, \mathbf{y})$. We then obtain that both $\phi_i^*(y_i) + \frac{\epsilon}{2}y_i^2$ and $g(\mathbf{x}) + \frac{\epsilon}{2}\|\mathbf{x}\|_2^2$ are ϵ-strongly convex. In the following corollary, we show that the perturbed function $f_\epsilon(\mathbf{x}, \mathbf{y})$ is a good approximation of the origin function $f(\mathbf{x}, \mathbf{y})$.

Corollary 1. *Assume that ϕ_i is convex and L_ϕ-Lipschitz continuous, and g is convex. Let $(\mathbf{x}^\star, \mathbf{y}^\star)$ be the unique saddle point of $f(\mathbf{x}, \mathbf{y})$, and $(\mathbf{x}_\epsilon^\star, \mathbf{y}_\epsilon^\star)$ be the unique saddle point of $f_\epsilon(\mathbf{x}, \mathbf{y})$. Then we have*

$$\mathcal{P}(\mathbf{x}_\epsilon^\star) - \mathcal{P}(\mathbf{x}^\star) \leq \frac{\epsilon}{2}\left(\|\mathbf{x}^\star\|_2^2 + L_\phi^2\right).$$

As a result, we could apply Algorithm 2 to the perturbed function $f_\epsilon(\mathbf{x}, \mathbf{y})$. By applying Theorem 2 and the relationship between $f_\epsilon(\mathbf{x}, \mathbf{y})$ and $f(\mathbf{x}, \mathbf{y})$, we obtain that the communication complexity of Algorithm 2 for non-smooth and non-strongly convex functions is $\tilde{O}\left((1/\epsilon)\log\left(\frac{1}{\epsilon^2}\right)\right)$, and the proof is similar to the one in (Zhang and Xiao, 2015b).

6 NUMERICAL EXPERIMENTS

In this section, we conduct numerical experiments of DiSPA and A-DiSPA on several real-world distributed datasets. We compare the performance of our algorithms with the state-of-the-art distributed optimization algorithm CoCoA+ (Ma et al., 2015).

For our experiments, we solve the standard binary classification task with datasets obtained from LIBSVM datasets (Chang and Lin, 2011). The statistics of the datasets are summarized in Table 1. Our goal is to minimize the regularized empirical risk with smoothed hinge loss:

$$\min_{\mathbf{x}\in\mathbb{R}^d} \mathcal{P}(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^n \phi_i(a_i^\top \mathbf{x}) + \frac{\lambda}{2}\|\mathbf{x}\|_2^2.$$

For each task, the data point takes the form of (a_i, b_i), where a_i is a feature vector, $b_i \in \{1, -1\}$ is the corresponding class label and associated with loss function ϕ_i. The smoothed hinge loss function (Shalev-Shwartz and Zhang, 2013) ϕ_i is defined as

$$\phi_i(z) = \begin{cases} 0 & \text{if } b_i z \geq 1, \\ 1 - b_i z - \frac{\gamma}{2} & \text{if } b_i z \leq 1 - \gamma, \\ \frac{1}{2\gamma}(1 - b_i z)^2 & \text{otherwise,} \end{cases}$$

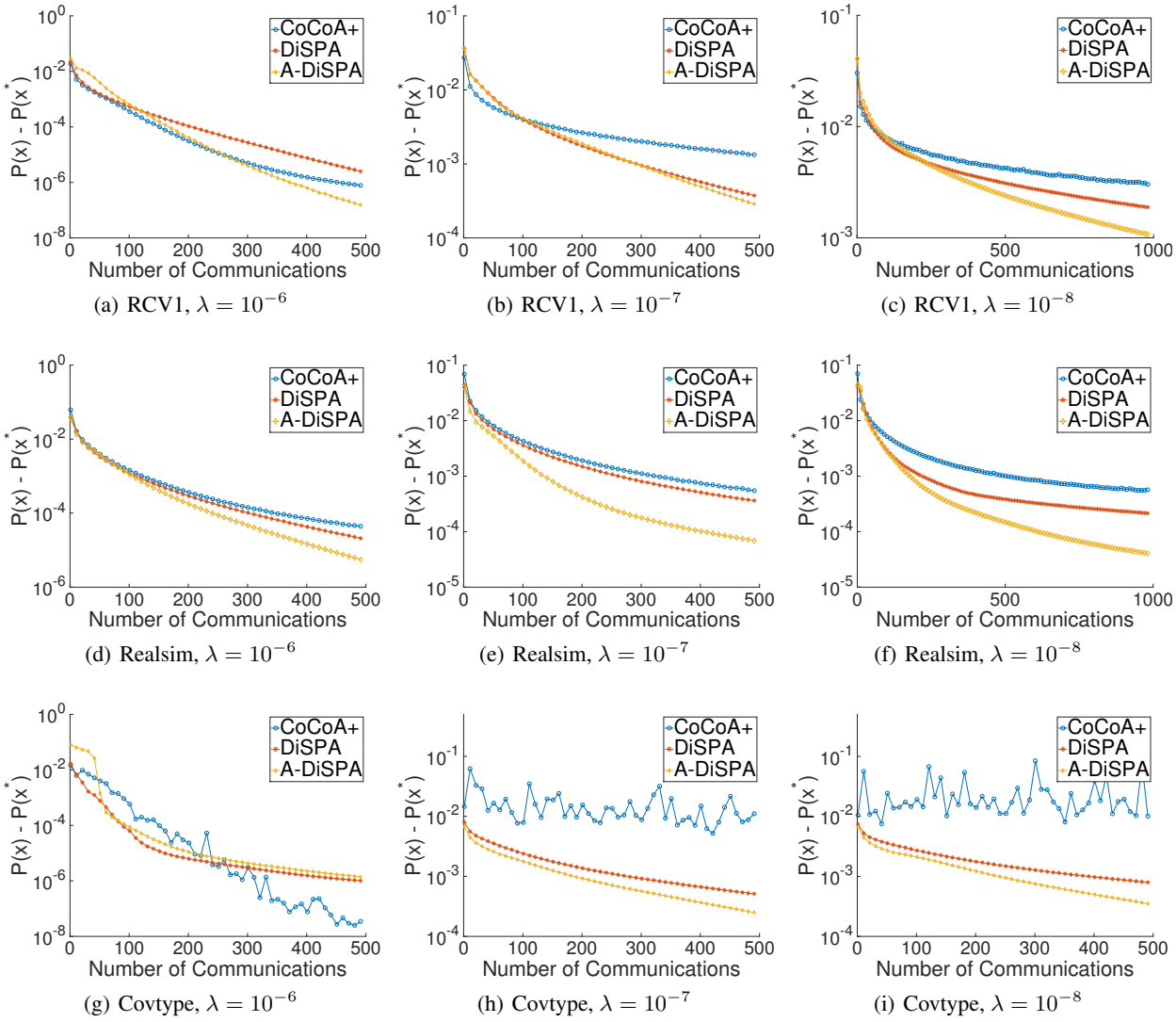
where ϕ_i is $(1/\gamma)$-smooth. We set $\gamma = 1$ in our experiments. The conjugate function of ϕ_i is $\phi_i^*(\beta) = b_i\beta + \frac{\gamma}{2}\beta^2$ for $b_i\beta \in [-1, 0]$ and $+\infty$ otherwise.

Table 1: Three Datasets for Numerical Experiments

DATASET	**# SAMPLES** n	**# FEATURES** d
RCV1	677,399	47,236
Realsim	72,309	20,958
Covtype	581,012	54

6.1 IMPLEMENTATION DETAILS

We implement DiSPA, A-DiSPA and CoCoA+ in Petuum (Xing et al., 2015). For DiSPA and A-DiSPA, we use SPDC (Zhang and Xiao, 2015b) as the local solver, and use SDCA (Shalev-Shwartz and Zhang, 2013) as the local solver for CoCoA+. When doing comparison among different algorithms, we run the same number of local iterations on local machines (i.e., iterations of SDCA or SPDC) before communication with the central node. We compare different algorithms based on communication iterations. From our results, we find that DiSPA converges faster when parameters τ and σ are large, which is similar to the findings reported in (Johnson and Zhang, 2013, Zhang and Xiao, 2015a). We adopt



(a) RCV1, $\lambda = 10^{-6}$ (b) RCV1, $\lambda = 10^{-7}$ (c) RCV1, $\lambda = 10^{-8}$

(d) Realsim, $\lambda = 10^{-6}$ (e) Realsim, $\lambda = 10^{-7}$ (f) Realsim, $\lambda = 10^{-8}$

(g) Covtype, $\lambda = 10^{-6}$ (h) Covtype, $\lambda = 10^{-7}$ (i) Covtype, $\lambda = 10^{-8}$

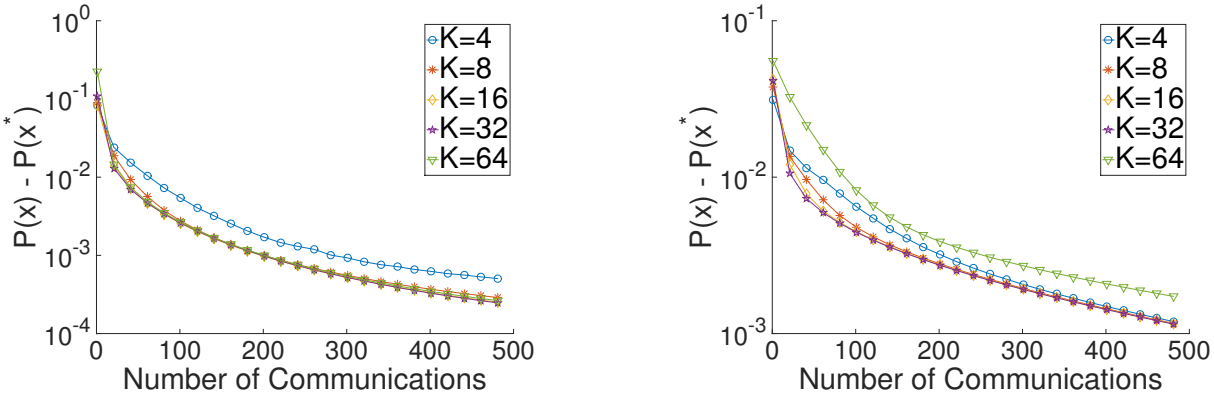
Figure 1: Comparison of DiSPA, A-DiSPA and CoCoA+ on three datasets: RCV1, Realsim, Covtype. The horizontal axis is the number of communications, and the vertical axis is the optimality gap $(P(\mathbf{x}^{(T)}) - P(\mathbf{x}^\star))$. Each plot contains a comparison of CoCoA+ (blue, circle), DiSPA (red, asterisk) and A-DiSPA (yellow, diamond). All the plots are shown on log-y scale . The number of local iterations performed in each communication iteration is 5,000 for Realsim, and 10,000 iterations for RCV1 and Covtype.

tuned parameters τ and σ from a predefined range. For CoCoA+, we present the results of the selected optimal parameter σ'.

6.2 COMPARISON BETWEEN DISPA AND COCOA+

We compare DiSPA and CoCoA+ on three datasets: RCV1, Realsim, Covtype, across different values of the regularization parameter λ. To be specific, the regularization parameter λ is set to be $\lambda \in \{10^{-6}, 10^{-7}, 10^{-8}\}$ on the three datasets. Both DiSPA and CoCoA+ are implemented on 16 machines ($K = 16$). For both DiSPA and CoCoA+, the numbers of local iterations performed in each communication iteration are the same.

In Figure 1, the results show that the performance of DiSPA is comparable with CoCoA+. We can see that when $\lambda = 10^{-6}$, DiSPA has comparable convergence performance with regard to CoCoA+. On all the three datasets, when $\lambda \in \{10^{-7}, 10^{-8}\}$, DiSPA converges to the optimal solution faster and more stable than CoCoA+. We also notice that the performance of Co-



(a) Realsim, $\lambda = 10^{-7}$

(b) RCV1, $\lambda = 10^{-7}$

Figure 2: The performance of DiSPA with increasing number of machines, $K \in \{4, 8, 16, 32, 64\}$. The local iterations performed in each communication iteration is 1,000 for Realsim, and 5,000 iterations for RCV1.

CoA+ on Covtype is not stable compared to that on the other two datasets, especially when λ is very small, i.e., $\lambda = 10^{-7}$ or $\lambda = 10^{-8}$, which is consistent with the experimental results of CoCoA+ on Covtype in Reddi et al. (2016). In Section 6.3, we will show the superior performance of A-DiSPA on ill-conditioned problems.

6.3 COMPARISON OF DISPA AND A-DISPA

In Figure 1, we also compare A-DiSPA and DiSPA to show the acceleration obtained by applying Catalyst acceleration. We compare DiSPA and A-DiSPA on three datasets (RCV1, Realsim, Covtype) across different regularization parameter under the same settings as mentioned in Section 6.2. The results show that when problem is ill-conditioned, i.e. $\kappa = R^2/(\lambda\gamma) \gg 1$, A-DiSPA converges substantially faster than DiSPA. This confirms our theoretical analysis, as the communication complexity of A-DiSPA is $\tilde{O}\left(\sqrt{\kappa}\log\left(\frac{1}{\epsilon}\right)\right)$ and DiSPA is $\tilde{O}\left(\kappa\log\left(\frac{1}{\epsilon}\right)\right)$.

6.4 SCALABILITY

Since scalability is an important metric of distributed optimization algorithms, in Figure 2, we study the scalability of DiSPA by observing the numerical performance with increasing number of machines. We conduct experiments on two datasets (RCV1, Realsim), and the machine number is set to be $K \in \{4, 8, 16, 32, 64\}$. For comparison, we keep the iterations and parameters τ, σ the same for each dataset. The experiments show that DiSPA can scale effectively with number of machines, which confirms our theory in Section 4. We observe that performance slightly drops on RCV1 with 64 machines, this is possibly because that local subproblems are solved with higher accuracy which may affect the effectiveness of aggregation in the central update.

7 CONCLUSION

In this work, we present a novel distributed optimization framework for solving the saddle point problem, which can be applied to solving a generic class of convex optimization problems. We provide the theoretical guarantee of our algorithms, and show that the accelerated algorithm can achieve the optimal communication complexity. We also extend our algorithm for solving non-smooth and non-strongly convex loss functions. The experimental results demonstrate that our algorithms obtain better performance compared to the state-of-the-art distributed optimization method.

Acknowledgements

This work is supported by NTU Singapore Nanyang Assistant Professorship (NAP) grant M4081532.020, Singapore MOE AcRF Tier-2 grant MOE2016-T2-2-060, and MOE AcRF Tier-1 grant 2016-T1-001-159.

References

Y. Arjevani and O. Shamir. Communication complexity of distributed convex learning and optimization. In *Advances in Neural Information Processing Systems*, pages 1756–1764, 2015.

S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers.

Foundations and Trends® in Machine Learning, 3(1): 1–122, 2011.

A. Chambolle and T. Pock. A first-order primal-dual algorithm for convex problems with applications to imaging. *Journal of Mathematical Imaging and Vision*, 40(1):120–145, 2011.

C.-C. Chang and C.-J. Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

T.-H. Chang, A. Nedic, and A. Scaglione. Distributed constrained optimization by consensus-based primal-dual perturbation method. *IEEE Transactions on Automatic Control*, 59(6):1524–1538, 2014.

B. Dai, N. He, Y. Pan, B. Boots, and L. Song. Learning from conditional distributions via dual kernel embeddings. *arXiv preprint arXiv:1607.04579*, 2016.

J. Duchi, M. I. Jordan, and B. McMahan. Estimation, optimization, and parallelism when data is sparse. In *Advances in Neural Information Processing Systems*, pages 2832–2840, 2013.

M. Jaggi, V. Smith, M. Takác, J. Terhorst, S. Krishnan, T. Hofmann, and M. I. Jordan. Communication-efficient distributed dual coordinate ascent. In *Advances in Neural Information Processing Systems*, pages 3068–3076, 2014.

R. Johnson and T. Zhang. Accelerating stochastic gradient descent using predictive variance reduction. In *Advances in Neural Information Processing Systems*, pages 315–323, 2013.

H. Lin, J. Mairal, and Z. Harchaoui. A universal catalyst for first-order optimization. In *Advances in Neural Information Processing Systems*, pages 3384–3392, 2015.

C. Ma, V. Smith, M. Jaggi, M. Jordan, P. Richtarik, and M. Takac. Adding vs. averaging in distributed primal-dual optimization. In *Proceedings of The 32nd International Conference on Machine Learning*, pages 1973–1982, 2015.

Y. Nesterov. *Introductory lectures on convex optimization: A basic course*, volume 87. Springer Science & Business Media, 2013.

S. J. Reddi, J. Konečnỳ, P. Richtárik, B. Póczós, and A. Smola. Aide: Fast and communication efficient distributed optimization. *arXiv preprint arXiv:1608.06879*, 2016.

S. Shalev-Shwartz and T. Zhang. Stochastic dual coordinate ascent methods for regularized loss minimization. *Journal of Machine Learning Research*, 14(Feb):567–599, 2013.

O. Shamir, N. Srebro, and T. Zhang. Communication-efficient distributed optimization using an approximate newton-type method. In *Proceedings of The 31st International Conference on Machine Learning*, pages 1000–1008, 2014.

W. Shi, Q. Ling, K. Yuan, G. Wu, and W. Yin. On the linear convergence of the admm in decentralized consensus optimization. *IEEE Transactions on Signal Processing*, 62(7):1750–1761, 2014.

V. Smith, S. Forte, M. I. Jordan, and M. Jaggi. L1-regularized distributed optimization: A communication-efficient primal-dual framework. *arXiv preprint arXiv:1512.04011*, 2015.

E. P. Xing, Q. Ho, W. Dai, J. K. Kim, J. Wei, S. Lee, X. Zheng, P. Xie, A. Kumar, and Y. Yu. Petuum: A new platform for distributed machine learning on big data. *IEEE Transactions on Big Data*, 1(2):49–67, 2015.

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

A. W. Yu, Q. Lin, and T. Yang. Doubly stochastic primal-dual coordinate method for regularized empirical risk minimization with factorized data. *CoRR, abs/1508.03390*, 2015.

Y. Zhang and L. Xiao. Disco: Distributed optimization for self-concordant empirical loss. In *Proceedings of the 32nd International Conference on Machine Learning*, pages 362–370, 2015a.

Y. Zhang and L. Xiao. Stochastic primal-dual coordinate method for regularized empirical risk minimization. In *Proceedings of The 32nd International Conference on Machine Learning*, pages 353–361, 2015b.

Y. Zhang, M. J. Wainwright, and J. C. Duchi. Communication-efficient algorithms for statistical optimization. In *Advances in Neural Information Processing Systems*, pages 1502–1510, 2012.