Group-Based Alternating Direction Method of Multipliers for Distributed Linear Classification

Huihui Wang, Yang Gao, Yinghuan Shi, and Ruili Wang

***Abstract*—The alternating direction method of multipliers (ADMM) algorithm has been widely employed for distributed machine learning tasks. However, it suffers from several limitations, e.g., a relative low convergence speed, and an expensive time cost. To this end, in this paper, a novel method, namely the group-based ADMM (GADMM), is proposed for distributed linear classification. In particular, to accelerate the convergence speed and improve global consensus, a group layer is first utilized in GADMM to divide all the slave nodes into several groups. Then, all the local variables (from the slave nodes) are gathered in the group layer to generate different group variables. Finally, by using a weighted average method, the group variables are coordinated to update the global variable (from the master node) until the solution of the global problem is reached. According to the theoretical analysis, we found that: 1) GADMM can mathematically converge at the rate $O(1/k)$, where k is the number of outer iterations and 2) by using the grouping methods, GADMM can improve the convergence speed compared with the distributed ADMM framework without grouping methods. Moreover, we systematically evaluate GADMM on four publicly available LIBSVM datasets. Compared with disADMM and stochastic dual coordinate ascent with alternating direction method of multipliers-ADMM, for distributed classification, GADMM is able to reduce the number of outer iterations, which leads to faster convergence speed and better global consensus. In particular, the statistical significance test has been experimentally conducted and the results validate that GADMM can significantly save up to 30% of the total time cost (with less than 0.6% accuracy loss) compared with disADMM on large-scale datasets, e.g., webspam and epsilon.**

***Index Terms*—Alternating direction method of multipliers (ADMM), distributed linear classification, group-based ADMM (GADMM), support vector machine (SVM).**

I. Introduction

In RECENT years, linear classification [1], [2] has become a crucial research topic in machine learning, especially for dealing with large-scale datasets. Thus, many efficient linear classification algorithms have been proposed, e.g., gradient method [3]–[5], mirror descent [6], and dual averaging method [7]. Nowadays, regarding the amount of data in our world has been exploding, and also the data is usually stored in a distributed environment, these conventional linear classification algorithms, which run on a single computer, become infeasible and impracticable for directly handling large-scale datasets in real practice. Therefore, distributed classification algorithms have been developed for solving the large-scale classification problem [8], [9].

Basically, these distributed classification algorithms can be categorized in two classes: 1) the primal problem-oriented algorithms and 2) the dual problem-oriented algorithms [10]–[12]. In the first class, gradient methods were adopted to handle the gradients of parallel support vector machines (SVMs) in the Cascade SVM [13]. In the other class, the alternating direction method of multipliers (ADMM) [14] is one of the most well-known algorithms. ADMM decomposes the original problem into two local subproblems, and then solves them in an alternating fashion. Recent works in [15] and [16] proved that ADMM has a linear convergence rate under favorable assumptions, and has been adopted to solve large-scale machine learning problems. In addition, a recent study [2] has shown that, for large-scale classification, the performance (e.g., accuracy) of linear algorithms is close to that of nonlinear algorithms, but with much less training time. Therefore, we study ADMM for large-scale linear classification.

Recently, according to the different network-connection modes, the distributed ADMM can be classified into two categories: 1) peer-to-peer mode, which can be represented by an undirected/directed graph [17]–[21], and nodes are allowed to exchange information with their neighbors and 2) master-slave mode, where an unique master node controls slave nodes [22]. Also, according to the different communication protocols, the distributed ADMM can be summarized into two classes: 1) asynchronous, where some nodes of the network are allowed to wake up at random and perform local updates in a noncoordinated fashion [23], [24] and 2) synchronous, where the master node is triggered only if it receives the information from all the slave nodes in each iteration [25]. In this paper, we only focus on the synchronous distributed ADMM under the master-slave mode. In distributed linear classification algorithms, many synchronous distributed ADMM algorithms have recently attracted much research interest due to the high decomposability of ADMM [26], [27].

Manuscript received September 2, 2015; revised January 6, 2016 and March 20, 2016; accepted May 10, 2016. This work was supported in part by the National Science Foundation of China under Grant 61432008, Grant 61305068, and Grant 61321491, in part by the Collaborative Innovation Center of Novel Software Technology and Industrialization, and in part by the full Marsden Grant in New Zealand. This paper was recommended by Associate Editor Y. Zhang. *(Corresponding author: Yang Gao.)*

H. Wang, Y. Gao, and Y. Shi are with the State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210023, China (e-mail: huihui.wang8917@gmail.com; gaoy@nju.edu.cn; syh@nju.edu.cn).

R. Wang is with the School of Engineering of Advanced Technology, Massey University, Auckland 102-904, New Zealand (e-mail: R.wang@massey.ac.nz).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCYB.2016.2570808

Specifically, Forero *et al*. [28] proposed a method to train an SVM model via ADMM in a fully distributed fashion by decomposing the classification problem as a series of subproblems. Zhang *et al*. [25] proposed a novel distributed linear classification method via ADMM (namely disADMM in this paper). In disADMM, subproblems can be solved by linear classification algorithms, and the local variables of the subproblems on slave nodes were used to guide the global variable optimization. Also, the distributed ADMM was extended to train the linear classifier for specific tasks, such as face recognition and structured SVM [29], [30]. To discuss communication-efficient for distributed methods, Arjevani and Shamir [31] have proved the number of communication rounds under different assumptions from theoretical perspective. Vemula [22] proposed a method, ADMM-Block Coordinate Descent for distributed learning problems and performed a comparison with ADMM-Dual Coordinate Ascent. A stochastic dual coordinate ascent with ADMM and the mini-batch extension also were proposed in [32].

Conventional distributed ADMM algorithms (i.e., the synchronous distributed ADMM) are based on the idea of global variable consensus, i.e., all the local variables of the subproblems on the slave nodes are required to be in consensus with the global variable when the final solution is found. Although the previous distributed ADMM algorithms [25], [28] for linear classification have been validated for promising results, they still suffer from several limitations, from either theoretical or practical perspectives: 1) they are normally lacking in theoretical analysis, which is needed to deeply understand the scalability and generalization ability of the distributed algorithms and 2) they usually have low convergence speed and expensive communication cost [23], [33].

To this end, a novel distributed linear classification method, named group-based ADMM (GADMM), is proposed to accelerate the convergence speed and improve global consensus under the synchronous mechanism in this paper. During the optimization process, previous distributed ADMM algorithms [25], [27] update the global variable directly using the local variables, and all the local variables are required to reach consensus with the global variable until the final consensus is found, but this process is time-consuming. To obtain the faster convergence speed and better global consensus, a group layer is added into the framework of GADMM, which generates the group variables obtained by local variable grouping. In contrast with the previous distributed ADMM algorithms, instead of using local variables, the group variables are aggregated to update the global variable, and then reupdate the local variables in the associated groups in order to find the solution to the global problem. The global consensus in distributed ADMM can be regarded as seeking consensus between the newly introduced group and global variables in GADMM. Since the number of group variables is much smaller than that of the local variables (using grouping techniques), faster convergence speed can be expected. The theoretical analysis will be presented in Section IV. Therefore, the highlights of our GADMM are that: 1) the updating strategy for global variables is novel, as it leverages the group layer and 2) the theoretical analysis is sufficiently sound to guarantee the performance of GADMM. The intuitive illustration of the major difference between the distributed ADMM and our method can be found in Fig. 1.

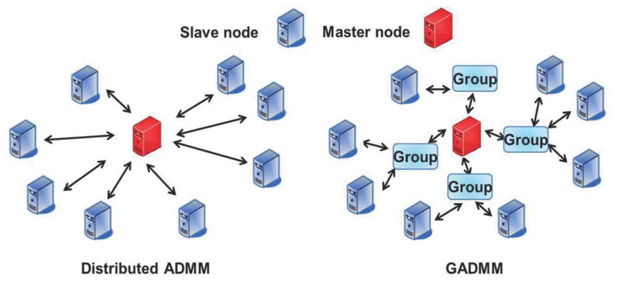

Fig. 1. Intuitive illustrations of the distributed ADMM (left) and GADMM (right). In the distributed ADMM, the global variable on the master node is updated directly using local variables on slave nodes, while the group variables in the group layer are used to update the global variable in GADMM.

In particular, the local variables of the subproblems on the slave nodes are utilized as features to measure the similarities between datasets on different slave nodes. GADMM divides slave nodes into several groups by grouping methods (i.e., clustering) [34]–[36] in the group layer. After grouping, GADMM aggregates the similar local variables of subproblems in the same group to generate the group variable of the associated group. Then, group variables are coordinated to update the global variable by the weighted average method. After that, the local variables are reupdated of the subproblems by the group variable in the same group. The experimental results on four benchmark datasets demonstrate that GADMM can achieve a significant improvement in the number of outer iterations compared with disADMM [25] and stochastic dual coordinate ascent with alternating direction method of multipliers (SDCA)-ADMM [32]. Moreover, our experiments have shown that GADMM can significantly reduce the total time cost (the running time cost and communication time cost) with an acceptable accuracy loss on large-scale datasets.

The main contributions of this paper are threefold and can be summarized in the following.

1) A novel distributed linear classification method, GADMM, is proposed by introducing the group layer. The original global consensus problem in the distributed ADMM can be regarded as seeking a consensus solution between the group and global variables, thus faster convergence speed and better global consensus can be achieved.
2) The convergence analysis of our GADMM is presented, which shows that GADMM has a convergence rate of $O(1/k)$, where k is the number of outer iterations. Moreover, GADMM can reduce the number of outer iterations and thus improve the convergence speed compared with the distributed ADMM, according to our theoretical analysis.
3) We present a practical version of GADMM and compare it with disADMM and SDCA-ADMM. The experimental results on four benchmark datasets show that GADMM reduces the number of outer iterations significantly, and saves up to 30% of the total time

cost compared with disADMM, with less than 0.6% accuracy loss.

The remainder of this paper is organized as follows. ADMM and the distributed ADMM framework for linear classification are briefly described in Section II. The framework and procedures of GADMM are introduced in Section III. In Section IV, we present the convergence analysis of GADMM and the convergence speed comparison between GADMM and the conventional distributed ADMM. Section V reports experimental results evaluated on four publicly available LIBSVM datasets. Finally, Section VI gives some concluding remarks.

II. Relevant Background

A. Alternating Direction Method of Multipliers

According to previous studies [15], [27], ADMM has been demonstrated its effectiveness and scalability in solving convex optimization problems with equality constraints. Normally, ADMM solves problems in the following form:

$$\begin{aligned}\min_{\mathbf{x},\mathbf{z}}\ & f(\mathbf{x})+g(\mathbf{z})\\ \text{s.t.}\ & \mathbf{Ax}+\mathbf{Bz}=\mathbf{C}\end{aligned}\tag{1}$$

where $\mathbf{x}\in\mathbb{R}^{N_x}$ and $\mathbf{z}\in\mathbb{R}^{N_z}$, $f(\mathbf{x})$ and $g(\mathbf{z})$ are two convex functions. $\mathbf{A}\in\mathbb{R}^{N_C\times N_x}$, $\mathbf{B}\in\mathbb{R}^{N_C\times N_z}$, and $\mathbf{C}\in\mathbb{R}^{N_C}$ are used to impose the linear constraints. Basically, the augmented Lagrangian methods [37], [38] for the original problem in (1) can be mathematically formulated as

$$\begin{aligned}L_\rho(\mathbf{x},\mathbf{z},\boldsymbol{\lambda})=&f(\mathbf{x})+g(\mathbf{z})+\boldsymbol{\lambda}^T(\mathbf{Ax}+\mathbf{Bz}-\mathbf{C})\\ &+\frac{\rho}{2}\|\mathbf{Ax}+\mathbf{Bz}-\mathbf{C}\|^2\end{aligned}\tag{2}$$

where $\boldsymbol{\lambda}>0$ is a dual variable, and $\rho>0$ is a penalty parameter. For simplification, the linear and quadratic terms can be combined by scaling $\boldsymbol{\lambda}$ in (2). Thus, ADMM can be rewritten in a slightly scaled form [27] as follows:

$$L_\rho(\mathbf{x},\mathbf{z},\mathbf{u})=f(\mathbf{x})+g(\mathbf{z})+\frac{\rho}{2}\|\mathbf{Ax}+\mathbf{Bz}-\mathbf{C}+\mathbf{u}\|^2\tag{3}$$

where $\mathbf{u}=(1/\rho)\boldsymbol{\lambda}$. The scaled form in (3) is clearly equivalent to the augmented Lagrangian in (2). Furthermore, the augmented Lagrangian is more convenient to solve the original problems. In the $(k+1)_{th}$ iteration, the alternating minimization over $\mathbf{x}$ and $\mathbf{z}$ is performed with the following steps:

$$\mathbf{x}^{k+1}=\arg\min_{\mathbf{x}} f(\mathbf{x})+\frac{\rho}{2}\left\|\mathbf{Ax}+\mathbf{Bz}^k-\mathbf{C}+\mathbf{u}^k\right\|^2\tag{4}$$

$$\mathbf{z}^{k+1}=\arg\min_{\mathbf{z}} g(\mathbf{z})+\frac{\rho}{2}\left\|\mathbf{Ax}^{k+1}+\mathbf{Bz}-\mathbf{C}+\mathbf{u}^k\right\|^2\tag{5}$$

$$\mathbf{u}^{k+1}=\mathbf{u}^k+\left(\mathbf{Ax}^{k+1}+\mathbf{Bz}^{k+1}-\mathbf{C}\right).\tag{6}$$

B. Distributed ADMM Framework for Linear Classification

According to [25] and [27], ADMM can be applied to distributed computing due to its high decomposability property. If function $f(\mathbf{x})$ is separable with respect to variable $\mathbf{x}$, it can be rewritten as

$$f(\mathbf{x})=\sum_{i=1}^n f_i(\mathbf{x}_i)$$

where $\mathbf{x}=(\mathbf{x}_1,\ldots,\mathbf{x}_n)$. Therefore, the original problem in (1) can be reformed as

$$\begin{aligned}\min_{\mathbf{x}_i,\ldots,\mathbf{x}_n,\mathbf{z}}\ & \sum_{i=1}^n f_i(\mathbf{x}_i)+g(\mathbf{z})\\ \text{s.t.}\ & \mathbf{Ax}_i+\mathbf{Bz}=\mathbf{C},\ i=1,2,\ldots,n.\end{aligned}\tag{7}$$

The problem in (7) is the global consensus problem [28], where $\mathbf{x}_i$ and $\mathbf{z}$ are the local and global variables, respectively. Typically, $f(\mathbf{x})$ can be split into n separate subproblems $f_i(\mathbf{x}_i)$ solved in terms of $\mathbf{x}_i$ in parallel

$$\mathbf{x}_i^{k+1}=\arg\min_{\mathbf{x}_i} f_i(\mathbf{x}_i)+\frac{\rho}{2}\left\|\mathbf{Ax}_i+\mathbf{Bz}^k-\mathbf{C}+\mathbf{u}_i^k\right\|^2.\tag{8}$$

Supposing the data is split into n disjoint datasets $(D_1,D_2,\ldots,D_n)$ stored on n slave nodes $(S_1,S_2,\ldots,S_n)$, respectively, by adopting the L2-regularized L1-loss SVM [39], [40] as the distributed linear classification model, ADMM transforms the classification problem into a global consensus problem as follows:

$$\begin{aligned}\min_{\mathbf{w}_i,\ldots,\mathbf{w}_n,\mathbf{z}}\ & \sum_{i=1}^n f_i(\mathbf{w}_i)+g(\mathbf{z})\\ \text{s.t.}\ & \mathbf{w}_i=\mathbf{z},\ i=1,2,\ldots,n\end{aligned}\tag{9}$$

where $f_i(\mathbf{w}_i)=C\sum_{j=1}^s\max\{0,1-y_j\mathbf{w}_i^T\mathbf{x}_j\}$, $g(\mathbf{z})=(1/2)\|\mathbf{z}\|^2$. C is a penalty parameter. $\mathbf{x}_j\in\mathbb{R}^d$ is an example with $y_j\in\{-1,+1\}$ as its label, and s is the number of examples in dataset D_i. The augmented Lagrangian function for distributed classification in (9) can also be rewritten in a scaled form

$$L_\rho(\mathbf{w},\mathbf{z},\mathbf{u})=g(\mathbf{z})+\sum_{i=1}^n\Big(f_i(\mathbf{w}_i)+\frac{\rho}{2}\|\mathbf{w}_i-\mathbf{z}+\mathbf{u}_i\|^2\Big).\tag{10}$$

The distributed classification problem in (9) can be solved by the alternating minimization over $\mathbf{w}_i$ and $\mathbf{z}$ as follows:

$$\mathbf{w}_i^{k+1}=\arg\min_{\mathbf{w}_i} f_i(\mathbf{w}_i)+\frac{\rho}{2}\left\|\mathbf{w}_i-\mathbf{z}^k+\mathbf{u}_i^k\right\|^2\tag{11}$$

$$\mathbf{z}^{k+1}=\arg\min_{\mathbf{z}} g(\mathbf{z})+\frac{\rho}{2}\sum_{i=1}^n\left\|\mathbf{w}_i^{k+1}-\mathbf{z}+\mathbf{u}_i^k\right\|^2\tag{12}$$

$$\mathbf{u}_i^{k+1}=\mathbf{u}_i^k+\mathbf{w}_i^{k+1}-\mathbf{z}^{k+1}.\tag{13}$$

Note that in the global consensus problem (9), each computing node (slave node) only optimizes its local variable $\mathbf{w}_i$ and sends it to the master node. Then, the local variables are averaged to obtain a joint global variable $\mathbf{z}$. After the updating of $\mathbf{z}$, the master node broadcasts it to all the slave nodes for the updating of dual variables $\mathbf{u}_i$s and optimization of local variables $\mathbf{w}_i$s in the next iteration. The process repeats until all the local variables reach consensus with the global variable. In the distributed ADMM, the total time cost is the sum of the computation time and the communication time. However, under the synchronous protocol, the master node is triggered only if it receives the required message from all the slave nodes. Therefore, the speed of distributed ADMM is limited by the slowest worker nodes especially when the nodes have different computation and communication delays. This will create a new bottleneck that the conventional distributed ADMM suffers from low convergence speed and expensive time cost in practice [2], [23].

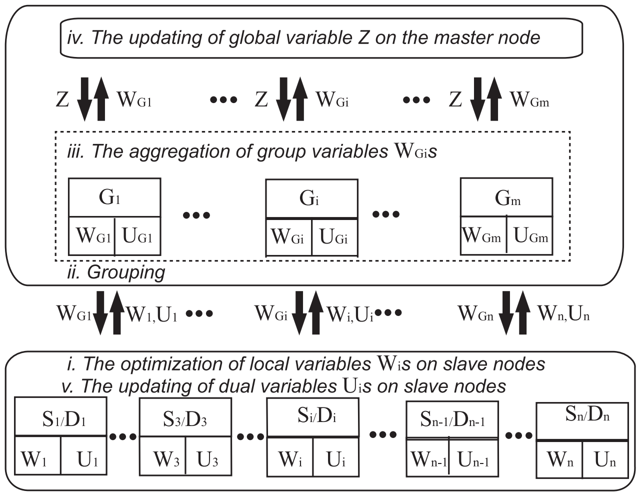
iv. The updating of global variable Z on the master node
Z
WG1
Z
WGi
Z
WGm
iii. The aggregation of group variables WGis
G1
WG1
UG1
Gi
WGi
UGi
Gm
WGm
UGm
ii. Grouping
WG1
W1,U1
WGi
Wi,Ui
WGn
Wn,Un
i. The optimization of local variables Wis on slave nodes
v. The updating of dual variables Uis on slave nodes
S1/D1
W1
U1
S3/D3
W3
U3
Si/Di
Wi
Ui
Sn-1/Dn-1
Wn-1
Un-1
Sn/Dn
Wn
Un

Fig. 2. Illustration of the framework of GADMM. In (i), local variables $\mathbf{w}_i$s of subproblems are optimized on slave nodes in parallel. Then, local variables $\mathbf{w}_i^1$s are sent to the group layer for grouping in (ii). After that, local variables in the same group are used to generate group variable $\mathbf{w}_{Gj}$ in (iii). Global variable $\mathbf{z}$ is updated with the group variables according to its update procedure in (iv). Finally, the master node broadcasts the global variable and group variables to the slave nodes, and then dual variables $\mathbf{u}_i$s in (v) are updated for the next iteration.

III. Group-Based Alternating Direction Method for Multipliers

As indicated above, the conventional distributed ADMM has the following deficiencies: 1) an expensive time cost and 2) a relatively low convergence speed (a detailed comparative study can be found in Section V-B). Therefore, we propose the GADMM to prevent the aforementioned deficiencies. In the following sections, we will describe the details of GADMM, and then give the stopping criteria for experiments.

A. Distributed Framework of Group-Based Alternating Direction Method of Multipliers

The framework of GADMM is illustrated in Fig. 2. First, local variables $\mathbf{w}_i$s of subproblems are optimized by a dual coordinate descent method (DCD) on slave nodes in parallel. Second, according to $\mathbf{w}_i^1$s in the first outer iteration, all the slave nodes (i.e., $S_1, S_2, \ldots, S_n$) are divided into several groups (i.e., $G_1, G_2 \ldots, G_m$, where m is the number of groups) by grouping methods, which will be described in Section III-B. Once the number of groups and slave nodes in different groups are determined, the group information stored on the master node is fixed and immovable. Third, local variables $\mathbf{w}_i$s in the same group are used to generate group variable $\mathbf{w}_{Gj}$. Fourth, global variable $\mathbf{z}$ is updated by the aggregation of variables $\mathbf{w}_{Gj}$s and $\mathbf{u}_{Gj}$s on the master node. Finally, dual variables $\mathbf{u}_i$s are updated on the slave nodes in parallel. All the procedures, with the exception of grouping, are iterated until convergence.

To get a faster convergence speed, a group layer is added to divide slave nodes whose subproblems have similar local variables into the same group in GADMM. Supposing all n slave nodes are divided into m groups, each subproblem can be mathematically formulated

$$\mathbf{w}_i^{k+1} = \arg\min_{\mathbf{w}_i} f_i(\mathbf{w}_i) + \frac{\rho}{2}\left\|\mathbf{w}_i - \mathbf{z}^k + \mathbf{u}_i^k\right\|^2. \quad (14)$$

Compared with the conventional distributed ADMM, the difference is that group variable $\mathbf{w}_{Gj}$ of group G_j is generated as follows:

$$\mathbf{w}_{Gj}^{k+1} = \sum_{S_i \in G_j} \beta_{i,j}\mathbf{w}_i^{k+1} \quad (15)$$

where $S_i \in G_j$ means that slave node S_i belongs to group G_j and $\beta_{i,j} > 0$ (described in Section III-D) is the weight parameter of local variable $\mathbf{w}_i$ in group G_j. In the group layer, all the slave nodes are divided into several groups by local variables grouping, which might result in different groups containing different numbers of the slave nodes. Considering that the different sizes of groups might affect the updating of the global variable in GADMM, the weighted average method is used to prevent the potential problem. Therefore, the updating of $\mathbf{z}$ can be mathematically formulated as follows:

$$\mathbf{z}^{k+1} = \arg\min_{\mathbf{z}} g(\mathbf{z}) + \frac{\rho}{2}\sum_{j=1}^{m}\frac{mn_j}{n}\left\|\mathbf{w}_{Gj}^{k+1} - \mathbf{z} + \mathbf{u}_{Gj}^k\right\|^2 \quad (16)$$

$$\mathbf{u}_{Gj}^{k+1} = \mathbf{u}_{Gj}^k + \mathbf{w}_{Gj}^{k+1} - \mathbf{z}^{k+1} \quad (17)$$

where n_j is the number of slave nodes in group G_j. After that, the local variables of subproblems on slave nodes in the same group are reupdated

$$\mathbf{w}_i^{k+1} = \mathbf{w}_{Gj}^{k+1}, S_i \in G_j, i = 1, \ldots, n, j = 1, \ldots, m. \quad (18)$$

Finally, the dual variables of the subproblems are updated on slave node as follows:

$$\mathbf{u}_i^{k+1} = \mathbf{u}_i^k + \mathbf{w}_i^{k+1} - \mathbf{z}^{k+1}. \quad (19)$$

Although each local variable $\mathbf{w}_i$ is forced to reach consensus with global variable $\mathbf{z}$ in (9), basically, it only requires that group variables $\mathbf{w}_{Gj}$s reach consensus with the global variable in (18) until GADMM finds the solution to the global consensus problem. In the following, a more detailed description of GADMM will be presented.

B. Model Similarity-Based Grouping Methods

In the grouping phase, the important issue is how to appropriately divide these slave nodes into groups. It is infeasible to measure the similarities between examples on different slave nodes because of the expensive communication cost. The computation of the mean and covariance matrix of a large-scale dataset is also extremely expensive. To a certain extent, we consider that the local variables of the subproblems can be used as the features of datasets to measure the similarities between datasets. By using grouping methods, the local variables with intragroup are similar, while the local variables with intergroup are dissimilar, thus, the local variables of the subproblems are suitable and reasonable as features to measure the similarities between datasets. Each slave node learns its local variable $\mathbf{w}_i$ from its own examples, as no communication cost between slave nodes is involved. Moreover, using local variables as features, no additional computation cost will be involved. Therefore, the local variables $\mathbf{w}_i^1$s are used as features of datasets for grouping. Moreover, to preferably using local variables of subproblems for grouping, the local variables $\mathbf{w}_i^1$s are iterative optimized for M times, where M is the number of inner iterations (will be described in Section V-A).

To sufficiently evaluate the performance of GADMM, we typically utilize two clustering algorithms based on local variables for grouping: 1) agglomerative Nesting (AGNES) [34] and 2) L method [35], which is adopted to determine a suitable number of clusters.

In AGNES, each example forms a small cluster by itself, and then the two closest clusters are merged to form a cluster until AGNES receives the predefined number of clusters. However, the downside of this method is that the number of clusters k needs to be predetermined. To solve the above problem, the L method is adopted in this paper, which can automatically determine a suitable number of clusters from any hierarchical clustering methods [35]. First, we initially group the slave nodes into several groups by measuring the distances between different slave nodes. Second, an evaluation graph containing the information of clusters is created by clustering. Finally, the L method is used to efficiently determine a suitable number of clusters by finding the knee of an evaluation graph curve.

The Euclidean distance is utilized to measure the similarities between different slave nodes. The grouping step needs to be done only once, and the information of groups, e.g., group variables, is stored on the master node. Note that many other clustering methods can be directly borrowed in GADMM. However, it is out of the scope to list and compare all the related clustering methods.

C. Subproblem Optimization

We adopt an efficient DCD, which is similar to that in [41], to optimize local variable $\mathbf{w}_i$ on the SVM dual objective in this section. Since $L_\rho(\mathbf{w}, \mathbf{z}, \mathbf{u})$ is separable in terms of $\mathbf{w}_i$, the $\mathbf{w}$-minimization problem can be split into n separated subproblems that can be solved in parallel. Therefore, in the $(k+1)$th iteration, local variable $\mathbf{w}_i$ can be rewritten as follows:

$$\mathbf{w}_i^{k+1} = \arg\min_{\mathbf{w}_i} f_i(\mathbf{w}_i) + \frac{\rho}{2}\left\|\mathbf{w}_i - \mathbf{z}^k + \mathbf{u}_i^k\right\|^2. \tag{20}$$

The Lagrangian function for (20) is

$$L_\rho\left(\mathbf{w}_i^k, \boldsymbol{\alpha}\right) = \frac{\rho}{2}\left\|\mathbf{w}_i^k - \mathbf{z}^k + \mathbf{u}_i^k\right\|^2 - \sum_{j=1}^{\mathbf{s}} \alpha_j \mathbf{y}_j\left(\mathbf{w}_i^k \mathbf{x}_j\right) + \sum_{j=1}^{\mathbf{s}} \alpha_j$$

where s is the number of examples in dataset D_i. Setting the partial derivative of local variable $\mathbf{w}_i$ to zero

$$\nabla_{\mathbf{w}_i^k} L_\rho\left(\mathbf{w}_i^k, \boldsymbol{\alpha}\right) = \rho\left(\mathbf{w}_i^k - \mathbf{z}^k + \mathbf{u}_i^k\right) - \sum_{j=1}^{\mathbf{s}} \alpha_j \mathbf{y}_j \mathbf{x}_j \to 0.$$

Thus

$$\mathbf{w}_i^k = \frac{\sum_{j=1}^{\mathbf{s}} \alpha_j \mathbf{y}_j \mathbf{x}_j + \rho \mathbf{v}_i^k}{\rho} \tag{21}$$

where $\mathbf{v}_i^k = (\mathbf{z}^k - \mathbf{u}_i^k)$. The optimal solution of the primal problem in (20) can be obtained by solving its dual problem

$$\min_{\boldsymbol{\alpha}} \ f(\boldsymbol{\alpha}) = \frac{1}{2\rho}\boldsymbol{\alpha}^T \mathbf{Q} \boldsymbol{\alpha} - \mathbf{b}_i^T \boldsymbol{\alpha}$$
$$\text{s.t.} \ \ 0 \le \alpha_j \le C, \ j = 1, \ldots, s \tag{22}$$

where $Q_{jt} = \mathbf{y}_j \mathbf{y}_t \mathbf{x}_j^T \mathbf{x}_t$, $\mathbf{b}_i = [1 - \mathbf{y}_1 \mathbf{v}_i^T \mathbf{x}_1, \ldots, 1 - \mathbf{y_s} \mathbf{v}_i^T \mathbf{x}_s]^T$. For problem (22), it optimizes α_j^{k+1} while fixing all the other variables. The partial derivative of $f(\boldsymbol{\alpha})$ is denoted as $\triangledown_j f(\boldsymbol{\alpha}^k)$ with respect to α_j in the kth iteration

$$\triangledown_j f\left(\boldsymbol{\alpha}^k\right) = \frac{1}{\rho}\sum_{t=1}^{\mathbf{s}} \alpha_t^k Q_{jt} - b_{ij} \tag{23}$$

where $\triangledown_j f(\boldsymbol{\alpha}^k)$ means the projected gradient, to determine the stopping criteria

$$\triangledown_j f\left(\boldsymbol{\alpha}^k\right) = \begin{cases} \min\left(0, \triangledown_j f\left(\boldsymbol{\alpha}^k\right)\right) & \text{if } \alpha_j^k = 0 \\ \max\left(0, \triangledown_j f\left(\boldsymbol{\alpha}^k\right)\right) & \text{if } \alpha_j^k = C \\ \triangledown_j f\left(\boldsymbol{\alpha}^k\right) & \text{if } 0 < \alpha_j^k < C. \end{cases} \tag{24}$$

Then, α_j can be updated

$$\alpha_j^{k+1} = \min\left(\max\left(\alpha_j^k - \frac{\triangledown_j f\left(\boldsymbol{\alpha}^k\right)}{Q_{jj}}, 0\right), C\right).$$

According to (21), (23) can be simplified as

$$\triangledown_j f\left(\boldsymbol{\alpha}^k\right) = \mathbf{y}_j \mathbf{w}_j^k \mathbf{x}_j - 1. \tag{25}$$

Thus, the update procedure of $\mathbf{w}_i$ is

$$\mathbf{w}_i^{k+1} = \mathbf{w}_i^k + \left(\alpha_j^{k+1} - \alpha_j^k\right)\mathbf{y}_j \mathbf{x}_j. \tag{26}$$

The objective in (20) can be seen as L1-SVM in terms of $\mathbf{w}_i$. Therefore, the dual problem in (22) of the subproblems is similar to that in [41] in the dual SVM form. According to [25] and [41], the DCD can converge to an ϵ-accurate solution [i.e., $f(\boldsymbol{\alpha}) \le f(\boldsymbol{\alpha}^*) + \epsilon$] in $O(\log(1/\epsilon))$ iterations.

D. Update Procedures of Group and Global Variables

After all the n slave nodes have been divided into m groups ($m \le n$), now we describe the updating of group variable $\mathbf{w}_{Gj}$. Here, we define group G_j as a set of slave nodes $S_{i,j} \in \{S_i | S_i \in G_j\}$, and dataset D_i on slave node $S_{i,j}$ is also named as $D_{i,j}$. The variable $\sigma_{i,j}$, as the ratio of the positive and negative examples in dataset $D_{i,j}$ is defined as

$$\sigma_{i,j} = \frac{\sigma_{i,j}^+ * \sigma_{i,j}^-}{\sum_{S_i \in G_j} \sigma_{i,j}^+ * \sigma_{i,j}^-}$$

where variables $\sigma_{i,j}^+$ and $\sigma_{i,j}^-$ are the ratios of different binary label examples in dataset $D_{i,j}$. Moreover, taking the numbers of examples on slave nodes in different groups into account, the size ratio $\gamma_{i,j}$ of dataset $D_{i,j}$ in group G_j can be defined as

$$\gamma_{i,j} = \frac{\left|D_{i,j}\right|}{\sum_{S_i \in G_j}\left|D_{i,j}\right|}$$

where $|D_{i,j}|$ is the number of examples in dataset $D_{i,j}$. Therefore, $\mathbf{w}_{Gj}$ can be defined as follows:

$$\mathbf{w}_{Gj} = \sum_{S_i \in G_j} \beta_{i,j} * \mathbf{w}_{i,j} \tag{27}$$

where synthesis ratio $\beta_{i,j} = ((\sigma_{i,j} * \gamma_{i,j})/(\sum_{S_i \in G_j}(\sigma_{i,j} * \gamma_{i,j})))$, is the weight of slave node S_i in group G_j. Furthermore, in the conventional distributed ADMM, the updating of global variable $\mathbf{z}^{k+1}$ in the $(k+1)_{th}$ iteration can be formulated as

$$\mathbf{z}^{k+1} = \arg\min_{\mathbf{z}} g(\mathbf{z}) + \frac{\rho}{2}\sum_{i=1}^{n}\left\|\mathbf{w}_i^{k+1} - \mathbf{z} + \mathbf{u}_i^k\right\|^2.$$

Global variable $\mathbf{z}^{k+1}$ can be obtained by a simple average of $\mathbf{w}_i^{k+1}$ and $\mathbf{u}_i^k$. In contrast with the distributed ADMM, the group variables are used to update global variable $\mathbf{z}^{k+1}$ in GADMM. Considering that the different sizes of groups might affect the performance of GADMM, the weighted average method is used to prevent the potential problem. Therefore, the updating of the global variable can be mathematically formulated as follows:

$$\mathbf{z}^{k+1} = \arg\min_{\mathbf{z}} g(\mathbf{z}) + \frac{\rho}{2}\sum_{j=1}^{m}\frac{mn_j}{n}\left\|\mathbf{w}_{Gj}^{k+1} - \mathbf{z} + \mathbf{u}_{Gj}^k\right\|^2$$

where n_j is the number of slave nodes in group G_j. Since the optimization objective of global variable $\mathbf{z}^{k+1}$ is differentiable, it can be solved by the weighted average of $\mathbf{w}_{Gj}^{k+1}$ and $\mathbf{u}_{Gj}^k$

$$\mathbf{z}^{k+1} = \frac{m\rho}{1+m\rho}\sum_{j=1}^{m}\frac{n_j}{n}\left(\mathbf{w}_{Gj}^{k+1} + \mathbf{u}_{Gj}^k\right). \tag{28}$$

E. Stopping Criteria

In this paper, we refer to the primal residual as $\mathbf{r}^{k+1} = \mathbf{w}^{k+1} - \mathbf{z}^{k+1}$, and the dual residual as $\mathbf{s}^{k+1} = -\rho(\mathbf{z}^{k+1} - \mathbf{z}^k)$ in the $(k+1)$th iteration. It has been proved that these two residuals converge to zero as those methods proceed [27]. This means that the primal and dual residual should be small enough to stop the iteration, as shown below

$$\left\|\mathbf{r}^k\right\|_2 \le \epsilon^{\text{pri}} \quad \left\|\mathbf{s}^k\right\|_2 \le \epsilon^{\text{dual}} \tag{29}$$

where ϵ^{pri} and ϵ^{dual} are feasible tolerances. Normally, the tolerances used in our experiments are

$$\epsilon^{\text{pri}} = \sqrt{dn} * A + R * \max(\|\mathbf{w}\|_2, \|\mathbf{z}\|_2)$$
$$\epsilon^{\text{dual}} = \sqrt{dn} * A + R * \|\rho\mathbf{u}\|_2$$

where d is the dimensionality of examples and n is the number of slave nodes. A and R as the absolute and relative tolerances are both set to 0.001. Typically, the specific process of GADMM is described in Algorithm 1.

Algorithm 1 GADMM

```
Input: datasets: $\mathbf{D} = \{D_1, D_2, \ldots, D_n\}$, parameters ($C$, $\rho$, $\epsilon^{pri}$, $\epsilon^{dual}$, $\mathbf{r}$, $\mathbf{s}$)
Output: local variables $\mathbf{w}_1, \ldots, \mathbf{w}_n$
1: $k \leftarrow 0$
2: $\beta_{i,j} \leftarrow 0$.
3: while $\|\mathbf{r}^k\|_2 > \epsilon^{\text{pri}}$ or $\|\mathbf{s}^k\|_2 > \epsilon^{\text{dual}}$ do
4:     $\mathbf{w}_i^{k+1} = \arg\min_{\mathbf{w}_i}(f_i(\mathbf{w}_i) + \frac{\rho}{2}\|\mathbf{w}_i - \mathbf{z}^k + \mathbf{u}_i^k\|^2)$
5:     if $k == 0$ then
6:         $\mathbf{d}_{ij} \leftarrow \|(\mathbf{w}_i^1 - \mathbf{w}_j^1)\|_2$
7:         $m \leftarrow n$ slave nodes are divided into $m$ groups according to Euclidean distance $\mathbf{d}_{ij}$
8:         $\beta_{i,j} \leftarrow \frac{\sigma_{i,j}+\gamma_{i,j}}{\sum_{S_i \in G_j}(\sigma_{i,j}+\gamma_{i,j})}$
9:     end if
10:    $\mathbf{w}_{Gj}^{k+1} \leftarrow \sum_{\mathbf{s}_i \in G_j} \beta_{i,j} * \mathbf{w}_i^{k+1}$
11:    $\mathbf{z}^{k+1} \leftarrow \frac{m\rho}{1+m\rho}\sum_{j=1}^{m}\frac{n_j}{n}(\mathbf{w}_{Gj}^{k+1} + \mathbf{u}_{Gj}^k)$
12:    $\mathbf{u}_{Gj}^{k+1} \leftarrow \mathbf{u}_{Gj}^k + \mathbf{w}_{Gj}^{k+1} - \mathbf{z}^{k+1}$
13:    $\mathbf{w}_i^{k+1} \leftarrow \mathbf{w}_{Gj}^{k+1}, \quad S_i \in G_j$
14:    $\mathbf{u}_i^{k+1} \leftarrow \mathbf{u}_i^k + \mathbf{w}_i^{k+1} - \mathbf{z}^{k+1}$
15:    $\mathbf{r}^{k+1} \leftarrow \mathbf{w}^{k+1} - \mathbf{z}^{k+1}, \quad \mathbf{s}^{k+1} = -\rho(\mathbf{z}^{k+1} - \mathbf{z}^k)$
16:    $k \leftarrow k+1$
17: end while
```

IV. Theoretical Analysis

We now present our analysis of the convergence property of GADMM. Specifically, we first introduce the notations, and derive three Lemmas and one Theorem. Then, we use them to establish the convergence property of GADMM with an $O(1/k)$ convergence rate, where k is the number of outer iterations. The theoretical analysis also shows that by using the grouping methods, GADMM can improve the convergence speed compared with the conventional distributed ADMM. The proofs of the following theorem and lemmas are shown in Appendixes.

For the convenience of notation, the augmented Lagrangian for the original problem (9) can be formulated as

$$\mathcal{L}_\rho(\mathbf{w}, \mathbf{z}, \boldsymbol{\lambda}) = f(\mathbf{w}) + g(\mathbf{z}) + \boldsymbol{\lambda}(\mathbf{w} - \mathbf{z}) + \frac{\rho}{2}\|\mathbf{w} - \mathbf{z}\|^2 \tag{30}$$

where $\mathbf{w} = (\mathbf{w}_1, \ldots, \mathbf{w}_n)$ and $f(\mathbf{w}) = \sum_{i=1}^n f_i(\mathbf{w}_i)$.

A. Convergence Bound

The convergence analysis relies on the optimality of λ_i^k, which is similar to that in [42] in each iteration. This is established in the following lemmas.

Assumption 1: Both functions $f(\mathbf{w})$ and $g(\mathbf{z})$ are convex with moduli δ_f and δ_g, respectively.

Because of the property of the convex function, the solution of the original problem can be solved by its conjugate form. The conjugates of $f(\mathbf{w})$ and $g(\mathbf{z})$ can be denoted as $f^*(\boldsymbol{\lambda})$ and $g^*(\boldsymbol{\lambda})$, respectively. Thus, the conjugate of functions $f(\mathbf{w})$ and $g(\mathbf{z})$, $D(\boldsymbol{\lambda})$ can be solved as follows:

$$\arg\max \quad D(\boldsymbol{\lambda}) = -f^*(\boldsymbol{\lambda}) - g^*(\boldsymbol{\lambda}). \tag{31}$$

Note that the dual problem is maximized in order to find a solution, which meets $-\triangledown f^*(\boldsymbol{\lambda}^*) - \triangledown g^*(\boldsymbol{\lambda}^*) = 0$.

In this paper, the hinge loss term $f(\mathbf{w})$ and the regularization term $g(\mathbf{z})$ are convex functions that satisfy Assumption 1. In the $(k+1)$th iteration, the dual variable $\boldsymbol{\lambda}$ can be updated as follows:

$$\boldsymbol{\lambda}^{k+1} = \boldsymbol{\lambda}^k + \rho\left(\mathbf{w}^k - \mathbf{z}^k\right).$$

For convenience to analyze the convergence bound, we define

$$\boldsymbol{\lambda}^{\frac{1}{2}} = \boldsymbol{\lambda} + \rho\left(\mathbf{w}^{+} - \mathbf{z}\right)$$
$$\boldsymbol{\lambda}^{+} = \boldsymbol{\lambda} + \rho\left(\mathbf{w}^{+} - \mathbf{z}^{+}\right).$$

Next, we provide the convergence bound on the dual function related to dual variable $\boldsymbol{\lambda}$.

Lemma 1: If $\rho^3 \le \delta_f \delta_g^2$ [ρ is a penalty parameter of the augmented Lagrangian in (30)], δ_f and δ_g are the moduli of functions $f(\mathbf{w})$ and $g(\mathbf{z})$, respectively. Then, for any $\boldsymbol{\gamma} \in \boldsymbol{\lambda}$

$$D\left(\boldsymbol{\lambda}^{+}\right) - D(\boldsymbol{\gamma}) \ge \frac{1}{\rho}(\boldsymbol{\gamma} - \boldsymbol{\lambda})\left(\boldsymbol{\lambda} - \boldsymbol{\lambda}^{+}\right) + \frac{1}{2\rho}\left\|\boldsymbol{\lambda} - \boldsymbol{\lambda}^{+}\right\|^2.$$

The derivation of this result is given in Appendix A. With the preceding lemma, we can now obtain a global convergence rate for GADMM.

Theorem 1: Suppose that the objective functions $f(\mathbf{w})$ and $g(\mathbf{z})$ satisfy Assumption 1 and the conditions of Lemma 1. Then, for $k \ge 1$, the sequence of dual variables $\boldsymbol{\lambda}$ satisfies

$$D(\boldsymbol{\lambda}^*) - D\left(\boldsymbol{\lambda}^k\right) \le \frac{\left\|\boldsymbol{\lambda}^* - \boldsymbol{\lambda}^1\right\|^2 - \left\|\boldsymbol{\lambda}^* - \boldsymbol{\lambda}^k\right\|^2}{2\rho(k-1)}$$

where $\boldsymbol{\lambda}^*$ is the optimal value of the augmented Lagrange multiplier function. The proof of Theorem 1 is given in Appendix B. The global convergence result is quite remarkable. The GADMM method converges at the rate $O(1/k)$, where k is the number of outer iterations. The convergence bound in Theorem 1 only involves dual variables $\boldsymbol{\lambda}^*$, $\boldsymbol{\lambda}^1$, and $\boldsymbol{\lambda}^k$. Since $\boldsymbol{\lambda}^1$s are the same as those in the conventional distributed ADMM, the effect of $\|\boldsymbol{\lambda}^* - \boldsymbol{\lambda}^1\|^2$ can be ignored in the analysis. After grouping the similar slave nodes into groups, GADMM makes λ_i^ks in the same group consistent with each other. Therefore, $\|\boldsymbol{\lambda}^* - \boldsymbol{\lambda}^k\|^2$ will affect the convergence bound. To analyze the effect of grouping on the convergence bound, we suppose that all the n slave nodes are divided into m groups. Based on the average value inequality, we obtain that

$$\left\|\boldsymbol{\lambda}^* - \boldsymbol{\lambda}^k\right\|^2 = \sum_{i=1}^{n}\left(\lambda_i^* - \lambda_i^k\right)^2 \ge \frac{1}{n}\left(\sum_{i=1}^{n}\lambda_i^* - \sum_{i=1}^{n}\lambda_i^k\right)^2.$$

Local variables $\mathbf{w}_i$s in the same group are reupdated by the group variable. Thus, dual variables $\underline{\lambda_i^k}$s in the same group G_j also are reupdated by dual variable $\overline{\lambda_j^k}$ in GADMM. Thus

$$\left\|\overline{\boldsymbol{\lambda}^*} - \overline{\boldsymbol{\lambda}^k}\right\|^2 = \sum_{j=1}^{m} n_j\left(\overline{\lambda_j^*} - \overline{\lambda_j^k}\right)^2$$

where n_j is the number of the slave nodes in group G_j, $\overline{\lambda_j^*} = ((\sum_{s_i \in G_j}\lambda_i^*)/(n_j))$ and $\overline{\lambda_j^k} = ((\sum_{s_i \in G_j}\lambda_i^k)/(n_j))$. Based on the average value inequality, it also satisfies

$$-\sum_{i=1}^{n}\left(\lambda_i^* - \lambda_i^k\right)^2 \le -\sum_{j=1}^{m} n_j\left(\overline{\lambda_j^*} - \overline{\lambda_j^k}\right)^2$$
$$\le -\frac{1}{n}\left(\sum_{i=1}^{n}\lambda_i^* - \sum_{i=1}^{n}\lambda_i^k\right)^2.$$

We can find that the group strategy affects both how local variables of the subproblems in the same group are reupdated by the group variable and how is the global variable updated. Therefore, the grouping methods affect the convergence bound of GADMM for the distributed classification problem, i.e., as the number of groups increases, the upper bound of the suboptimum solution decreases, the difference between $D(\boldsymbol{\lambda}^*)$ and $D(\boldsymbol{\lambda}^k)$ will be smaller. The dual variable $\boldsymbol{\lambda}^k$, closer to the optimal variable $\boldsymbol{\lambda}^*$, will be obtained.

B. Convergence Speed

We now compare the convergence speed between GADMM and the conventional distributed ADMM. For simple notation, we define the function $F(\mathbf{w}_i)$ as

$$F(\mathbf{w}_i) = f(\mathbf{w}_i) + h(\mathbf{w}_i)$$

where $h(\mathbf{w}_i) = (\rho/2)\|(\mathbf{w}_i - \mathbf{z} + \mathbf{u}_i)\|^2$. Let $\mathbf{w}^k$, $\mathbf{u}^k$ be generated by our GADMM for the $\mathbf{w}$-minimization problem in (14).

Lemma 2: Suppose that the update procedure of the global variable $\mathbf{z}$ of GADMM is the same as that in the conventional distributed ADMM, and n slave nodes are divided into m groups in the computing cluster. For $k \ge 1$, the sequence of local variables $\{\mathbf{w}_i^k\}$ satisfies

$$\sum_{i=1}^{n} F\left(\mathbf{w}_i^k\right) \ge \sum_{j=1}^{m} n_j F\left(\sum_{S_i \in G_j} \beta_{i,j}\mathbf{w}_i^k\right)$$

where n_j is the number of the slave nodes in group G_j and $\beta_{i,j}$ is the weight of local variable $\mathbf{w}_i$ in group G_j. The proof is listed in Appendix C. Note that the difference between GADMM and the distributed ADMM is based on the sum of the sequence of local variables $\{\mathbf{w}^i\}_{i=1}^k$ in the kth iteration. The descent speed of $F(\boldsymbol{\beta}\mathbf{w}_i)$ is faster than that of $F(\mathbf{w}_i)$.

Moreover, we compare the convergence speed of the global objective between GADMM and the conventional distributed ADMM. In order to succinctly analyze the issue, we define the objectives of the conventional distributed ADMM and GADMM in the kth iteration as follows:

$$G\left(\mathbf{w}^k, \mathbf{z}^k\right) = \sum_{i=1}^{n} f_i\left(\mathbf{w}_i^k\right) + g\left(\mathbf{z}^k\right)$$
$$\overline{G}\left(\overline{\mathbf{w}^k}, \overline{\mathbf{z}^k}\right) = \sum_{i=1}^{n} f_i\left(\overline{\mathbf{w}_i^k}\right) + g\left(\overline{\mathbf{z}^k}\right).$$

According to (18), (27), and (28), $\overline{\mathbf{w}_i}^k = \sum_{S_i \in G_j}\beta_{i,j}\mathbf{w}_i^k$ and $\overline{\mathbf{z}}^k = ((m\rho)/(n(1+m\rho)))\sum_{j=1}^{m} n_j(\mathbf{w}_{Gj}^{k+1} + \mathbf{u}_{Gj}^k)$. Thus, $\sum_{i=1}^{n} f_i(\overline{\mathbf{w}_i^k}) = \sum_{j=1}^{m} n_j f_i(\sum_{S_i \in G_j}\beta_{i,j}\mathbf{w}_i^k)$.

Lemma 3: Supposing n slave nodes are divided into m groups. For $k \ge 1$, the difference between the conventional distributed ADMM and GADMM satisfies

$$G\left(\mathbf{w}^k, \mathbf{z}^k\right) - \overline{G}\left(\overline{\mathbf{w}^k}, \overline{\mathbf{z}^k}\right) \ge \frac{\rho^2 m(n-m)}{n^2(1+m\rho)^2(1+n\rho)}\left\|\mathbf{v}_i^{k+1}\right\|^2$$

where $\mathbf{v}_i^{k+1} = \sum_{j=1}^{n}(\mathbf{w}_i^{k+1} + \mathbf{u}_i^k)$. The proof result is shown in Appendix D. When $m = n$, GADMM is the same as the conventional distributed ADMM, while $m < n$, the difference between the conventional distributed ADMM and GADMM in the kth iteration is usually larger than 0. As the number of groups decreases, the difference between the conventional distributed ADMM and GADMM will increase, and the descent speed of GADMM is faster than that of the distributed ADMM. Moreover, the stopping criteria, $\|\mathbf{s}^k\|_2$ and $\|\mathbf{r}^k\|_2$ will be smaller than those of the conventional distributed ADMM. Therefore, by the grouping methods, GADMM can reduce the number of outer iterations, and converge faster than the conventional distributed ADMM.

Theorem 1 guarantees that the optimal $\boldsymbol{\lambda}^*$ of the Lagrangian function can be obtained by the sequence $\{\boldsymbol{\lambda}^k\}$ with a convergence rate of $O(1/k)$. The comparison on the convergence speed has shown that the descent speed of the objective function in GADMM is faster than that in the conventional distributed ADMM. Therefore, GADMM can theoretically accelerate the convergence speed and improve global consensus.

V. Experimental Results

A. Datasets and Experimental Settings

Four large-scale binary classification datasets: two document datasets (webspam and rcv1), an education dataset (kdd2010), and a synthetic dataset (epsilon)[1] are adopted for performance evaluation in this paper. All datasets are randomly split into training/testing datasets with a ratio of 8 : 2 and normalized for experiments. The details of these datasets are tabulated in Table I. For the implementation of the proposed GADMM, 18 computing machines are used as the slave nodes in the master-slave mode computer network, where each machine has an Intel(R) Xeon(R) CPU E5-2650 (2.6Ghz/30M Cache) processor and 64 GB RAM. Each machine S_i broadcasts $\mathbf{w}_i^{k+1}$ and $\mathbf{u}_i^k$ and waits to collect the sum of $\mathbf{w}_{Gj}^{k+1}$ and $\mathbf{u}_{Gj}^{k+1}$ for z-update. Therefore, communications involve the transmission of variables of each machine in each outer iteration, and synchronization corresponds to the waiting time to collect this message. The message passing interface [43] is used as the parallel platform, while the DCD [41] is utilized to solve the optimization of subproblems.

For parameter settings, we automatically choose the hyperparameter C determined by five-fold cross-validation, which is consistent with that of disADMM [25] for fair comparison. Moreover, local variable $\mathbf{w}_i^{k+1}$ can be replaced with its relaxation form in the updating procedures (16) and (17) to facilitate the solution

$$\mathbf{w}_i^{k+1} \leftarrow \alpha \mathbf{w}_i^{k+1} + (1-\alpha)\mathbf{z}_i^k$$

where $\alpha \in (0, 2)$ is a relaxation parameter. For the parameter α, which had been analyzed and suggested that the over-relaxation parameter $\alpha \in (1.5, 1.8)$ can improve the convergence speed [27]. In this paper, for parameters ρ, α and M (the number of inner iterations) which can be empirically well chosen, we can set them as predefined constants which are consistent with those of disADMM. Therefore, we can set (ρ, M, α) as predefined constants 1, 50, 1.6 for comparative tests to validate the performance of GADMM. In all experiments, the total time cost (Ttime) is the sum of the communication time cost (Ctime) and running time cost (Rtime). The running time cost is denoted as the computational time taken on slave nodes, while the communication time cost is the time used to transmit and receive variables between the master node and slave nodes.

TABLE I
Details of Dataset. l Is the Number of Examples, d Is the Dimension of Examples, and C Is the Hyperparameter

Dataset	l	d	C
webspam	$350,000$	$16,609,143$	32
epsilon	$500,000$	$2,000$	1
rcv1	$6,797,641$	$47,236$	1
kdd2010	$20,012,498$	$29,890,095$	1

B. Comparison With Other Distributed Methods

According to the previous study [25], it has been shown that disADMM was faster than several existing distributed methods, such as parallel stochastic gradient descent [44] and Vowpal Wabbit.[2] Also, note that disADMM was designed based on the framework of the conventional distributed ADMM, while in this paper, we focus on improving the convergence speed of the conventional distributed ADMM, and thus, our GADMM is also designed based on the same framework, but GADMM has a group layer to accelerate the convergence speed. Therefore, to evaluate the performance of our proposed method, as well as to compare with disADMM, we adopt the number of outer iterations, total time cost, communication time cost, running time cost, and accuracy as the evaluation metrics. To compare the time costs of GADMM with disADMM, the experiments are conducted repeatedly for ten times. The total ten times results are generated, and also evaluated by their means of the evaluation metrics. Also, the paired t-tests between GADMM and disADMM are further conducted from the evaluation indices on respective four datasets. The details of the comparison are described as follows.

1) *disADMM [25]:* The method was based on the framework of ADMM for distributed linear classification.
2) *GADMM[3]:* A group layer is added into GADMM and the L method is adopted to determine a suitable number of groups in GADMM.

1) Number of Outer Iterations: Table II shows that GADMM converges and meets the stop criteria faster than disADMM, except with a little acceptable accuracy loss, which is in agreement with our expected results as analyzed in Theorem 1. Comparing GADMM with disADMM, the number of outer iterations of GADMM is significantly smaller than that of disADMM on four datasets. Therefore, GADMM performs better than disADMM in terms of lower number of outer iterations, and thus GADMM can improve the convergence speed. The reason is because all slave nodes in the

[1]The datasets are available at http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary.html.

[2]Vowpal Wabbit is a fast online learning algorithm at https://github.com/JohnLangford/vowpal_wabbit.

[3]Due to the randomness of the data division, we simply repeat the grouping method in GADMM ten times, and use a voting strategy to select the group number.

TABLE II
NUMBER OF OUTER ITERATIONS, TTIME, CTIME, RTIME AND ACCURACY OF GADMM COMPARED WITH DISADMM

Dataset	GADMM					disADMM				
	Iter	Ttime(s)	Ctime(s)	Rtime(s)	Acc(%)	Iter	Ttime(s)	Ctime(s)	Rtime(s)	Acc(%)
webspam	**5.8**	**561.4**	**505.6**	**55.8**	99.26(0.03)	24.4	861.1	703	158.1	**99.57(0.02)**
epsilon	**32.4**	**30.4**	**4.6**	**25.8**	89.61(0.09)	73.2	50.0	9.5	40.5	**89.81(0.08)**
rcv1	**10.8**	**4.5**	**2.7**	**1.8**	97.60(0.02)	44.7	8.5	4.8	3.7	**97.82(0.03)**
kdd2010	**8.4**	**1506**	**1347.2**	**158.8**	87.81(0.02)	31.0	2174.5	1659.3	515.2	**88.41(0.02)**

computing cluster are divided into several groups through model similarity-based grouping methods, and then GADMM generates the group variables to update the global variable by the weighted averaged method. Finally, the master node broadcasts the group variables to the slave nodes, for reupdating the local variables on slave nodes in the same group, respectively. Compared with disADMM, fewer different variables need to be in consensus with the global variable until the solution to the global problem is found in GADMM.

2) Time Cost and Accuracy: In the experiment, the total time cost is the sum of the communication time cost and running time cost. The experimental results on most large-scale datatsets show that the time cost of GADMM can be up to 30% leaa than that of disADMM, but with less than 0.6% accuracy loss, e.g., webspam and kdd2010 datasets. In Table II, we find that the communication time cost is closely related to the dimension of datasets. The communication time costs in each outer iteration of datasets kdd2010 and webspam are much larger than those of lower-dimension datasets, such as epsilon and rcv1. Also, in the optimization of subproblems, the DCD needs to train all the examples in each inner iteration. Therefore, the running time cost increases with the increase in the size of datasets. For the dataset epsilon, whose number of examples is much larger than the dimension of feature, most of the total time cost is taken in optimizing the subproblems. Also, we can find that the communication time cost on epsilon in each outer iteration is smaller than the higher-dimension dataset rcv1. For the high dimensional and large-scale datasets webspam and kdd2010, the results show that the total time cost and communication time cost of GADMM are significantly reduced compared with that of disADMM. In Table II, we find that GADMM can reduce the total time cost and communication time cost significantly, compared with disADMM on large-scale experimental datasets.

In this paper, we adopt GADMM to solve the global consensus problem over variables $\mathbf{w}_i$ and $\mathbf{z}$. In Algorithm 1, the optimization of subproblems is implemented on slave nodes in parallel, Thus, the only time cost is the running time, as no communication time cost is involved. Therefore, the complexity per outer iteration of the running time cost is $O(\#nz)$ based on the DCD, where $\#nz$ is the total number of nonzero values of training examples. In GADMM, only the processes of sending the local and dual variables to the master node, and broadcasting the global variable and group variables to slave nodes, affect the communication time cost. Therefore, the complexity of the communication time cost in GADMM is $O(k_{\text{out}}d)$, where k_{out} is the number of outer iterations. In each outer iteration, the communication time cost is closely related to the dimension of datasets, which is in agreement

TABLE III
STATISTICAL TEST OF GADMM COMPARED WITH DISADMM. √ INDICATES OUR METHOD IS STATISTICALLY BETTER THAN DISADMM

Dataset	GADMM compared with disADMM					
	Iter	Ttime	Ctime	Rtime	Lacc	STime
webspam	√	√	√	√	√	√
epsilon	√	√	√	√	√	√
rcv1	√	√	√	√	√	√
kdd2010	√	√	√	√	√	√

with the communication time cost results in Table II, while the complexity of the communication time cost in disADMM is also $O(k'_{\text{out}}d)$, where k'_{out} is the number of outer iterations. The theoretical analysis and experimental results on large-scale datasets have demonstrated that GADMM can significantly reduce the number of outer iterations, converge much faster than distributed ADMM. Therefore, k_{out} is much smaller than k'_{out}, GADMM can reduce the communication time cost and improve the convergence speed, which is in good agreement with the experimental results in Table II.

3) Statistical Hypothesis Test: To statistically validate the superiority in the convergence speed of the proposed GADMM, the paired t-tests comparing GADMM with disADMM are computed from the four evaluation indices: the number of outer iterations, total time cost, communication time cost and running time cost, respectively, (see Table III). For accuracy loss and speed-up compared with disADMM, to validate the experimental results of GADMM in Table II, we adopt the right one-sided t-test between GADMM and disADMM with 0.6% accuracy loss (Lacc) and left one-sided t-test between GADMM and disADMM with 30% speed-up (STime), respectively. When the P-value is less than 0.05, our method is significantly different from (better than) disADMM. For simplification, we used the "√" to indicate the value where our method is statistically better than disADMM. The GADMM method can significantly reduce the number of outer iterations, total time cost, communication time cost and running time cost, as seen in Table III, saving up to 30% of the total time cost with less than 0.6% accuracy loss on large-scale datasets. Therefore, the hypothesis test shows that GADMM can achieve significant improvement in convergence speed and reducing the time cost, which is faster than disADMM with a little acceptable accuracy loss (i.e., less than 0.6% accuracy loss) for handling large-scale datasets in the results.

To further test the scalability of the proposed GADMM method, we extend the SDCA-ADMM [32] to the framework of GADMM (namely SDCA-GADMM), and make a comparison with SDCA-ADMM under the same distributed cluster.

TABLE IV
NUMBER OF OUTER ITERATIONS, TTIME AND ACCURACY OF SDCA-GADMM COMPARED WITH SDCA-ADMM

Dataset	SDCA-GADMM			SDCA-ADMM		
	Iter	Ttime(s)	Acc(%)	Iter	Ttime(s)	Acc(%)
epsilon	**33.3**	**14.1**	89.63	76.5	22.3	**89.73**
kdd2010	**10.2**	**1629.5**	87.84	34.6	1973.8	**88.44**
rcv1	**13.4**	**3.8**	97.60	47.3	6.1	**97.79**
webspam	**9.3**	**842.7**	99.28	30.6	1063.8	**99.31**

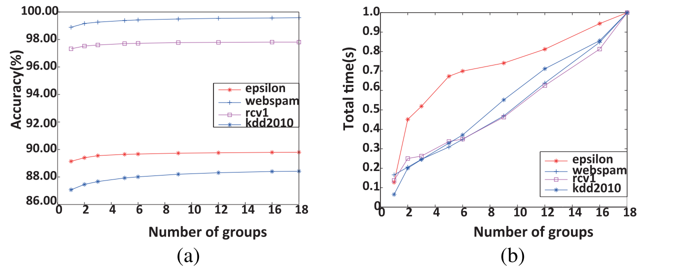

Fig. 3. (a) Accuracy and (b) relative total time cost on each dataset with different groups cases. The longest total time cost on each dataset is set to 1, other total time costs with different groups are scaled down.

In each iteration, we randomly choose one tenth of examples for training. The experiments are conducted repeatedly for ten times, and the means of the number of outer iterations, total time cost and accuracy are used to evaluate the performances. The experimental results in Table IV show that SDCA-GADMM can converge faster to reach the global consensus, and also reduce the total time cost compared with SDCA-ADMM in an acceptable accuracy loss.

C. Evaluation of GADMM With Different Group Cases

To validate the performance of GADMM with different group cases divided by AGNES [34], we adopt five-fold cross-validation to evaluate the experimental results using the means of the number of outer iterations, total time costs and accuracies. The accuracies and relative total time costs on four large-scale datasets are shown in Fig. 3. As the number of groups increases, GADMM can improve accuracy with the gradual increase of the total time cost on all the datasets. The total time cost curves on datasets webspam, rcv1 and kdd2010 are approximately linear, while the curve on epsilon fluctuates slightly. The possible reason is that the feature density is 100%, and all the examples need to be trained in each inner iteration. Therefore, most of the total time cost on epsilon is taken on the slave nodes, and the tiny difference in the numbers of outer iterations makes the total time cost change greatly (one outer iteration includes 50 inner iterations). Also, the number of examples and dimension of a dataset also can affect the total time cost. Table V shows the classification results of GADMM with different group cases on four large-scale datasets. As the number of groups increases, the accuracies on all the datasets improve, while the number of outer iterations also increases. The total time cost is increased proportional to the number of outer iterations. Also, in Fig. 3(a) and Table V, we find that the improvement in accuracy gradually decreases as the number of groups increases. The suitable interval of group numbers based on the better improvement in accuracy is [3, 4, 5]. The experimental results with different group cases chosen by GADMM-*L* are in bold. The grouping results show that the *L* method can efficiently determine a suitable number of groups by finding the knee of an evaluation graph curve.

TABLE V
NUMBER OF OUTER ITERATIONS AND ACCURACIES ON FOUR DATASET, WHERE NUM IS THE NUMBER OF GROUPS. THE RESULTS WITH DIFFERENT GROUP CASES CHOSEN BY GADMM-L ARE IN BOLD

Num	webspam	epsilon	rcv1	kdd2010
	Iter/Acc(%)	Iter/Acc(%)	Iter/Acc(%)	Iter/Acc(%)
1	4.0/98.89	6.0/89.14	5.0/97.32	3.0/87.07
3	**5.8/99.26**	25.8/89.55	**10.8/97.60**	7.4/87.67
4	6.6/99.34	**32.4/89.60**	11.8/97.67	**8.4/87.82**
5	7.6/99.38	39.0/89.65	13.4/97.70	10.0/87.93
9	11.6/99.49	46.4/89.73	20.2/97.77	16.0/88.17
12	15.8/99.54	53.6/89.76	27.0/97.79	21.0/88.30
15	20.0/99.55	61.2/89.79	34.4/97.80	26.0/88.38
18	24.8/99.58	72.6/89.80	44.4/97.81	31.0/88.41

TABLE VI
STATISTICAL TEST *P*-VALUE ON THE NUMBER OF OUTER ITERATIONS AND THE TOTAL TIME COST WITH DIFFERENT ACCURACY LOSSES. √ INDICATES OUR METHOD IS SIGNIFICANTLY BETTER THAN DISADMM (BLANK INDICATES THAT THE MAXIMUM ACCURACY LOSS BETWEEN DIFFERENT GROUP CASES ON EACH DATASET IS LESS THAN 1.0%.)

Accuracy loss	webspam	epsilon	rcv1	kdd2010
	Iter/Ttime	Iter/Ttime	Iter/Ttime	Iter/Ttime
0.2%	√/√	√/√	√/√	√/0.96
0.4%	√/√	√/√	√/√	√/√
0.6%	√/√	√/√	√/√	√/√
0.8%	√/√	√/√	√/√	√/√
1.0%				√/√

The experimental results also show that there is a trade-off between the total time cost and accuracy in dealing with large-scale datasets. In general, the total time cost is inversely proportional to classification accuracy. In this paper, we focus on improving the convergence speed and saving the total time cost, with an acceptable accuracy loss for the large-scale classification problem. To improve the reliability of the classification results, we statistically validate the superiority in the number of outer iterations and total time cost of GADMM compared with that of disADMM with different degrees of loss in classification accuracy. The paired *t*-test values of GADMM with disADMM are compared by two evaluation indices: the number of outer iterations and total time cost. Table VI shows GADMM can significantly reduce the number of outer iterations, and save the total time cost compared with disADMM on most large-scale datasets.

Moreover, several model similarity-based grouping methods are introduced and investigated to divide slave nodes into groups in different ways. The means of the number of outer iterations and accuracy with five-fold cross-validation are used to further validate the robustness of GADMM. The details of the comparison grouping methods are described as follows.

1) *GADMM-SL:* Single-linkage clustering is adopted to divide slave nodes into groups in GADMM.
2) *GADMM-AL:* The distance between two clusters is defined as the average distance between each point in one cluster to every point in the other cluster in Average-linkage clustering.

TABLE VII
NUMBER OF OUTER ITERATIONS AND ACCURACY OF GADMM COMPARED WITH DIFFERENT OPTIMIZATION METHODS

	Methods	epsilon	kdd2010	rcv1	webspam
		Iter/Acc(%)	Iter/Acc(%)	Iter/Acc(%)	Iter/Acc(%)
G_3	GADMM-AL	28.2/89.53	8.0/87.69	14.4/97.56	7.0/99.25
	GADMM-SL	15.8/89.51	7.4/87.69	9.8/97.59	5.2/99.23
G_4	GADMM-AL	36.4/89.57	9.2/87.81	15.2/97.62	8.2/99.32
	GADMM-SL	21.8/89.54	8.2/87.83	10.4/97.65	6.2/99.31
G_5	GADMM-AL	39.0/89.64	10.2/87.94	15.2/97.66	9.0/99.38
	GADMM-SL	26.4/89.62	10.6/87.92	12.2/97.71	7.2/99.35

TABLE VIII
NUMBER OF OUTER ITERATIONS, TTIME, CTIME, AND ACCURACY OF GADMM COMPARED WITH DIFFERENT OPTIMIZATION METHODS

Methods		epsilon			webspam		
		Iter/Ttime(s)	Ctime(s)	Acc(%)	Iter/Ttime(s)	Ctime(s)	Acc(%)
G_3	DCD	**29.8/27.6**	**1.8**	89.55(0.09)	**5.8/560.6**	**502.4**	99.26(0.03)
	TRON	43.4/199.6	79.2	**89.78**(0.11)	6.2/1306.4	695.2	**99.34**(0.03)
G_4	DCD	**32.4/30.2**	**4.6**	89.60(0.10)	**6.6/645.0**	**587.8**	99.34(0.02)
	TRON	56.8/266.2	88.6	**89.81**(0.12)	7.0/1444.0	719.4	**99.40**(0.03)
G_5	DCD	**39.0/35.8**	**5.2**	89.65(0.10)	**7.6/741.6**	**674.8**	99.38(0.02)
	TRON	69.0/316.4	52.4	**89.82**(0.10)	8/1624.8	851.2	**99.44**(0.02)

In the experiment, we typically divide slave nodes into three, four, and five groups, because the interval, [3, 4, 5] is the suitable number of groups discussed in Section V-C, denoted as G_3, G_4, and G_5, respectively. The experimental results in Table VII show that although GADMM with different grouping methods may slightly affect the performance and time cost, GADMM can accelerate the convergence speed and reduce the number of outer iterations.

D. Evaluation of GADMM With Different Optimization Methods

To investigate the effect of using different optimization methods on the performance of GADMM, we compare two typical methods: 1) DCD and 2) trust region Newton method (TRON) [45] in GADMM with the same group cases using AGNES [34]. DCD is a dual-based method, which uses only low-order information to optimize the classification problem, while TRON is a primal-based method, which employs high-order information using Newton directions.

To improve the reliability of the classification results, we adopt five-fold cross-validation to evaluate the experimental results using the means of the number of outer iterations, total time cost, communication time cost and accuracy with variance, respectively. The results of GADMM compared with different optimization methods are shown in Table VIII. In the experiment, we also divide slave nodes into three, four, and five groups. The lowest means of the number of outer iterations and total time cost, and the highest mean of accuracy are in bold.

Table VIII shows that TRON can obtain a little better accuracy than that of DCD in GADMM. However, DCD can converge faster to reach the global consensus than TRON. Thus, a lower total time cost and communication time cost are needed in DCD than in TRON. In all experiments, the total time cost is the sum of the communication time cost and running time cost. In Table VIII, we find that the running time of DCD is much shorter than that of TRON in GADMM. The main reason is that the complexity per outer iteration of DCD is $O(\#nz)$, where $\#nz$ is the total number of nonzero values of training examples. The complexity per outer iteration of TRON is $O(l\#nz)$, where l is the number of the conjugate gradient iterations. TRON employs high-order information (e.g., the Hessian matrix) using Newton directions, and the Hessian matrix may be a huge d by d matrix in handling large-scale datasets. Consequently, the time cost per step of TRON is more expensive. Thus, there are some difficulties in applying the Newton methods to solve the distributed classification problem on large-scale datasets, especially sparse datasets. Therefore, taking the time cost into account, we consider that the DCD is suitable to deal with the large-scale classification problem.

VI. CONCLUSION

In this paper, we propose the GADMM for the distributed linear classification task, for seeking faster convergence speed and better global consensus. GADMM casts a large-scale global problem into a number of medium size local subproblems that can be solved by the DCD in parallel. In particular, the slave nodes that have similar local variables are divided into the same group by the model similarity-based grouping methods. In the group layer, the local variables are gathered to generate the group variables, and then the group variables are aggregated to update the global variable, and that process repeats until global convergence. The theoretical analysis demonstrates that GADMM has an $O(1/k)$ convergence rate and converges faster than the conventional distributed ADMM, where k is the number of outer iterations.

For performance evaluation, four publicly available benchmark LIBSVM datasets have been utilized to conduct the experiments for comparing different methods. The experimental results show that GADMM can reduce the number of outer iterations and improve the convergence speed compared with disADMM and SCDA-ADMM. Moreover, it is experimentally validated that GADMM can significantly reduce up to 30% of the total time cost, with less than 0.6% accuracy loss compared with disADMM on large-scale datasets.

Due to its relatively high convergence speed, GADMM is suitable for solving large-scale problems arising in statistics, machine learning, and in particular to distributed nonlinear classification. In our future work, we will further improve the time efficiency of distributed optimization and design advanced optimization strategies.

APPENDIX A

Proof of Lemma 1: The optimality conditions for $\mathbf{w}^+$ and $\mathbf{z}^+$ in (29) can be solved based on their differentiable property

$$\partial f(\mathbf{w}^+) + \boldsymbol{\lambda} + \rho(\mathbf{w}^+ - \mathbf{z}) = 0$$
$$\partial g(\mathbf{z}^+) - \boldsymbol{\lambda} - \rho(\mathbf{w}^+ - \mathbf{z}^+) = 0.$$

Thus, based on the important identity of the conjugate of a convex function, it satisfies

$$\boldsymbol{\lambda}^{\frac{1}{2}} = -\partial f\left(\mathbf{w}^{+}\right) \leftrightarrow \mathbf{w}^{+} = -\bigtriangledown f^{*}\left(\boldsymbol{\lambda}^{\frac{1}{2}}\right)$$
$$\boldsymbol{\lambda}^{+} = \partial g\left(\mathbf{z}^{+}\right) \leftrightarrow \mathbf{z}^{+} = \bigtriangledown g^{*}\left(\boldsymbol{\lambda}^{+}\right).$$

Since Assumption 1 guarantees that both $\bigtriangledown f^{*}(\boldsymbol{\lambda})$ and $\bigtriangledown g^{*}(\boldsymbol{\lambda})$ are Lipschitz continuous. Define $\alpha \geq \delta_f^{-1}$ and $\beta \geq \delta_g^{-1}$, we choose a value of ρ, which meets that $\rho^3 \leq (1/(\alpha\beta^2))$. Thus

$$\begin{aligned}\left\|\boldsymbol{\lambda}^{+} - \boldsymbol{\lambda}^{\frac{1}{2}}\right\|^2 &= \rho^2\left\|\mathbf{z} - \mathbf{z}^{+}\right\|^2\\ &= \rho^2\left\|\bigtriangledown g^{*}\left(\boldsymbol{\lambda}^{+}\right) - \bigtriangledown g^{*}(\boldsymbol{\lambda})\right\|^2\\ &\leq \rho^2\beta^2\left\|\boldsymbol{\lambda}^{+} - \boldsymbol{\lambda}\right\|^2.\end{aligned}$$

We now derive some estimates for $f(\boldsymbol{\lambda})$ and $g(\boldsymbol{\lambda})$

$$\begin{aligned}f^{*}(\boldsymbol{\gamma}) - f^{*}\left(\boldsymbol{\lambda}^{+}\right) &= f^{*}(\boldsymbol{\gamma}) - f^{*}\left(\boldsymbol{\gamma}^{\frac{1}{2}}\right) + f^{*}\left(\boldsymbol{\gamma}^{\frac{1}{2}}\right) - f^{*}\left(\boldsymbol{\lambda}^{+}\right)\\ &\geq \left(\boldsymbol{\gamma} - \boldsymbol{\lambda}^{\frac{1}{2}}\right)\bigtriangledown f^{*}\left(\boldsymbol{\lambda}^{\frac{1}{2}}\right) - \frac{\alpha}{2}\left\|\boldsymbol{\lambda}^{+} - \boldsymbol{\lambda}^{\frac{1}{2}}\right\|^2\\ &\quad - \left(\boldsymbol{\lambda}^{+} - \boldsymbol{\lambda}^{\frac{1}{2}}\right)\bigtriangledown f^{*}\left(\boldsymbol{\lambda}^{\frac{1}{2}}\right)\\ &\geq \bigtriangledown f^{*}\left(\boldsymbol{\lambda}^{\frac{1}{2}}\right)\left(\boldsymbol{\gamma} - \boldsymbol{\lambda}^{+}\right) - \frac{\alpha\rho^2\beta^3}{2}\left\|\boldsymbol{\lambda}^{+} - \boldsymbol{\lambda}\right\|^2\\ &\geq \bigtriangledown f^{*}\left(\boldsymbol{\lambda}^{\frac{1}{2}}\right)\left(\boldsymbol{\gamma} - \boldsymbol{\lambda}^{+}\right) - \frac{1}{2\rho}\left\|\boldsymbol{\lambda}^{+} - \boldsymbol{\lambda}\right\|^2.\end{aligned}$$

In the same way, it also has

$$g^{*}(\boldsymbol{\gamma}) - g^{*}\left(\boldsymbol{\lambda}^{+}\right) \geq \left(\boldsymbol{\gamma} - \boldsymbol{\lambda}^{+}\right)\bigtriangledown g^{*}\left(\boldsymbol{\lambda}^{+}\right).$$

Since $D(\boldsymbol{\lambda}) = -f^{*}(\boldsymbol{\lambda}) - g^{*}(\boldsymbol{\lambda})$, and thus

$$\begin{aligned}D\left(\boldsymbol{\lambda}^{+}\right) - D(\boldsymbol{\gamma}) &= f^{*}(\boldsymbol{\gamma}) - f^{*}\left(\boldsymbol{\lambda}^{+}\right) + g^{*}(\boldsymbol{\gamma}) - g^{*}\left(\boldsymbol{\lambda}^{+}\right)\\ &\geq \bigtriangledown f^{*}\left(\boldsymbol{\lambda}^{\frac{1}{2}}\right)\left(\boldsymbol{\gamma} - \boldsymbol{\lambda}^{+}\right) - \frac{1}{2\rho}\left\|\boldsymbol{\lambda}^{+} - \boldsymbol{\lambda}\right\|^2\\ &\quad + \bigtriangledown g^{*}\left(\boldsymbol{\lambda}^{+}\right)\left(\boldsymbol{\gamma} - \boldsymbol{\lambda}^{+}\right)\\ &= \left(\boldsymbol{\gamma} - \boldsymbol{\lambda}^{+}\right)\left(\bigtriangledown f^{*}\left(\boldsymbol{\lambda}^{\frac{1}{2}}\right) + \bigtriangledown g^{*}\left(\boldsymbol{\lambda}^{+}\right)\right)\\ &\quad - \frac{1}{2\rho}\left\|\boldsymbol{\lambda}^{+} - \boldsymbol{\lambda}\right\|^2\\ &= \frac{1}{\rho}(\boldsymbol{\gamma} - \boldsymbol{\lambda})\left(\boldsymbol{\lambda} - \boldsymbol{\lambda}^{+}\right) + \frac{1}{2\rho}\left\|\boldsymbol{\lambda}^{+} - \boldsymbol{\lambda}\right\|^2. \quad (32)\end{aligned}$$

■

Appendix B

Proof of Theorem 1: We set $\boldsymbol{\gamma} = \boldsymbol{\lambda}^{*}$, $\boldsymbol{\lambda} = \boldsymbol{\lambda}^{k}$ and $\boldsymbol{\lambda}^{+} = \boldsymbol{\lambda}^{k+1}$ in Lemma 1. Therefore, $D(\boldsymbol{\lambda}^{k+1}) - D(\boldsymbol{\lambda}^{*})$ satisfies

$$\begin{aligned}2\rho\left(D\left(\boldsymbol{\lambda}^{k+1}\right) - D\left(\boldsymbol{\lambda}^{*}\right)\right) &\geq 2\left(\boldsymbol{\lambda}^{k} - \boldsymbol{\lambda}^{*}\right)\left(\boldsymbol{\lambda}^{k+1} - \boldsymbol{\lambda}^{k}\right)\\ &\quad + \left\|\boldsymbol{\lambda}^{k+1} - \boldsymbol{\lambda}^{k}\right\|^2\\ &= \left\|\boldsymbol{\lambda}^{*} - \boldsymbol{\lambda}^{k+1}\right\|^2 - \left\|\boldsymbol{\lambda}^{*} - \boldsymbol{\lambda}^{k}\right\|^2. \quad (33)\end{aligned}$$

We can find that the difference between $D(\boldsymbol{\lambda}^{k+1})$ and $D(\boldsymbol{\lambda}^{*})$ is related to $\boldsymbol{\lambda}^{k}$, $\boldsymbol{\lambda}^{k+1}$, and $\boldsymbol{\lambda}^{*}$. Therefore, summing the differences between $D(\boldsymbol{\lambda}^{k})$ and $D(\boldsymbol{\lambda}^{*})$ over $k = 1, 2, \ldots, n-1$, it also satisfies

$$\begin{aligned}&2\rho\left(\sum_{k=1}^{n-1} D\left(\boldsymbol{\lambda}^{k+1}\right) - (n-1)D\left(\boldsymbol{\lambda}^{*}\right)\right)\\ &\qquad \geq \left\|\boldsymbol{\lambda}^{*} - \boldsymbol{\lambda}^{n}\right\|^2 - \left\|\boldsymbol{\lambda}^{*} - \boldsymbol{\lambda}^{1}\right\|^2. \quad (34)\end{aligned}$$

In the same way, we set $\boldsymbol{\gamma} = \boldsymbol{\lambda} = \boldsymbol{\lambda}^{k}$ and $\boldsymbol{\lambda}^{+} = \boldsymbol{\lambda}^{k+1}$ in Lemma 1, the difference between $D(\boldsymbol{\lambda}^{k+1})$ and $D(\boldsymbol{\lambda}^{k})$ is

$$2\rho\left(D\left(\boldsymbol{\lambda}^{k+1}\right) - D\left(\boldsymbol{\lambda}^{k}\right)\right) \geq \left\|\boldsymbol{\lambda}^{k} - \boldsymbol{\lambda}^{k+1}\right\|^2. \quad (35)$$

We multiply this estimate in (35) by $k-1$, and then sum this estimate in (35) over $k = 1, 2, \ldots, n-1$. Thus, the final result can be expressed as

$$\begin{aligned}&2\rho\sum_{k=1}^{n-1}\left(kD\left(\boldsymbol{\lambda}^{k+1}\right) - D\left(\boldsymbol{\lambda}^{k+1}\right) - (k-1)D\left(\boldsymbol{\lambda}^{k}\right)\right)\\ &\qquad \geq \sum_{k=1}^{n-1} k\left\|\boldsymbol{\lambda}^{k} - \boldsymbol{\lambda}^{k+1}\right\|^2. \quad (36)\end{aligned}$$

The telescoping sum on the left in (36) can be reduced to

$$2\rho\left((n-1)D\left(\boldsymbol{\lambda}^{n}\right) - \sum_{k=1}^{n-1} D\left(\boldsymbol{\lambda}^{k+1}\right)\right) \geq \sum_{k=1}^{n-1} k\left\|\boldsymbol{\lambda}^{k} - \boldsymbol{\lambda}^{k+1}\right\|^2. \quad (37)$$

Then, we add this estimate in (35) and that in (37), and find the difference between $D(\boldsymbol{\lambda}^{n})$ and $D(\boldsymbol{\lambda}^{*})$ satisfies

$$\begin{aligned}2\rho(n-1)\left(D\left(\boldsymbol{\lambda}^{n}\right) - D\left(\boldsymbol{\lambda}^{*}\right)\right) &\geq \left\|\boldsymbol{\lambda}^{*} - \boldsymbol{\lambda}^{n}\right\|^2 - \left\|\boldsymbol{\lambda}^{*} - \boldsymbol{\lambda}^{1}\right\|^2\\ &\quad + \sum_{k=1}^{n-1} k\left\|\boldsymbol{\lambda}^{k} - \boldsymbol{\lambda}^{k+1}\right\|^2.\end{aligned}$$

Therefore, the convergence bound in the kth iteration can be mathematically formulated as

$$D\left(\boldsymbol{\lambda}^{*}\right) - D\left(\boldsymbol{\lambda}^{k}\right) \leq \frac{\left\|\boldsymbol{\lambda}^{*} - \boldsymbol{\lambda}^{1}\right\|^2 - \left\|\boldsymbol{\lambda}^{*} - \boldsymbol{\lambda}^{k}\right\|^2}{2\rho(k-1)}.$$

■

Appendix C

Proof of Lemma 2: In Section IV-B, function $F(\mathbf{w})$ is defined as

$$F(\mathbf{w}) = f(\mathbf{w}) + h(\mathbf{w})$$

where $h(\mathbf{w}) = (\rho/2)\|\mathbf{w} - \mathbf{z} + \mathbf{u}\|^2$. Function $f(\mathbf{w})$ is the hinge loss function and a non-negative function. If there are only two slave nodes, S_a and S_b, in a group, $\mathbf{w}_c$ is the average of $\mathbf{w}_a$ and $\mathbf{w}_b$. Since function $f(\mathbf{w})$ is a convex function, it satisfies

$$f(\mathbf{w}_a) + f(\mathbf{w}_b) \geq f(\mathbf{w}_c).$$

According to the update procedures in (16) and (19), when $k = 2$, the difference is

$$\begin{aligned}F\left(\mathbf{w}_a^2\right) &+ F\left(\mathbf{w}_b^2\right) - 2F\left(\mathbf{w}_c^2\right)\\ &\geq \frac{\rho}{2}\left\|\mathbf{w}_a^2 - \mathbf{z}^1 + \mathbf{u}_a^1\right\|^2 + \frac{\rho}{2}\left\|\mathbf{w}_b^2 - \mathbf{z}^1 + \mathbf{u}_b^1\right\|^2\\ &\quad - \rho\left\|\mathbf{w}_c^2 - \mathbf{z}^1 + \mathbf{u}_c^1\right\|^2\\ &\geq \frac{\rho}{2}\left\|\mathbf{w}_a^2 + \mathbf{w}_a^1 - 2\mathbf{z}^1\right\|^2 + \frac{\rho}{2}\left\|\mathbf{w}_b^2 + \mathbf{w}_b^1 - 2\mathbf{z}^1\right\|^2\\ &\quad - \frac{\rho}{4}\left\|\mathbf{w}_a^2 + \mathbf{w}_a^1 + \mathbf{w}_b^2 + \mathbf{w}_b^1 - 4\mathbf{z}^1\right\|^2\\ &\geq 0.\end{aligned}$$

By the same analogy, when $k \geq 2$, the difference in the kth iteration satisfies

$$\begin{aligned}F\left(\mathbf{w}_a^k\right) &+ F\left(\mathbf{w}_b^k\right) - 2F\left(\mathbf{w}_c^k\right)\\ &\geq \frac{\rho}{2}\left\|\sum_{i=1}^{k}\mathbf{w}_a^i - \overline{z}^k\right\|^2 + \frac{\rho}{2}\left\|\sum_{i=1}^{k}\mathbf{w}_b^i - \overline{z}^k\right\|^2\\ &\quad - \rho\left\|\frac{\sum_{i=1}^{k}\left(\mathbf{w}_a^i + \mathbf{w}_b^i\right)}{2} - \overline{z}^k\right\|^2\\ &\geq \frac{\rho}{2}\left\|\sum_{i=1}^{k}\mathbf{w}_a^i - \overline{z}^k\right\|^2 + \frac{\rho}{2}\left\|\sum_{i=1}^{k}\mathbf{w}_b^i - \overline{z}^k\right\|^2\\ &\quad - \frac{\rho}{4}\left\|\sum_{i=1}^{k}\left(\mathbf{w}_a^i + \mathbf{w}_b^i\right) - 4\overline{z}^k\right\|^2\\ &\geq 0\end{aligned}$$

where $\overline{z}^k = \sum_{i=1}^{k-1}\mathbf{z}^i + \mathbf{z}^{k-1}$. Moreover, if there are s slave nodes in a group, the difference in the kth iteration satisfies

$$\begin{aligned}\sum_{j=1}^{s}F\left(\mathbf{w}_j^k\right) &- sF\left(\frac{1}{s}\sum_{j=1}^{s}\mathbf{w}_j^k\right)\\ &\geq \frac{\rho}{2}\sum_{j=1}^{s}\left\|\sum_{i=1}^{k}\mathbf{w}_j^i - \overline{z}^k\right\|^2 - \frac{\rho}{2s}\left\|\sum_{i=1}^{k}\sum_{j=1}^{s}\mathbf{w}_j^i - s\overline{z}^k\right\|^2 \geq 0.\end{aligned}$$

When slave nodes are divided into m groups, the difference is

$$\begin{aligned}\sum_{j=1}^{n}F\left(\mathbf{w}_i^k\right) - \sum_{j=1}^{m}n_jF\left(\frac{1}{n_j}\sum_{S_i\in G_j}\mathbf{w}_i^k\right) &\geq \frac{\rho}{2}\sum_{j=1}^{n}\left\|\sum_{i=1}^{k}\mathbf{w}_i^l - \overline{z}^k\right\|^2\\ &-\frac{\rho}{2n_j}\sum_{j=1}^{m}\left\|\sum_{i=1}^{k}\sum_{S_i\in G_j}\mathbf{w}_j^i - n_j\overline{z}^k\right\|^2 \geq 0.\end{aligned} \tag{38}$$

Since $F(\mathbf{w})$ is a non-negative and convex function, it satisfies

$$\begin{aligned}\sum_{j=1}^{n}F\left(\mathbf{w}_i^k\right) - n_j\sum_{j=1}^{m}F\left(\frac{\sum_{S_i\in G_j}\mathbf{w}_i^k}{n_j}\right) &\geq \sum_{j=1}^{n}F\left(\mathbf{w}_i^k\right)\\ &-\sum_{j=1}^{m}\sum_{S_i\in G_j}F\left(\mathbf{w}_i^k\right).\end{aligned}$$

In the distributed classification problem, the data is randomly split into n disjoint datasets. $\beta_{i,j}$, the weight of slave node S_i in group G_j, can be considered similar to those of other slave nodes in the same group. Therefore, the value of $\beta_{i,j}$ can be regarded as the reciprocal of the number of slave nodes in group G_j. The total loss satisfies

$$sF\left(\sum_{i=1}^{s}\beta_{i,j}\mathbf{w}_i^k\right) \leq sF\left(\frac{1}{s}\sum_{i=1}^{s}\mathbf{w}_i^k\right).$$

By the same analogy, when all the n slave nodes are divided into m groups in the distributed computing cluster, the difference in the kth iteration is

$$\sum_{i=1}^{n}F\left(\mathbf{w}_i^k\right) \geq \sum_{j=1}^{m}n_jF\left(\sum_{S_i\in G_j}\beta_{i,j}\mathbf{w}_i^k\right).$$

■

Appendix D

Proof of Lemma 3: Since function $f_i(\mathbf{w}_i)$ and $g(\mathbf{z})$ are non-negative and convex functions, if there are s slave nodes in a group, in the kth iteration, it satisfies

$$\sum_{i=1}^{s}f_i\left(\mathbf{w}_i^k\right) \geq sf_i\left(\frac{1}{s}\sum_{j=1}^{s}\mathbf{w}_j^k\right).$$

Thus, if there are n slave nodes in m groups, based on the average value inequality, it still satisfies

$$\sum_{i=1}^{s}f_i\left(\mathbf{w}_i^k\right) \geq \sum_{j=1}^{m}n_jf_i\left(\frac{1}{m_j}\sum_{S_i\in G_j}\mathbf{w}_i^k\right) \geq \sum_{j=1}^{m}n_jf_i\left(\sum_{S_i\in G_j}\beta_{i,j}\mathbf{w}_i^k\right)$$

where n_j is the number of slave nodes in group G_j. Since global variable $\mathbf{z}^{k+1}$ is differentiable, thus it can be updated

$$g\left(\mathbf{z}^{k+1}\right) = g\left(\frac{\rho}{1+n\rho}\sum_{j=1}^{n}\left(\mathbf{w}_i^{k+1} + \mathbf{u}_i^k\right)\right).$$

According to the updating of the global variable of GADMM in (28), we can find that

$$\begin{aligned}\frac{m\rho}{1+m\rho}\sum_{j=1}^{m}\frac{n_j}{n}&\left(\mathbf{w}_{Gj}^{k+1} + \mathbf{u}_{Gj}^k\right)\\ &= \frac{m\rho}{n(1+m\rho)}\sum_{j=1}^{m}n_j\left(\mathbf{w}_{Gj}^{k+1} + \mathbf{u}_{Gj}^k\right)\\ &\leq \frac{m\rho}{n(1+m\rho)}\sum_{j=1}^{m}\sum_{S_i\in G_j}\left(\mathbf{w}_i^{k+1} + \mathbf{u}_i^k\right)\\ &\leq \frac{m\rho}{n(1+m\rho)}\sum_{i=1}^{n}\left(\mathbf{w}_i^{k+1} + \mathbf{u}_i^k\right).\end{aligned}$$

Moreover

$$\frac{\rho}{1+n\rho}\mathbf{v}_i^{k+1} - \frac{m\rho}{n(1+m\rho)}\mathbf{v}_i^{k+1} = \frac{\rho(n-m)}{n(1+m\rho)(1+n\rho)}\mathbf{v}_i^{k+1}$$

where $\mathbf{v}_i^{k+1} = \sum_{j=1}^{n}(\mathbf{w}_i^{k+1} + \mathbf{u}_i^k)$. When $m = n$, GADMM is the same as the conventional distributed ADMM, while $m < n$, since $g(\mathbf{z}) = (1/2)\|\mathbf{z}\|^2$, thus

$$\begin{aligned} g\left(\frac{\rho}{1+n\rho}\mathbf{v}_i^{k+1}\right) &> g\left(\frac{m\rho}{n(1+m\rho)}\mathbf{v}_i^{k+1}\right) \\ &\geq g\left(\frac{m\rho}{n(1+m\rho)}\overline{\mathbf{v}_i^{k+1}}\right) \end{aligned}$$

where $\overline{\mathbf{v}_i^{k+1}} = \sum_{j=1}^{m} n_j(\mathbf{w}_{Gj}^{k+1} + \mathbf{u}_{Gj}^k)$. Therefore, we can find

$$\begin{aligned} G\left(\mathbf{w}^k, \mathbf{z}^k\right) - \overline{G}\left(\overline{\mathbf{w}^k}, \overline{\mathbf{z}^k}\right) &\geq g\left(\frac{\rho\mathbf{v}_i^{k+1}}{1+n\rho}\right) - g\left(\frac{m\rho\mathbf{v}_i^{k+1}}{n(1+m\rho)}\right) \\ &= \frac{m\rho\left(\mathbf{v}_i^{k+1}\right)^T}{n(1+m\rho)}\left(\frac{\rho\mathbf{v}_i^{k+1}}{(1+n\rho)} - \frac{m\rho\mathbf{v}_i^{k+1}}{n(1+m\rho)}\right) \\ &= \frac{\rho^2 m(n-m)}{n^2(1+m\rho)^2(1+n\rho)}\left\|\mathbf{v}_i^{k+1}\right\|^2. \end{aligned}$$

■

References

[1] Y. Pang, K. Zhang, Y. Yuan, and K. Wang, “Distributed object detection with linear SVMs,” *IEEE Trans. Cybern.*, vol. 44, no. 11, pp. 2122–2133, Nov. 2014.

[2] G.-X. Yuan, C.-H. Ho, and C.-J. Lin, “Recent advances of large-scale linear classification,” *Proc. IEEE*, vol. 100, no. 9, pp. 2584–2603, Sep. 2012.

[3] X. Chen, Y. Gao, and R. Wang, “Online selective kernel-based temporal difference learning,” *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 24, no. 12, pp. 1944–1956, Dec. 2013.

[4] H. Chen *et al.*, “Error analysis of stochastic gradient descent ranking,” *IEEE Trans. Cybern.*, vol. 43, no. 3, pp. 898–909, Jun. 2013.

[5] X. Gao, X. Wang, D. Tao, and X. Li, “Supervised Gaussian process latent variable model for dimensionality reduction,” *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 41, no. 2, pp. 425–434, Apr. 2011.

[6] G. Lan, A. Nemirovski, and A. Shapiro, “Validation analysis of mirror descent stochastic approximation method,” *Math. Program.*, vol. 134, no. 2, pp. 425–458, 2012.

[7] L. Xiao, “Dual averaging methods for regularized stochastic learning and online optimization,” *J. Mach. Learn. Res.*, vol. 11, pp. 2543–2596, Oct. 2010.

[8] H.-F. Yu, C.-J. Hsieh, K.-W. Chang, and C.-J. Lin, “Large linear classification when data cannot fit in memory,” in *Proc. 16th ACM SIGKDD Conf. Knowl. Discov. Data. Mining*, Washington, DC, USA, Jul. 2010, pp. 833–842.

[9] Z. A. Zhu, W. Chen, G. Wang, C. Zhu, and Z. Chen, “P-packSVM: Parallel primal gradient descent kernel SVM,” in *Proc. 9th Int. Conf. Data Mining*, Miami, FL, USA, Dec. 2009, pp. 677–686.

[10] C. H. Teo, S. Vishwanthan, A. J. Smola, and Q. V. Le, “Bundle methods for regularized risk minimization,” *J. Mach. Learn. Res.*, vol. 11, pp. 311–365, Jan. 2010.

[11] Z.-G. Hou, L. Cheng, and M. Tan, “Multicriteria optimization for coordination of redundant robots using a dual neural network,” *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 40, no. 4, pp. 1075–1087, Aug. 2010.

[12] T. Suzuki, “Dual averaging and proximal gradient descent for online alternating direction multiplier method,” in *Proc. 30th Int. Conf. Mach. Learn.*, Atlanta, GA, USA, 2013, pp. 392–400.

[13] H. P. Graf, E. Cosatto, L. Bottou, I. Durdanovic, and V. Vapnik, “Parallel support vector machines: The cascade SVM,” in *Proc. Adv. Neural Inf. Process. Syst.*, Vancouver, BC, Canada, 2004, pp. 521–528.

[14] H. Ouyang, N. He, L. Tran, and A. Gray, “Stochastic alternating direction method of multipliers,” in *Proc. 30th Int. Conf. Mach. Learn.*, Atlanta, GA, USA, 2013, pp. 80–88.

[15] B. He and X. Yuan, “On the o(1/n) convergence rate of the Douglas–Rachford alternating direction method,” *SIAM J. Numer. Anal.*, vol. 50, no. 2, pp. 700–709, 2012.

[16] T. Erseghe, D. Zennaro, E. Dall’Anese, and L. Vangelista, “Fast consensus by the alternating direction multipliers method,” *IEEE Trans. Signal Process.*, vol. 59, no. 11, pp. 5523–5537, Nov. 2011.

[17] A. Mokhtari, W. Shi, Q. Ling, and A. Ribeiro, “Decentralized quadratically approximated alternating direction method of multipliers,” *arXiv preprint arXiv:1510.07356*, 2015.

[18] E. Wei and A. Ozdaglar, “Distributed alternating direction method of multipliers,” in *Proc. 51st IEEE Conf. Decis. Control*, Maui, HI, USA, Dec. 2012, pp. 5445–5450.

[19] C. Xi, Q. Wu, and U. A. Khan, “Distributed gradient descent over directed graphs,” *arXiv preprint arXiv:1510.02146*, 2015.

[20] F. Iutzeler, P. Bianchi, P. Ciblat, and W. Hachem, “Explicit convergence rate of a distributed alternating direction method of multipliers,” *arXiv preprint arXiv:1312.1085*, 2013.

[21] C. Xi and U. A. Khan, “On the linear convergence of distributed optimization over directed graphs,” *arXiv preprint arXiv:1510.02149*, 2015.

[22] L. Vemula, “Communication-efficient distributed optimization for regularized risk minimization,” Ph.D. dissertation, Dept. Comput. Sci. Eng., Indian Inst. Technol. Bombay, Mumbai, India, 2014.

[23] T.-H. Chang, M. Hong, W.-C. Liao, and X. Wang, “Asynchronous distributed ADMM for large-scale optimization—Part I: Algorithm and convergence analysis,” *arXiv preprint arXiv:1509.02597*, 2015.

[24] F. Iutzeler, P. Bianchi, P. Ciblat, and W. Hachem, “Asynchronous distributed optimization using a randomized alternating direction method of multipliers,” in *Proc. 52nd IEEE Conf. Decis. Control*, Florence, Italy, Dec. 2013, pp. 3671–3676.

[25] C. Zhang, H. Lee, and K. G. Shin, “Efficient distributed linear classification algorithms via the alternating direction method of multipliers,” in *Proc. 15th Int. Conf. Artif. Intell. Stat.*, La Palma, Spain, 2012, pp. 1398–1406.

[26] Q. Tao, Q.-K. Gao, D.-J. Chu, and G.-W. Wu, “Stochastic learning via optimizing the variational inequalities,” *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 25, no. 10, pp. 1769–1778, Oct. 2014.

[27] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, “Distributed optimization and statistical learning via the alternating direction method of multipliers,” *Found. Trends Mach. Learn.*, vol. 3, no. 1, pp. 1–122, 2011.

[28] P. A. Forero, A. Cano, and G. B. Giannakis, “Consensus-based distributed support vector machines,” *J. Mach. Learn. Res.*, vol. 11, pp. 1663–1707, May 2010.

[29] J. Li and Y. Chen, “Large-scale supervised hierarchical feature learning for face recognition,” *arXiv preprint arXiv:1407.1490*, 2014.

[30] C.-P. Lee, K.-W. Chang, S. Upadhyay, and D. Roth, “Distributed training of structured SVM,” *arXiv preprint arXiv:1506.02620*, 2015.

[31] Y. Arjevani and O. Shamir, “Communication complexity of distributed convex learning and optimization,” in *Proc. Adv. Neural Inf. Process. Syst.*, Montréal, QC, Canada, Dec. 2015, pp. 1747–1755.

[32] T. Suzuki, “Stochastic dual coordinate ascent with alternating direction method of multipliers,” in *Proc. 31st Int. Conf. Mach. Learn.*, Beijing, China, Jun. 2014, pp. 736–744.

[33] C.-P. Lee and D. Roth, “Distributed box-constrained quadratic optimization for dual linear SVM,” in *Proc. 32nd Int. Conf. Mach. Learn.*, Lille, France, 2015, pp. 987–996.

[34] L. Kaufman and P. J. Rousseeuw, *Finding Groups in Data: An Introduction to Cluster Analysis*. New York, NY, USA: Wiley, 1990.

[35] S. Salvador and P. Chan, “Determining the number of clusters/segments in hierarchical clustering/segmentation algorithms,” in *Proc. 16th Int. Conf. Tools Artif. Intell.*, Boca Raton, FL, USA, Nov. 2004, pp. 576–584.

[36] D. Wang, X. Gao, and X. Wang, “Semi-supervised nonnegative matrix factorization via constraint propagation,” *IEEE Trans. Cybern.*, vol. 46, no. 1, pp. 233–244, Jan. 2016.

[37] H. Z. Luo, X. L. Sun, and D. Li, “On the convergence of augmented Lagrangian methods for constrained global optimization,” *SIAM J. Optim.*, vol. 18, no. 4, pp. 1209–1230, 2007.

[38] D. Yuan, S. Xu, and H. Zhao, “Distributed primal–dual subgradient method for multiagent optimization via consensus algorithms,” *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 41, no. 6, pp. 1715–1724, Dec. 2011.

[39] M. Varewyck and J.-P. Martens, “A practical approach to model selection for support vector machines with a Gaussian kernel,” *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 41, no. 2, pp. 330–340, Apr. 2011.

[40] S. Shalev-Shwartz, Y. Singer, N. Srebro, and A. Cotter, “Pegasos: Primal estimated sub-gradient solver for SVM,” *Math. Program.*, vol. 127, no. 1, pp. 3–30, 2011.

[41] C.-J. Hsieh, K.-W. Chang, C.-J. Lin, S. S. Keerthi, and S. Sundararajan, “A dual coordinate descent method for large-scale linear SVM,” in *Proc. 25th Int. Conf. Mach. Learn.*, Helsinki, Finland, Jul. 2008, pp. 408–415.
[42] T. Goldstein, B. O’Donoghue, and S. Setzer, “Fast alternating direction optimization methods,” *SIAM J. Imag. Sci.*, vol. 7, no. 3, pp. 1588–1623, 2014.
[43] S. Manos *et al.*, “Distributed MPI cross-site run performance using MPIg,” in *Proc. 17th Int. Symp. High Perform. Distrib. Comput.*, Boston, MA, USA, Jun. 2008, pp. 229–230.
[44] M. Zinkevich, M. Weimer, A. Smola, and L. Li, “Parallelized stochastic gradient descent,” in *Proc. Adv. Neural Inf. Process. Syst.*, Vancouver, BC, Canada, Dec. 2010, pp. 2595–2603.
[45] C.-J. Lin, R. C. Weng, and S. S. Keerthi, “Trust region Newton method for logistic regression,” *J. Mach. Learn. Res.*, vol. 9, pp. 627–650, Apr. 2008.

Huihui Wang is currently pursuing the Ph.D. degree with the State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, China.

Her current research interests include machine learning, distributed computing, convex optimization, and linear classification.

Yang Gao received the Ph.D. degree from the Department of Computer Science and Technology, Nanjing University, Nanjing, China, in 2000.

He is a Professor and the Deputy Director with the Department of Computer Science and Technology, Nanjing University, where he is currently directing the Reasoning and Learning Research Group. He has published over 100 papers in top-tiered conferences and journals, such as the IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, the IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS, the IEEE TRANSACTIONS ON BIOMEDICAL ENGINEERING, the IEEE TRANSACTIONS ON CYBERNETICS, CVIU, Pattern Recognition, the IEEE Conference on Computer Vision and Pattern Recognition, and the International Joint Conference on Autonomous Agents and Multi-agent Systems. His current research interests include artificial intelligence and machine learning.

Prof. Gao serves as the Program Chair and the Area Chair for many international conferences.

Yinghuan Shi received the B.Sc. and Ph.D. degrees from the Department of Computer Science, Nanjing University, Nanjing, China, in 2007 and 2013, respectively.

He is currently an Assistant Researcher with the Department of Computer Science and Technology, Nanjing University. He was a Visiting Scholar with the University of Technology Sydney, Sydney, NSW, Australia, in 2014, for a month, and a visiting student with the University of North Carolina, Chapel Hill, NC, USA, from 2011 to 2012, and Massey University, Palmerston North, New Zealand, from 2009 to 2010. He has published over 20 research papers in related journals and conferences, such as the IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, the IEEE TRANSACTIONS ON BIOMEDICAL ENGINEERING, the IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS, and IEEE Conference on Computer Vision and Pattern Recognition. His current research interests include computer vision and medical image analysis.

Dr. Shi serves as a Program Committee Member for several conferences, and also as a referee for several journals.

Ruili Wang received the Ph.D. degree in computer science from Dublin City University, Dublin, Ireland, in 2003.

He is an Associate Professor of Computer Science and Information Technology with the School of Engineering and Advanced Technology, Massey University, Auckland, New Zealand. He has published over 80 papers in top conferences and journals. He has received one of the most prestigious research grants in New Zealand, the Marsden Fund. His current research interests include data mining, machine learning, and speech processing.

Dr. Wang is an Associate Editor and an Editorial Board Member of five international journals.