Fast Classification Algorithms via Distributed Accelerated Alternating Direction Method of Multipliers

Huihui Wang[1,3], Shunmei Meng[2,3], Yiming Qiao[4], and Jing Zhang[2,*]
[1] Key Lab of Intelligent Perception and Systems for High-Dimensional Information of Ministry of Education, School of Computer Science and Engineering, Nanjing University of Science and Technology, Nanjing, China
[2] School of Computer Science and Engineering, Nanjing University of Science and Technology, Nanjing, China
[3] State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, China
[4] School of Computer Science and Technology, School of Software, Nanjing University of Posts and Telecommunications, Nanjing, China
Email: {huihuiwang, mengshunmei}@njust.edu.cn, yimingqiao3163@gmail.com, jzhang@njust.edu.cn

***Abstract*—Distributed machine learning has gained lots of attention due to the rapid growth of data. In this paper, we focus regularized empirical risk minimization problems, and propose two novel Distributed Accelerated Alternating Direction Method of Multipliers (D-A2DM2) algorithms for distributed classification. Based on the framework of Alternating Direction Method of Multipliers (ADMM), we decentralize the distributed classification problem as a global consensus optimization problem with a series of sub-problems. In D-A2DM2, we exploit ADMM with variance reduction for sub-problem optimization in parallel. Taking global update and local update into consideration respectively, we propose two acceleration mechanisms in the framework of D-A2DM2. In particular, inspired by Nesterov's accelerated gradient descent, we utilize it for global update to further improve time efficiency. Moreover, we also introduce Nesterov's acceleration for local update, and develop the corrected local update and symmetric dual update to accelerate the convergence with only a little change in the computational effort. Theoretically, D-A2DM2 has a linear convergence rate. Empirically, experimental results show that D-A2DM2 converge faster than existing distributed ADMM-based classification, and could be a highly efficient algorithm for practical use.**

***Index Terms*—ADMM, Nesterov's Acceleration, Distributed Classification, Accelerated ADMM**

I. Introduction

Distributed machine learning becomes increasingly important in this big data era. Moreover, the data in many big data applications is usually generated and stored in a decentralized fashion over different machines. This is particularly natural for processing large data on a computer cluster. In particular, distributed classification has become a crucial research topic in distributed machine learning [1], which takes advantage of the computational power of multi-machines to solve the optimization problem in a distributed manner [2]. In recent years, many algorithms have been proposed for solving distributed learning problems [3]–[5]. Specifically, Alternating Direction Method of Multipliers (ADMM) is a widely-used optimization algorithm in machine learning due to its flexibility in distributed computation [6].

Distributed ADMM is an iterative method which involves the computation time for optimizing sub-problems that happens locally on each node, and the communication time of information between nodes. Generally, communication between nodes is unavoidable and its cost will be significantly large. It has been observed empirically that distributed ADMM usually converges slowly and suffers from high time cost [7]–[9], which will create a new bottleneck for consensus optimization. As a consequence, the development of efficient distributed ADMM algorithms simultaneously improve computational efficiency and communication efficiency is an important problem. Unfortunately, this issue has not been well investigated in previous literatures.

In order to address these aforementioned issues, in this paper, we focus on ADMM-based distributed classification problems, and propose two novel Distributed Accelerated ADMM (D-A2DM2) algorithms that can learn new distributed classification models. Specifically, we extend an acceleration strategy for global update mainly motivated by Nesterov's Acceleration (NA) to accelerate the convergence. Furthermore, we integrate ADMM with stochastic dual coordinate ascent method (SDCA), for sub-problem optimization in parallel. Taking local update into consideration, we exploit Nesterov's Acceleration (NA) for sub-problem optimization and develop the corrected local update and symmetric dual update to further improve the convergence. We investigate the theoretical analysis and the performance of our proposed algorithms. Experiments on various datasets empirically validate that D-A2DM2 outperforms distributed ADMM-based classification algorithms. The main contributions of our work are summarized as follows:

- Two efficient algorithms are proposed for distributed

*Dr. Jing Zhang is the corresponding author of this paper. The work is supported by the National Natural Science Foundation of China under Grants (61806096, 91846104, 61702264) and the Natural Science Foundation of Jiangsu Province, China, under Grant (BK20160843).

2374-8486/19/$31.00 ©2019 IEEE
DOI 10.1109/ICDM.2019.00173

classification. In the framework of D-A2DM2, accelerated methods and corrected schemes are integrated with distributed ADMM to improve time efficiency of local computation and communication.
- We present the convergence analysis of D-A2DM2. The NA method can be well integrated with distributed ADMM to accelerate the convergence rate.
- The experimental results verify that D-A2DM2 converges faster than distributed ADMM-based classification algorithms, and thus can improve time efficiency. D-A2DM2 could be an effective and efficient algorithm for practical use on large-scale data.

The rest of this paper is organized as follows. Section II presents related work, followed by problem statement in Section III. Afterward, we describe the details of our algorithms in Section IV. The experimental results are showed in Section V. Finally, we present the conclusion in Section VI.

II. Related Work

In this section, we briefly discuss and give a overview of the mainly related work in both accelerated ADMM and distributed ADMM for classification.

A. Accelerated ADMM algorithms

Theoretical anlysis on ADMM has revealed that it has a $O(1/k)$ convergence rate, where k is the number of ADMM iterations [10], [11]. In recent years, variance reduction with stochastic sampling and Nesterov's acceleration method have been used to accelerate the convergence of ADMM.

Variance Reduction (VR): It is well known that the variance produced by stochastic sampling makes stochastic algorithms have a slower convergence rate than batch algorithms. To alleviate this problem, many variance reduction methods have been proposed to improve the convergence speed of ADMM [12], [13]. we focus on the method proposed in [12], which solves problem optimization on its dual form of the original problem. That is, in each inner loop, a randomly sampled instance i is used to update the related coordinate.

Nesterov's Acceleration (NA): Nesterov proposed an acceleration scheme for gradient methods in [14]. In particular, one or several auxiliary variables are introduced to update the parameter with respect to the the gradient at the current auxiliary variable. Recently, Nesterov-type acceleration method has been integrated with ADMM to improve the convergence [6].

B. Distributed ADMM for Classification

In the literature, distributed ADMM has been widely-used for distributed machine learning due to its flexibility towards distributed computation [15]. In particularly, Zhang *et al.* proposed distributed classification algorithms via ADMM to solve linear SVM problems [16]. Also, distributed ADMM was extended to train the linear classifier for specific tasks [17]. Nevertheless, these distributed ADMM algorithms usually converge slowly, and are time-consuming [7], [18]. Recently, various versions of stochastic ADMM were proposed to improve communication-efficient in [19], [20]. Besides, Wang *et al.* recently proposed a group-based ADMM method (GADMM) to accelerate the convergence speed, which relaxes global consensus condition, but an additional communication cost is needed in each communication iteration [8].

It has been shown that VR or NA methods can accelerate ADMM convergence. However, the parallelization of these methods has not been well investigated. Distributed ADMM usually converges slowly in practical use, and the above works lack of efforts in studying accelerated methods integrated with distributed ADMM. This is exactly what we are going to do in this paper.

III. Problem Statement

The Alternating Direction Method of Multipliers (ADMM) is a powerful optimization algorithm and has recently gained lots of attention in many applications [16]. Specifically, ADMM solves optimization problems of the form:

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

where $x \in \mathbb{R}^{N_x}$ and $z \in \mathbb{R}^{N_z}$ are variables, $f(x)$ and $g(z)$ are two convex functions. $A \in \mathbb{R}^{N_c \times N_x}$, $B \in \mathbb{R}^{N_c \times N_z}$ and $c \in \mathbb{R}^{N_c}$ are used to impose the linear constraints. Assume that the data are stored on N nodes respectively, if function $f(x)$ is a loss function, which is separable with respect to variable x, $f(x)$ can be decomposed into N components. Hence, these empirical risk minimization problems can be easily casted into an distributed ADMM form:

$$\begin{aligned} \min_{x_i,\ldots,x_n,y} & \sum_{i=1}^{N} f_i(x_i) + g(y), \\ s.t. \quad & x_i = y, i = 1, 2, \ldots, N \end{aligned} \tag{2}$$

where each f_i the local function which involves only the samples on node i. x_i and y are the local variable and global variable, respectively. Given a penalty parameter $\rho > 0$, the global consensus problem can be solved equally by optimizing its augmented Lagrangian defined as:

$$\begin{aligned} \mathcal{L}_\rho(x, y, \lambda) = & \sum_{i=1}^{N} f_i(x_i) + g(y) + \sum_{i=1}^{N} \lambda_i^T (x_i - y) \\ & + \frac{\rho}{2} \sum_{i=1}^{N} \|x_i - y\|^2, \end{aligned} \tag{3}$$

where $x = \{x_1, \ldots, x_N\}$ and $\lambda = \{\lambda_1, \ldots, \lambda_N\}$. λ_i is the Lagrangian multiplier (dual variable). Moreover, the augmented Lagrangian also can be reformed in a scaled form as follows:

$$\mathcal{L}_\rho(x, y, u) = \sum_{i=1}^{N} f_i(x_i) + g(y) + \frac{\rho}{2} \sum_{i=1}^{N} \|x_i - y + u_i\|^2, \tag{4}$$

where $u_i = \frac{\lambda_i}{\rho}$ and u_i is the dual variable. Distributed ADMM can be easily implemented in a computer cluster with one master and N workers (nodes).

IV. Distributed Accelerated ADMM

The core idea of our proposed algorithms is that we cautiously study distributed ADMM-based framework and propose efficient accelerated strategies to improve classification performance. In the following, the details will be presented.

A. Framework of D-A2DM2

The classification problem over a computer network can be formulated as a global consensus problem in ADMM form as:

$$\begin{aligned} \min_{x_i,\ldots,x_N,y} \quad & \sum_{i=1}^{N}\sum_{j=1}^{l_i} C * \xi(x_i, a_j, b_j) + g(y), \\ s.t. \quad & x_i = y, i = 1, 2, \ldots, N, \end{aligned} \tag{5}$$

where $g(y) = \frac{1}{2}\|y\|^2$, $\sum_{j=1}^{l_i} C * \xi(x_i, a_j, b_j)$ is the loss function and l_i is the number of samples in node i, where a_j is an sample and b_j is its label. Therefore, the augmented Lagrangian function for distributed classification in (5) can be mathematically formulated as follows:

$$\mathcal{L}_\rho(x, y, u) = \sum_{i=1}^{N} f_i(x_i) + g(y) + \frac{\rho}{2}\sum_{i=1}^{N}\|x_i - y + u_i\|^2, \tag{6}$$

where $f_i(x_i) = \sum_{j=1}^{l_i} C * \xi(x_i, a_j, b_j)$, and $u = \frac{1}{\rho}\lambda$. It is easy to see that x_i and u_i can be locally updated on each node. Because the whole classification problem has been divided into N sub-problems which can be solved in parallel, our D-A2DM2 is actually a distributed ADMM framework.

B. Sub-problem Optimization

ADMM alternately minimizes $L_\rho(x, y, u)$ with respect to x and y. Moreover, $L_\rho(x, y, u)$ is separable in terms of x_i, the sub-problem of a particular worker i can be rewritten as:

$$\min_{x_i} F_i(x_i) = f_i(x_i) + \frac{\rho}{2}\|x_i - v\|^2, \tag{7}$$

where $v = \hat{y}^k + \hat{u}_i^k$ at the k-th ADMM iteration, and $\hat{u}_i$ is the auxiliary variable for dual update. Although the sub-problem in (7) is different from traditional machine learning problems, $\frac{\rho}{2}\|x_i - v\|^2$ also is a L_2 regularization in terms of x_i:

$$x_i^{k+1} = \min_{x_i} f_i(x_i) + \frac{\rho}{2}\|x_i - v^k\|^2. \tag{8}$$

The sub-problem can be solved efficiently by widely-used common optimization methods [21], [22]. In this paper, we use ADMM with Stochastic Dual Coordinate Ascent (SDCA-ADMM) for sub-problem optimization [12]. In each inner loop of SDCA-ADMM, a randomly sampled instance i is used to update the related coordinate. It also has been proved that the method has a linear convergence.

C. Accelerated global Update in D-A2DM2

The master waits for collecting the newest x_i^{k+1}s and $\hat{u}_i^k$s from N workers for updating y which is formulated as:

$$y^{k+1} = \arg\min_{y} g(y) + r(y), \tag{9}$$

where $r(y) = \frac{\rho}{2}\sum_{i=1}^{N}\|x_i^{k+1} - y + \hat{u}_i^k\|^2$, which is a L_2 regularization in terms of y. Hence, the global update for y usually can be efficiently solved. After global update, the master broadcasts the latest global variable to workers, each worker waits for the new y^{k+1} to update dual variable:

$$u_i^{k+1} = \hat{u}_i^k + x_i^{k+1} - y^{k+1}. \tag{10}$$

Inspired by NA method, accelerated global update updates global and dual variables twice, is shown below:

$$\hat{u}_i^{k+1} = u_i^{k+1} + \frac{a_k - 1}{a_{k+1}}(u_i^{k+1} - u_i^k),$$

$$\hat{y}^{k+1} = y^{k+1} + \frac{a_k - 1}{a_{k+1}}(y^{k+1} - y^k).$$

After that, the dual variable conveys to worker i how far its own local variable will agree upon the global consensus. In summary, the whole process is summarized as Algorithm 1.

Algorithm 1 D-A2DM2 with Accelerated Global Update

Input: datasets: $\{D_1, D_2, \ldots, D_N\}$, x^0, $a_1 = 1$, parameters (ρ, β, r^0, d^0), tolerances (ϵ^p, ϵ^d)

while $\|r^k\|_2 > \epsilon^p$ or $\|d^k\|_2 > \epsilon^d$ **do**
- $x_i^{k+1} = \arg\min_{x_i} f_i(x_i) + \frac{\rho}{2}\|x_i - \hat{y}^k + \hat{u}_i^k\|^2$
- $y^{k+1} = \arg\min_y g(y) + \frac{\rho}{2}\sum_{i=1}^{N}\|x_i^{k+1} - y + \hat{u}_i^k\|^2$
- $u_i^{k+1} = \hat{u}_i^k + x_i^{k+1} - y^{k+1}$
- $a_{k+1} = \frac{1+\sqrt{1+4a_k^2}}{2}$
- $\hat{u}_i^{k+1} = u_i^{k+1} + \frac{a_k - 1}{a_{k+1}}(u_i^{k+1} - u_i^k)$
- $\hat{y}^{k+1} = y^{k+1} + \frac{a_k - 1}{a_{k+1}}(y^{k+1} - y^k)$
- $r^{k+1} = \sum_{i=1}^{N}(x_i^{k+1} - y^{k+1})$
- $d^{k+1} = \rho(y^{k+1} - y^k)$
- $k = k + 1$

end while

Output: The global variable y^{k+1}

D. Accelerated Local Update in D-A2DM2

We extend an acceleration strategy to accelerate local update in D-A2DM2 which presents a corrected local update step, which uses the current result and last local result to further correct local update, is defined as follows:

$$x_i^{k+1} = x_i^{k+\frac{1}{2}} + \beta(x_i^{k+\frac{1}{2}} - \hat{x}_i^k), \tag{11}$$

where $\beta \in (0, 1)$ is a relaxation factor which can enforce local variable more stabilized during optimization. Then, before the update for y, a symmetric dual update similar as that in [23] is used to update the dual variable as follow:

$$u_i^{k+\frac{1}{2}} = u_i^k + x_i^{k+1} - y^k. \tag{12}$$

The update process in (12) leverages the latest local variable to alleviate the difference between x_is and y, and accelerates x_is into global consensus. Moreover, a predictor-corrector step by NA method is used to further accelerate local update:

$$\hat{u}_i^{k+1} = u_i^{k+1} + \frac{a_k - 1}{a_{k+1}}(u_i^{k+1} - u_i^k),$$

$$\hat{x_i}^{k+1} = {x_i}^{k+1} + \frac{a_k - 1}{a_{k+1}}({x_i}^{k+1} - {x_i}^{k}).$$

At each iteration, each work i sends its current corrected local variable x_i^{k+1}, together with $u_i^{k+\frac{1}{2}}$ to the master for updating y. Typically, the specific process is described in Algorithm 2.

Algorithm 2 D-A2DM2 with Accelerated Local Update

Input: datasets: $\{D_1, D_2, \ldots, D_N\}$, $x^0 = \hat{x}^1$, $a_1 = 1$, parameters (ρ, β, r^0, d^0), tolerances (ϵ^p, ϵ^d)

while $\|r^k\|_2 > \epsilon^p$ or $\|d^k\|_2 > \epsilon^d$ **do**

$x_i^{k+\frac{1}{2}} = \arg\min_{x_i} f_i(x_i) + \frac{\rho}{2}\|x_i - y^k + \hat{u}_i^k\|^2$

$x_i^{k+1} = x_i^{k+\frac{1}{2}} + \beta(x_i^{k+\frac{1}{2}} - \hat{x}_i^k)$

$u_i^{k+\frac{1}{2}} = \hat{u}_i^k + x_i^{k+1} - y^k$

$y^{k+1} = \arg\min_y g(y) + \frac{\rho}{2}\sum_{i=1}^N \|x_i^{k+1} - y + u_i^{k+\frac{1}{2}}\|^2$

$u_i^{k+1} = u_i^{k+\frac{1}{2}} + x_i^{k+1} - y^{k+1}$

$a_{k+1} = \frac{1+\sqrt{1+4a_k^2}}{2}$

$\hat{u}_i^{k+1} = {u_i^{k+1}}^2 + \frac{a_k - 1}{a_{k+1}}(u_i^{k+1} - u_i^k)$

$\hat{x}_i^{k+1} = x_i^{k+1} + \frac{a_k - 1}{a_{k+1}}(x_i^{k+1} - x_i^k)$

$r^{k+1} = \sum_{i=1}^N (x_i^{k+1} - y^{k+1})$

$d^{k+1} = \rho(y^{k+1} - y^k)$

$k = k + 1$

end while

Output: The global variable y^{k+1}

E. Convergence Analysis

We present our analysis of the convergence property of D-A2DM2 with both convex objectives. The augmented Lagrangian function in (2) also can be rewritten as:

$$\mathcal{L}_\rho(x, y, \lambda) = f(x) + g(y) + \lambda^T(x - y) + \frac{\rho}{2}\|x - y\|^2. \tag{13}$$

D-A2DM2 iteration gives the theoretical analysis relies on the optimality conditions of x and y, and presents the convergence analysis on the optimality of λ.

Lemma 1: For (x^{k+1}, y^{k+1}, λ^{k+1}) generated by D-A2DM2 in Algorithm 2, if

$$x^{k+1} - y^{k+1} = 0, \quad y^{k+1} - y^k = 0, \tag{14}$$

(x^{k+1}, y^{k+1}, λ^{k+1}) is a solution point of the problem (2).

Proof: First, by the conditions for x and y updates in Algorithm 2, we can obtain that

$$x^{k+1} - y^{k+1} = 0, \quad \lambda^{k+1} = \lambda^{k+\frac{1}{2}}. \tag{15}$$

Consequently, together with the updates for λ^{k+1} and $\lambda^{k+\frac{1}{2}}$, we get $\lambda^k = \lambda^{k+\frac{1}{2}}$. The primal and dual variables must be feasible and satisfy the Lagrange multiplier condition. From the optimality conditions for update steps of x and y in Algorithm 2, we can derive the useful expressions as:

$$\begin{cases} \partial f(x^{k+1}) + \hat{\lambda}^k + \rho(x^{k+1} - y^k) = 0, \forall x \\ \partial g(y^{k+1}) + \lambda^{k+\frac{1}{2}} + \rho(x^{k+1} - y^{k+1}) = 0, \forall y \\ x^{k+1} - y^{k+1} = 0. \end{cases} \tag{16}$$

Consequently, together with (15), (10) and (12), we get

$$\begin{cases} \partial f(x^{k+1}) + \lambda^{k+1} + \rho(y^{k+1} - y^k) = 0, \forall x \\ \partial g(y^{k+1}) + \lambda^{k+1} = 0, \forall y \\ x^{k+1} - y^{k+1} = 0. \end{cases} \tag{17}$$

By the same way, we can obtain the primal and dual residuals are $r^k = x^k - y^k$ and $d^k = \rho(y^k - \hat{y}^k)$ in Algorithm 1.

Convergence. It is same as the theoretical analysis in [24], the optimal λ^* of D-A2DM2 can be obtained with a convergence rate of $O(1/k^2)$. Moreover, distributed ADMM has a $O(1/k)$ convergence rate [24]. Therefore, the theoretical analysis shows that integrated with acceleration methods, D-A2DM2 has a faster convergence than distributed ADMM.

V. Experiments

A. Datasets and Experimental Settings

Six large-scale binary classification datasets: rcv1, kddcup10, kdd10raw, avazu, epsilon and news20 dataset [1] are adopted for performance evaluation in this paper. Moreover, all the datasets are normalized for experiments, and the details of these datasets are tabulated in Table I.

The classification model used in the experiments is the $L2$-regularized squared hinge loss SVM model. For parameter settings, we set the hyperparameter C as constant 1 consistent with distributed ADMM for fair comparison. Parameters ρ and α can be empirically well chosen, we can set them as 1 and 1.6 in all experiments. The total time (Ttime) is all the time taken in the training process. The running time (Rtime) is the computational time (Ctime) taken for sub-problem optimization in workers, while the communication time is the information transfer and synchronization. In addition to the above mentioned time, the number of outer iterations (Iter) and accuracy (Acc(%)) are also used as evaluation metrics.

TABLE I
THE DETAILS OF ALL THE DATASETS. l IS THE NUMBER OF SAMPLES, d IS THE DIMENSION OF SAMPLES

Dataset	l	d
new20	19,996	1,355,191
epsilon	500,000	2,000
rcv1	6,797,641	47,236
kdd10raw	20,012,498	1,163,024
kddcup10	20,012,498	29,890,095
avazu	45,006,431	1,000,000

We implement these algorithms in C++ based on the publicly available parallel platform Message Passing Interface [25]. Distributed ADMM can be easily implemented in a computer cluster with one master and N workers. In these algorithms, 10 computing machines are used as the slave nodes (workers) 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.

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

B. Comparison with Various Distributed ADMM Methods

To validate the performance of our proposed methods, as well as to compare with distributed ADMM-based algorithms, we use DCA for sub-problem optimization as the baseline. The details of the comparison algorithms are described as follows:

- **D-ADMM.DCA**: It is distributed ADMM in [16], and DCA is used for sub-problem optimization.
- **D-ADMM.DCA1** : In addition to DCA, NA is applied in distributed ADMM to accelerate convergence.
- **D-ADMM.SDCA**: In the framework of distributed ADMM, SDCA is utilized to alleviate the variance problem.
- **D-A2DM2.COM**: In the framework of distributed ADMM, NA and SDCA are combined to accelerate convergence speed, and improves time efficiency.

TABLE II
PERFORMANCE COMPARISON ON DATASET AVAZU

	Iter	Rtime	Ctime	Ttime	Acc(%)
D-ADMM.DCA	500.0	1856.5	384.7	2341.6	**99.61**
D-ADMM.DCA1	164.3	843.2	265.4	1162.5	99.59
D-ADMM.SDCA	500.0	296.6	305.4	645.2	99.49
D-A2DM2.COM	**92.6**	**62.3**	**78.8**	**173.5**	99.50

TABLE III
PERFORMANCE COMPARISON ON DATASET EPSILON

	Iter	Rtime	Ctime	Ttime	Acc(%)
D-ADMM.DCA	500.0	763.4	201.6	1038.8	89.76
D-ADMM.DCA1	173.3	314.6	74.6	431.4	**89.79**
D-ADMM.SDCA	500.0	176.3	101.5	298.7	88.80
D-A2DM2.COM	**97.4**	**43.2**	**24.6**	**74.7**	88.85

TABLE IV
PERFORMANCE COMPARISON ON DATASET KDD10RAW

	Iter	Rtime	Ctime	Ttime	Acc(%)
D-ADMM.DCA	500.0	1817.3	1969.5	4023.7	**89.14**
D-ADMM.DCA1	463.6	1457.3	1542.6	3132.2	89.13
D-ADMM.SDCA	500.0	662.3	587.7	1425.6	88.36
D-A2DM2.COM	**386.4**	**373.8**	**372.5**	**813.3**	88.28

Time efficiency of local computation First, we investigate the performance of the proposed method with stochastic optimization. We set the inner iteration of DCA and SDCA in each communication round as 10 and 100 for comparison. In SDCA, one-tenth samples are chosen randomly for training. Table II-IV show that distributed ADMM with SDCA converges faster and improves time efficiency of local computation than these algorithms only with DCA. In the experimental results, we find that SDCA can significantly reduce the running time and communication time with the acceptable accuracy loss on all datasets. Compared with D-ADMM.DCA, D-ADMM.SDCA can obviously save up the total time cost with about 0.1% accuracy loss on datasets. The main possible reason is that the number of training samples are very huge, and all the samples need to be trained in each inner iteration of DCA. Therefore, stochastic ADMM with SDCA can alleviate the variance problem, and improves the convergence speed of sub-problem optimization.

Communication efficiency We evaluate the convergence of our proposed method with Nesterov-type acceleration. Compared with D-ADMM.DCA and D-ADMM.SDCA, we find that D-ADMM.DCA1 and D-A2DM2.COM integrated with NA can converges faster and obviously save up the training time on all datasets in Table II-IV. Moreover, Nesterov-type acceleration step is proposed to extrapolate u_i and update y twice in distributed ADMM. Therefore, D-ADMM.DCA1 and D-A2DM2.COM can reduce the number of iterations and simultaneously save up the computational time of local update in workers and communication time between nodes. Using the same stochastic ADMM for sub-problem optimization, we also find that D-ADMM.DCA1 and D-A2DM2.COM can obtain the competitive accuracy compared with D-ADMM.DCA and D-ADMM.SDCA, respectively. In summary, D-A2DM2 can simultaneously improve computational efficiency through faster convergence, and is suitable to deal with the distributed classification problem.

C. Evaluation Performance with Accelerated Local Update

We further discuss Nesterov's Acceleration for local update. To evaluate the performance of our proposed algorithm, SDCA is used for sub-problem optimization for fair comparison. The comparison ADMM algorithms are showed as:

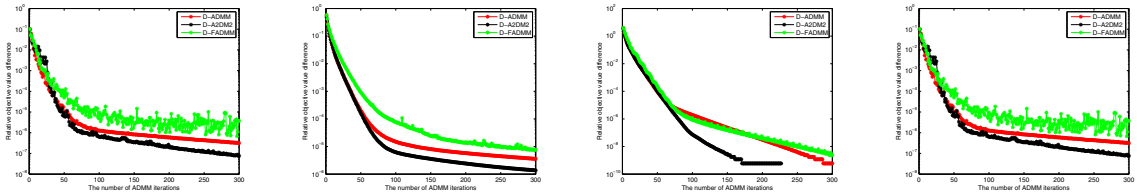
- **D-ADMM**: D-ADMM is distributed ADMM based the framework of distributed ADMM in [16].
- **DF-ADMM**: It is implemented based on the framework of fast ADMM in [24].
- **D-A2DM2**: Nesterov's Acceleration and two corrected updates are introduced in the framework.

In experiments, we measure the training performance with respect to the convergence speed. The relative objective value difference is defined as follow: [16]

$$(f - f_{min})/f_{min}, \tag{18}$$

where f is the primal function value, and f_{min} is the minimum function value found by all algorithms.

Training Efficiency Fig. 1 (a-d) show the convergence of the primal objective value as the number of outer iterations grows. We can observe that D-A2DM2 can converge faster than other distributed ADMM algorithms on all datasets. Furthermore, we can find that D-A2DM2 can significantly reduce the number of outer iterations, and thus save up the training time. In order to evaluate training efficiency of D-A2DM2 as well as to compare with the baselines, we adopt the number of outer iterations, training time and accuracy as the evaluation metrics. Table V shows that D-A2DM2 converges faster and meets the stop criterion faster than the baselines, and thus can reduce the number of outer iterations while achieving almost the same accuracy. In particularly, we can find that on the first three datasets with a huge number of samples or higher-dimension, D-A2DM2 can significantly reduce the number of outer iterations, and thus can improve time efficiency of local computation and communication. While for dataset kddcup10, D-A2DM2 can obtain the comparative performance against



(a) *epsilon* (b) *rcv1* (c) *news20* (d) *kddcup10*

Fig. 1. Convergence of these algorithms w.r.t the number of ADMM (outer) iterations.

TABLE V
NUMBER OF ITERATIONS (ITER), TOTAL TIME (TTIME) AND ACCURACY (ACC(%)) OF D-A2DM2 COMPARED WITH THE BASELINES.

Dataset	D-ADMM		D-A2DM2		DF-ADMM	
	Iter/Ttime (s)	Acc (%)	Iter/Ttime (s)	Acc (%)	Iter/Ttime (s)	Acc (%)
rcv1	73.7/6.4	**97.76**	**49.8/4.2**	97.75	184.0/14.3	97.75
news20	39.3/48.8	96.97	**19.7/25.3**	96.95	24.6/32.5	**97.07**
epsilon	280.2/114.5	**89.99**	**111.5/50.6**	**89.99**	300.0/121.4	89.98
kddcup10	38.6/1050.2	**89.07**	**26.1/788.2**	89.05	96.4/2651.4	89.04

with D-ADMM. As the results show, D-A2DM2 is suitable to solve the distributed classification problem.

VI. CONCLUSION

In this paper, we propose efficient distributed accelerated alternating direction method of multipliers (D-A2DM2) algorithms for distributed classification. Unlike traditional distributed ADMM, ADMM integrated with stochastic dual coordinate ascent is adopted for sub-problem optimization in parallel. In particular, we extend an acceleration strategy motivated by Nesterov's Acceleration (NA), and exploit two corrected schemes for local update to accelerate convergence and improve time efficiency. Our proposed methods have a linear convergence rate, and significantly saves up the time cost compared with other distributed ADMM-based classification algorithms.

REFERENCES

[1] J. Hare, S. Gupta, and T. Wettergren, "Pose. 3c: Prediction-based opportunistic sensing using distributed classification, clustering and control in heterogeneous sensor networks," *IEEE Transactions on Control of Network Systems*, 2019.

[2] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends® in Machine Learning*, vol. 3, no. 1, pp. 1–122, 2011.

[3] N. Agarwal, A. T. Suresh, F. X. X. Yu, S. Kumar, and B. McMahan, "cpsgd: Communication-efficient and differentially-private distributed sgd," in *NIPS*, 2018, pp. 7564–7575.

[4] L. Majzoobi, F. Lahouti, and V. Shah-Mansouri, "Analysis of distributed admm algorithm for consensus optimization in presence of node error," *IEEE Transactions on Signal Processing*, vol. 67, no. 7, pp. 1774–1784, 2019.

[5] Y. Zhuang, W.-S. Chin, Y.-C. Juan, and C.-J. Lin, "Distributed newton methods for regularized logistic regression," in *PAKDD*, 2015, pp. 690–703.

[6] M. Kadkhodaie, K. Christakopoulou, M. Sanjabi, and A. Banerjee, "Accelerated alternating direction method of multipliers," in *SIGKDD*, 2015, pp. 497–506.

[7] C.-P. Lee and D. Roth, "Distributed box-constrained quadratic optimization for dual linear SVM," in *ICML*, Lille, France, July. 2015.

[8] H. Wang, Y. Gao, Y. Shi, and R. Wang, "Group-based alternating direction method of multipliers for distributed linear classification." *IEEE transactions on cybernetics*, vol. 47, no. 11, pp. 2568–3582, 2017.

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

[10] B. He and X. Yuan, "On the o(1/n) convergence rate of the douglas-rachford alternating direction method," *SIAM Journal on Numerical Analysis*, vol. 50, no. 2, pp. 700–709, 2012.

[11] W. Deng, M.-J. Lai, Z. Peng, and W. Yin, "Parallel multi-block admm with o(1 / k) convergence," *Journal of Scientific Computing*, vol. 71, no. 2, pp. 712–736, 2017.

[12] T. Suzuki, "Stochastic dual coordinate ascent with alternating direction method of multipliers." in *ICML*, 2014, pp. 736–744.

[13] Y. Liu, F. Shang, and J. Cheng, "Accelerated variance reduced stochastic admm," in *AAAI*, 2017.

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

[15] W. Shi, Q. Ling, K. Yuan, G. Wu, and W. Yin, "On the linear convergence of the admm in decentralized consensus optimization," *IEEE Transactions on Signal Processing*, vol. 62, no. 7, pp. 1750–1761, 2014.

[16] C. Zhang, H. Lee, and K. G. Shin, "Efficient distributed linear classification algorithms via the alternating direction method of multipliers," in *AISTATS*, 2012, pp. 1398–1406.

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

[18] T. Yang, "Trading computation for communication: Distributed stochastic dual coordinate ascent," in *NIPS*, 2013, pp. 629–637.

[19] T. Suzuki, "Stochastic dual coordinate ascent with alternating direction method of multipliers," in *ICML*, 2014, pp. 736–744.

[20] H. Ouyang, N. He, L. Tran, and A. G. Gray, "Stochastic alternating direction method of multipliers," in *ICML*, 2013, pp. 80–88.

[21] Y. Zhuang, Y. Juan, G.-X. Yuan, and C.-J. Lin, "Naive parallelization of coordinate descent methods and an application on multi-core l1-regularized classification," in *CIKM*, 2018, pp. 1103–1112.

[22] S. J. Reddi, A. Hefny, S. Sra, B. Poczos, and A. J. Smola, "On variance reduction in stochastic gradient descent and its asynchronous variants," in *NIPS*, 2015, pp. 2647–2655.

[23] B. He, F. Ma, and X. Yuan, "On the step size of symmetric alternating directions method of multipliers," *Avaliable on http://www. optimization-online. org*, 2015.

[24] T. Goldstein, B. O'Donoghue, S. Setzer, and R. Baraniuk, "Fast alternating direction optimization methods," *SIAM Journal on Imaging Sciences*, vol. 7, no. 3, pp. 1588–1623, 2014.

[25] S. Manos, M. Mazzeo, O. Kenway, P. V. Coveney, N. T. Karonis, and B. Toonen, "Distributed MPI cross-site run performance using MPIg," in *HPDC*, 2008, pp. 229–230.