Distributed Stochastic Alternating Direction Method of Multipliers for Big Data Classification

Huihui Wang[1], Xinwen Li[2], Xingguo Chen[2], Lianyong Qi[3], and Xiaolong Xu[1](✉)

[1] Key Laboratory of Intelligent Perception and Systems for High-Dimensional Information of Ministry of Education, Nanjing University of Science and Technology, Nanjing, China
huihuiwang@njust.edu.cn, xlxu@nuist.edu.cn

[2] School of Computer Science and Technology, School of Software, Nanjing University of Posts and Telecommunications, Nanjing, China
lxinwen31@gmail.com, chenxg@njupt.edu.cn

[3] School of Information Science and Engineering, Qufu Normal University, Jining, China
lianyongqi@gmail.com

Abstract. In recent years, classification with big data sets has become one of the latest research topic in machine learning. Distributed classification have received much attention from industry and academia. Recently, the Alternating Direction Method of Multipliers (ADMM) is a widely-used method to solve learning problems in a distributed manner due to its simplicity and scalability. However, distributed ADMM usually converges slowly and thus suffers from expensive time cost in practice. To overcome this limitation, we propose a novel distributed stochastic ADMM (DS-ADMM) algorithm for big data classification based on the MPI framework. By formulating the original problem as a series of sub-problems through a cluster of multiple computers (nodes). In particular, we exploit a stochastic method for sub-problem optimization in parallel to further improve time efficiency. The experimental results show that our proposed distributed algorithm is suitable to enhance the performance of ADMM, and can be effectively applied for big data classification.

Keywords: Big data · ADMM · Stochastic ADMM · Distributed classification

1 Introduction

Linear classification has been widely applied in machine learning, such as medical diagnosis, pattern recognition, and credit scoring [1]. Under the explosive increase of global data, extreme-scale data requiring classification is a very challenging task. Formally, the size of big data challenges standard machine learning algorithms those are not usually able to deal with huge data on a single machine which the amount of data exceeds the capabilities of its system to process the data [2]. Moreover, the data in many applications are usually generated and collected in decentralized different machines. This is particularly natural for processing the large data on a computer cluster. Nowadays, big data is attracting much attention both from industry and academia in a wide variety of fields. This is because of the availability of distributed optimization can deal with a huge amount of data. Distributed classification is developed to take advantage of the computational power of multi-machines to solve the original classification problem in a decentralized environment. In these algorithms, the global classification problem is decomposed into as a series of small classification problems (sub-problems), and then the computing nodes (different machines) solve sub-problems in parallel, and then local results of sub-problems are aggregated to obtain a global result [3].

Recently, distributed classification algorithms can be roughly divided into two categories: (i) *primal optimization* [4], in these algorithms, stochastic gradient descent (SGD) is an efficient method, which only computes the gradient of one sample instead of all gradients of the whole samples in each iteration [5]. Because of the variance in stochastic optimization, initial SGD suffers from a slow convergence rate. Recently, some many accelerated versions of SGD are proposed to efficiently solve large-scale learning problems, and obtain some better convergence rate [6]. (ii) *dual optimization* [7], which introduces dual variables and designs the dual problem of the primal problem, and then obtain the final result by solving it. Specifically, alternating direction method of multipliers (ADMM) is an efficient and widely-used optimization method in many application fields [8]. Batch ADMM need to compute the empirical risk loss on all training samples in each iteration, which makes it unsuitable for the large-scale learning problem. Thus, various versions of ADMM have been developed to improve the convergence [9]. In recent years, ADMM has been widely-used for distributed optimization because of its high decomposability property [3]. The main idea of distributed ADMM is global consensus, it means that all the local models on each machine need to be in consensus with the global model when finding the global solution.

Distributed ADMM is an iterative method which includes the computation of sub-problem optimization that happens locally on each node, and the communication of information in each iteration. A lot of distributed ADMM algorithms have been proposed for solving large-scale optimization problems [10,11]. In particularly, a distributed classification algorithms based on the ADMM framework had been proposed to solve linear SVM problems in [12]. Moreover, distributed ADMM was applied for specific tasks in [13]. Nevertheless, distributed ADMM always converges slowly, and it needs much more iterations to obtain the final solution, and thus it is always time-consuming [14]. Therefore, the development of efficient distributed ADMM algorithms improve the convergence distributed ADMM for big data classification is an important problem. However, this issue has not been well studied in the previous works.

In this work, we focus on ADMM-based distributed classification problems, and propose a novel distributed stochastic ADMM algorithms (DS-ADMM) for big data classification that can improve the efficiency of local computation and communication. In particular, we integrate a stochastic gradient descent method, Pegasos [15], into ADMM for the sub-problem optimization in the distributed ADMM framework compared with traditional distributed ADMM, and then we utilize a symmetric dual update to further reduce the time cost. Finally, we investigate the performance of our proposed algorithm. Experiments on various datasets empirically validate that DS-ADMM outperforms distributed ADMM-based classification algorithms. The main contributions of our work are briefly outlined as follows:

– An efficient classification algorithm integrated with stochastic ADMM is proposed for big data classification. Specifically, the accelerated strategy can reduce local computation cost to improve efficiency.
– We implement our proposed DS-ADMM under message passing interface (MPI), which can run on clusters with many different machines (nodes). Moreover, stochastic ADMM can efficiently solve the sub-problems in parallel. Hence, DS-ADMM can be efficiently used to handle big data classification.
– Experiments on several binary classification datasets show that DS-ADMM outperforms other distributed ADMM-based algorithms, and converges faster to reduce time cost. Therefore, it could be an effective algorithm for big data classification.

The rest of this paper is organized as follows. The background of this work is presented in Sect. 2, followed by the details of our proposed algorithms in Sect. 3. And then, the experimental results are analyzed in Sect. 4. Finally, we present the conclusion in Sect. 5.

2 Background

In this section, we briefly introduce the related background of our work, which includes linear classification, ADMM and distributed ADMM.

2.1 Linear Classification

It has shown that the performance of linear classification is close to that of non-linear classification when handling large-scale data, but with much less training time [1]. Support Vector Machine (SVM) is an widely-used tool for solving data classification [16]. Moreover, The task of learning a support vector machine can be transformed as a constrained quadratic programming problem. Formally, given a dataset $\{(\mathbf{x}_i, y_i)\}_{i=1}^{l}$, where $\mathbf{x}_i \in \mathbb{R}^n$ and $y_j \in \{-1, +1\}$. We use SVM as the classification model, and the minimization problem can be reformed as:

$$\min_{\mathbf{w}} \quad \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{l} \xi(\mathbf{w}, \mathbf{x}_j, y_j), \tag{1}$$

where $C>0$ is the penalty parameter, and ξ is the misclassification loss function usually with two common forms:

$$\max\{0, 1 - y_j \mathbf{w}^T \mathbf{x}_j\}, \max\{0, 1 - y_j \mathbf{w}^T \mathbf{x}_j\}^2$$

Furthermore, the optimization method has been proved that it can obtain an ϵ-accurate solution $\mathbf{w}^*$, if it satisfies $f(\mathbf{w}^*) \leq f(\mathbf{w}) + \epsilon$ [15,17].

2.2 ADMM

ADMM is a powerful optimization algorithm and has recently get a lot of attention in many applications [9,12]. ADMM minimizes the sum of two local problems, and alternately solves them. Specifically, ADMM solves the optimization problem in the following form:

$$\begin{aligned} \min_{\mathbf{x},\mathbf{z}} \quad & f(\mathbf{x}) + g(\mathbf{z}), \\ s.t. \quad & \mathbf{Ax} + \mathbf{Bz} = \mathbf{C}, \end{aligned} \tag{2}$$

where $\mathbf{x} \in \mathbb{R}^{N_x}$ and $\mathbf{z} \in \mathbb{R}^{N_z}$ are the optimization variables, functions $f(\mathbf{x})$ and $g(\mathbf{z})$ are convex. $\mathbf{A} \in \mathbb{R}^{N_c \times N_x}$, $\mathbf{B} \in \mathbb{R}^{N_c \times N_z}$ and $\mathbf{C} \in \mathbb{R}^{N_c}$ are the linear constraints in the problem. Given a penalty parameter $\rho > 0$, the optimization problem (2) can be efficiently solved by minimizing the augmented Lagrangian as:

$$\begin{aligned} L_\rho(\mathbf{x}, \mathbf{z}, \boldsymbol{\lambda}) = f(\mathbf{x}) + g(\mathbf{z}) + \boldsymbol{\lambda}^T(\mathbf{Ax} + \mathbf{Bz} - \mathbf{C}) \\ + \frac{\rho}{2}\|\mathbf{Ax} + \mathbf{Bz} - \mathbf{C}\|^2, \end{aligned} \tag{3}$$

where $\boldsymbol{\lambda}$ is the Lagrangian multipliers. Furthermore, we can combine the linear and quadratic terms in (3) into a slightly scaled form [3] as follows:

$$L_\rho(\mathbf{x}, \mathbf{z}, \mathbf{u}) = f(\mathbf{x}) + g(\mathbf{z}) + \frac{\rho}{2}\|\mathbf{Ax} + \mathbf{Bz} - \mathbf{C} + \mathbf{u}\|^2, \tag{4}$$

where $\mathbf{u} = \frac{1}{\rho}\boldsymbol{\lambda}$. We can find that the problem in (4) is clearly equivalent to the problem in (3), but it is more convenient to solve. Hence, alternating minimization steps of $\mathbf{x}$ and $\mathbf{z}$ can be performed as follows:

$$\mathbf{x}^{k+1} = \arg\min_{\mathbf{x}} f(\mathbf{x}) + \frac{\rho}{2}\|\mathbf{Ax} + \mathbf{Bz}^k - \mathbf{C} + \mathbf{u}^k\|^2, \tag{5}$$

$$\mathbf{z}^{k+1} = \arg\min_{\mathbf{z}} g(\mathbf{z}) + \frac{\rho}{2}\|\mathbf{Ax}^{k+1} + \mathbf{Bz} - \mathbf{C} + \mathbf{u}^k\|^2, \tag{6}$$

$$\mathbf{u}^{k+1} = \mathbf{u}^k + (\mathbf{Ax}^{k+1} + \mathbf{Bz}^{k+1} - \mathbf{C}). \tag{7}$$

ADMM is an iterative method and its objective is separable across the variables. The known theoretical study on ADMM has shown that it has a sublinear convergence rate [3,18]. In recent years, ADMM has been applied to solve large-scale machine learning problems. Moreover, many stochastic or online variants of ADMM have been proposed to further improve the time efficiency [19,20].

2.3 Distributed ADMM

Recently, distributed ADMM has been widely-used for machine learning tasks due to its simplicity and flexibility towards distributed computation [21]. Unlike the method of multipliers, ADMM minimizes $L_\rho(\mathbf{x}, \mathbf{z}, \boldsymbol{\lambda})$ in terms of $\mathbf{x}$ and $\mathbf{z}$ alternatively and then updates $\boldsymbol{\lambda}$, which enables the problem to be easily decomposed for distributed optimization. Assume that the data are stored on N nodes respectively, and the loss function $f(\mathbf{x})$ can be decomposed into N components with respect to $\mathbf{x}$. Hence, the general optimization problem can be defined as:

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

where $\mathbf{x} = (\mathbf{x}_1, \ldots, \mathbf{x}_N)$ and each f_i the local function which involves only the samples on node i. By formulating the original problem as a global optimization problem, we reforms the minimization of (2) in the ADMM form as follow:

$$\begin{aligned} &\min_{\mathbf{x}_i, \ldots, \mathbf{x}_n, \mathbf{z}} \sum_{i=1}^{n} f_i(\mathbf{x}_i) + g(\mathbf{z}), \\ &s.t. \quad \mathbf{A}\mathbf{x}_i + \mathbf{B}\mathbf{z} = \mathbf{C}, i = 1, 2, \ldots, n. \end{aligned} \tag{8}$$

In the problem (8), $\mathbf{x}_i$ and $\mathbf{z}$ can be called ad the local and global variables, respectively. Given the global consensus problem, the original optimization problem is decomposed into N sub-problems, and $f_i(\mathbf{x}_i)$ is the local objective of sub-problem i. We want to find the global optimal variable $\mathbf{z}$ that minimizes the sum of objective functions, and the constraint $\mathbf{x}_i = \mathbf{z}$ is used to force all the local variables to reach a global result. In a distributed computing environment, the global formulation is suitable for solving distributed optimization in machine learning and computer vision areas [3,22]. Moreover, the global consensus problem can be solved equally by optimizing its augmented Lagrangian which can be mathematically formulated in a scaled form as follow:

$$\mathcal{L}_\rho(\mathbf{x}, \mathbf{z}, \mathbf{u}) = \sum_{i=1}^{N} f_i(\mathbf{x}_i) + g(\mathbf{z}) + \frac{\rho}{2} \sum_{i=1}^{N} \|\mathbf{A}\mathbf{x}_i + \mathbf{B}\mathbf{z} - \mathbf{C} + \mathbf{u}_i\|^2, \tag{9}$$

where $\mathbf{u} = \frac{\lambda}{\rho}$ and $\mathbf{u}_i$ is the dual variable. As mentioned in the introduction, although distributed ADMM and its convergence rate have been studied in recent years, it usually converges slowly.

3 Distributed Stochastic ADMM for Big Data Classification

In this section, we introduce our distributed stochastic ADMM (DS-ADMM) in detail. We first present the proposed distributed ADMM framework to solve sub-problems optimization in parallel. Then, we propose a stochastic learning algorithm to speed up the convergence speed. Moreover, a symmetric dual update is used to reduce the difference between local and global variables.

3.1 Framework of DS-ADMM

Based on the framework of ADMM over a computer cluster with a star topology which has a master node and N slave nodes, the classification problem can be formulated as a global optimization problem:

$$\begin{aligned} \min_{\mathbf{w}_i,\ldots,\mathbf{w}_N,\mathbf{z}} \quad & \sum_{i=1}^{N}\sum_{j=1}^{l_i} C * \xi(\mathbf{w}_i,\mathbf{x}_j,y_j) + g(\mathbf{z}), \\ s.t. \quad & \mathbf{w}_i = \mathbf{z}, i = 1,2,\ldots,N, \end{aligned} \tag{10}$$

where $g(\mathbf{z}) = \frac{1}{2}\|\mathbf{z}\|^2$, $\sum_{j=1}^{l_i} C * \xi(\mathbf{w}_i,\mathbf{x}_j,y_j)$ is the loss function in slave node i, and l_i is the number of samples on slave node i. For simplicity, we denote $\mathbf{w} = \{\mathbf{w}_1,\ldots,\mathbf{w}_N\}$, and $\boldsymbol{\lambda} = \{\boldsymbol{\lambda}_1,\ldots,\boldsymbol{\lambda}_N\}$. Basically, The augmented Lagrangian function for distributed classification in (10) can be rewritten as:

$$\begin{aligned} L_\rho(\mathbf{w},\mathbf{z},\boldsymbol{\lambda}) = g(\mathbf{z}) + & \sum_{i=1}^{N}\sum_{j=1}^{l_i} C * \xi(\mathbf{w}_i,\mathbf{x}_j,y_j) \\ & + \sum_{i=1}^{N}(\boldsymbol{\lambda}_i(\mathbf{w}_i - \mathbf{z}) + \frac{\rho}{2}\|\mathbf{w}_i - \mathbf{z}\|^2). \end{aligned} \tag{11}$$

where $\rho > 0$ is a penalty parameter, and $\boldsymbol{\lambda} > 0$ is the dual variable. By scaling $\boldsymbol{\lambda}$ with $\mathbf{u} = \frac{\boldsymbol{\lambda}}{\rho}$ in (11), the distributed classification problem in (10) is formulated as follows:

$$L_\rho(\mathbf{w},\mathbf{z},\mathbf{u}) = \frac{1}{2}\|\mathbf{z}\|^2 + \sum_{i=1}^{N} f_i(\mathbf{w}_i) + \frac{\rho}{2}\sum_{i=1}^{N}\|\mathbf{w}_i - \mathbf{z} + \mathbf{u}_i\|^2. \tag{12}$$

where $f_i(\mathbf{w}_i) = \sum_{j=1}^{l_i} C * \xi(\mathbf{w}_i,\mathbf{x}_j,y_j)$. Therefore, the problem in (10) can be solved by the minimization in terms of $\mathbf{w}_i$ and $\mathbf{z}$ alternately:

$$\mathbf{w}_i^{k+1} = \arg\min_{\mathbf{w}_i} L_\rho(\mathbf{w},\mathbf{z}^k,\mathbf{u}^k), \tag{13}$$

$$\mathbf{z}^{k+1} = \arg\min_{\mathbf{z}} L_\rho(\mathbf{w}^{k+1},\mathbf{z},\mathbf{u}^k), \tag{14}$$

$$\mathbf{u}_i^{k+1} = \mathbf{u}_i^k + \mathbf{w}_i^{k+1} - \mathbf{z}^{k+1}. \tag{15}$$

Note that distributed ADMM updates $\mathbf{w}_i$s and $\mathbf{u}_i$s locally on different nodes. Because distributed classification on big data can be decomposed into N subproblems solved in parallel. Actually, our proposed DS-ADMM is a distributed ADMM framework. In the problem (10), each slave node optimizes its local variable $\mathbf{w}_i$ by oneself, and then sends $\mathbf{w}_i$ to generate the global variable $\mathbf{z}$ on the master node. Finally, the latest $\mathbf{z}$ is broadcasted to each slave node for $\mathbf{u}_i$-updating in each ADMM iteration until the train process obtain the consensus global variable.

3.2 Stochastic Learning for Sub-problem Optimization

In sub-problem optimization. it easy to find that $L_\rho(\mathbf{w}, \mathbf{z}, \mathbf{u})$ is separable with respect to $\mathbf{w}_i$. Consider a particular worker i, the sub-problem optimization problem can be formulated in a readable way:

$$\min_{\mathbf{w}_i} F_i(\mathbf{w}_i) = f_i(\mathbf{w}_i) + \frac{\rho}{2}\|\mathbf{w}_i - \mathbf{v}\|^2, \tag{16}$$

where $\mathbf{v} = \mathbf{z}^k - \mathbf{u}_i^k$ at the k_{th} ADMM iteration. Although the sub-problem is different from traditional machine learning problems in its regularization term, $\frac{\rho}{2}\|\mathbf{w}_i - \mathbf{v}\|^2$ also is a $L2$ regularization function in the $\mathbf{w}_i$-update of sub-problem i. Therefore, the sub-problem can be solved efficiently by widely used optimization methods such as gradient descent method, dual coordinate descent method and trust region Newton method [6,17,23]

In this paper, we utilize Pegasos in [15], a stochastic method, to solve $L2$-regularized $L1$-loss SVM with the objective function as follows:

$$\min_{\mathbf{w}} F_i(\mathbf{w}, A^k) = \frac{\lambda}{2}\|\mathbf{w}\|^2 + \frac{1}{m}\sum_{(\mathbf{x},y)\in A^k} \max\{0, 1 - y\mathbf{w}^T\mathbf{x}\}, \tag{17}$$

where subset A^k is the subset of size m chosen at iteration k. Pegasos is selected owing to the following reasons: (i) Pegasos performs SGD on the primal objective (16), which can be used to accelerate the convergence. (ii) A projection step is incorporated in Pegasos, and it has been proved that Pegasos has an ϵ-accurate solution after $O(1/(\epsilon))$ iterations [1]. Hence, Pegasos can obtain a fast convergence result. Given a training subset D^k of size m at each iteration, Pegasos approximately solves the objective in Eq. (16) replaced as follows:

$$\min_{\mathbf{w}_i} F_i(\mathbf{w}_i, D) = \frac{\rho\lambda}{2}\|\mathbf{w}_i - \mathbf{v}\|^2 + \frac{C}{m}\sum_{(\mathbf{x}_j,y_j)\in D^k} \max\{0, 1 - y_j\mathbf{w}_i^T\mathbf{x}_j\}, \tag{18}$$

Here, if D is the whole training set, Pegasos is a batch method with the deterministic setting. Pegasos can take the subgradient direction of $F_i(\mathbf{w}_i, D)$ because that $L1$-loss is not differentiable:

$$\bigtriangledown F_i(\mathbf{w}_i^k, D^k) = \rho\lambda(\mathbf{w}_i^k - \mathbf{v}^k) - \frac{C}{m}\sum_{j\in D_+^k} y_j\mathbf{x}_j, \tag{19}$$

where $D_+^k = \{\mathbf{x}_j | 1 - y_j\mathbf{w}_i^T\mathbf{x}_j > 0\}$, and update $\mathbf{w}_i$ by

$$\mathbf{w}_i^{k+\frac{1}{2}} \leftarrow \mathbf{w}_i^k - \eta\bigtriangledown F_i(\mathbf{w}_i^k, D^k), \tag{20}$$

where $\eta^k = C/\rho\lambda k$ is the learning rate and k is the iteration number. Compared with subgradient descent, Pegasos obtains $\mathbf{w}_i^{k+\frac{1}{2}}$, and then projects it onto the ball set where $\{\mathbf{w}_i | \|\mathbf{w}_i\| \le 1/\sqrt{\frac{\rho\lambda}{C}}\}$. Therefore, the procedure of Pegasos can be summarized in Algorithm 1.

Algorithm 1. Pegasos for sub-problem optimization

```
Input: Choose $\mathbf{w}_i^1$ such that $\|\mathbf{w}_i^1\| \leq 1/\sqrt{\frac{\rho\lambda}{C}}\}$.
1: for $k = 1, 2, 3, \ldots, T$ do
2:     Choose training subset $D^k$, where $|D^k| = m$, uniformly at random.
3:     Compute the learning rate $\eta = C/\rho\lambda k$.
4:     Compute the subgradient of $F_i(\mathbf{w}_i^k, D^k)$ by (19).
5:     Update the latest $\mathbf{w}_i^{k+\frac{1}{2}} \leftarrow \mathbf{w}_i^k - \eta\bigtriangledown F_i(\mathbf{w}_i^k, D^k)$.
6:     Project $\mathbf{w}_i^{k+1} \leftarrow \min\{1, \frac{1/\sqrt{\frac{\rho\lambda}{C}}}{\|\mathbf{w}_i^{k+\frac{1}{2}}\|}\}\mathbf{w}_i^{k+\frac{1}{2}}$.
7: end for
Output: The local result $\mathbf{w}_i^{k+1}$
```

3.3 Update Procedures of Global and Dual Variables

Firstly, before y-update, a symmetric dual update similar as that in [24] is used to update the dual variable $\mathbf{u}_i^{k+\frac{1}{2}}$ as

$$\mathbf{u}_i^{k+\frac{1}{2}} \leftarrow \mathbf{u}_i^k + \mathbf{w}_i^{k+1} - \mathbf{z}^k. \tag{21}$$

The update process in (21) can reduce the difference between $\mathbf{w}_i$ and $\mathbf{z}$, and pull $\mathbf{w}_i$ into global consensus when solving sub-problems. Then, the new $\mathbf{w}_i^{k+1}$, together with $\mathbf{u}_i^{k+\frac{1}{2}}$, are sent to the master for $\mathbf{z}$-update which can be mathematically formulated as

$$\mathbf{z}^{k+1} = \frac{1}{2}\|\mathbf{z}\|^2 + \frac{\rho}{2}\sum_{i=1}^{N}\|\mathbf{w}_i - \mathbf{z} + \mathbf{u}_i\|^2. \tag{22}$$

Because of the right term in (22) is differentiable, thus the global variable $\mathbf{z}^{k+1}$ can be updated by the weighted average of $\mathbf{x}_i^{k+1}$s and $\mathbf{u}_i^{k+\frac{1}{2}}$s which are showed as follows:

$$\mathbf{z}^{k+1} = \frac{\rho}{1+N\rho}\sum_{i=1}^{N}(\mathbf{w}_i^{k+1} + \mathbf{u}_i^{k+\frac{1}{2}}). \tag{23}$$

Finally, each worker waits for the new $\mathbf{z}^{k+1}$ which is broadcasted from the master for the dual update which is same as that in distributed ADMM

$$\mathbf{u}_i^{k+1} \leftarrow \mathbf{u}_i^{k+\frac{1}{2}} + \mathbf{w}_i^{k+1} - \mathbf{z}^{k+1}. \tag{24}$$

Hence, the update for $\mathbf{z}$ usually can be efficiently solved in many optimization problems. Moreover, each worker can send $\mathbf{w}_i^{k+1}$ and $\mathbf{u}_i^{k+\frac{1}{2}}$ to the master for $\mathbf{z}$-update, and can reduce the communication at each iteration. In summary, the overall procedure of our proposed DS-ADMM is showed in Algorithm 2.

Algorithm 2. DS-ADMM for Big Data Classification

```
Input: $\mathbf{w}_i^0$, $\mathbf{u}_i^0$ and $\mathbf{z}^0$, parameters ($\mathbf{r}^0$, $\mathbf{s}^0$), tolerances ($\epsilon^p$, $\epsilon^d$)
1: while $\|\mathbf{r}^k\|_2 > \epsilon^p$ or $\|\mathbf{s}^k\|_2 > \epsilon^d$ do
2:     $\mathbf{w}_i^{k+1} = \min_{\mathbf{w}_i} f_i(\mathbf{w}_i) + \frac{\rho}{2}\|\mathbf{w}_i - \mathbf{v}\|^2$ solved by Pegasos in parallel.
3:     $\mathbf{u}_i^{k+\frac{1}{2}} \leftarrow \mathbf{u}_i^k + \mathbf{w}_i^{k+1} - \mathbf{z}^k$.
4:     $\mathbf{z}^{k+1} = \frac{\rho}{1+N\rho}\sum_{i=1}^N(\mathbf{w}_i^{k+1} + \mathbf{u}_i^{k+\frac{1}{2}})$.
5:     $\mathbf{u}_i^{k+1} \leftarrow \mathbf{u}_i^{k+\frac{1}{2}} + \mathbf{w}_i^{k+1} - \mathbf{z}^{k+1}$.
6:     $\mathbf{r}^{k+1} \leftarrow \sum_{i=1}^N(\mathbf{w}_i^{k+1} - \mathbf{z}^{k+1})$
7:     $\mathbf{s}^{k+1} \leftarrow \rho(\mathbf{z}^{k+1} - \mathbf{z}^k)$
8:     $k \leftarrow k+1$
9: end while
Output: The global variable $\mathbf{z}^{k+1}$
```

4 Experiments

4.1 Experimental Datasets and Settings

In this paper, we perform binary classification tasks on three benchmark datasets: *webspam*, *rcv1* and *epsilon* whose detailed information can be found from LibSVM website[1] for performance evaluation. Furthermore, The details of experimental datasets are showed in Table 1. For parameter settings, we choose the hyperparameter C is consistent with distributed ADMM in [12] for fair comparison. Also, it has been studied that its relaxation form of local variable $\mathbf{w}_i^{k+1}$ can facilitate the solution, which is defined as followes:

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

where $\alpha \in (0, 2)$ is a relaxation parameter, and it had been analyzed and suggested that $\alpha \in (1.5, 1.8)$ can improve the convergence in [3]. In this paper, we define M as the number of inner iterations of sub-problem optimization in each ADMM iteration. For ρ, α and M, we set them as 1, 1.6, 50 which is same with [12] for comparative tests. Moreover, we empirically choose the parameter in Pegasos which is guided by [15] for sub-problem optimization. All the algorithms are implemented under an MPI-cluster with ten nodes, each of which has a 2.6 GHz Intel(R) Xeon(R) processor and 64 GB RAM.

4.2 Comparison with Other Distributed ADMM-Based Algorithms

To validate the effectiveness of our proposed algorithm, CS-ADMM is compared with distributed ADMM-based algorithms. In sub-problem optimization, we use DCD [17] and a trust region newton method (TRN) [23] as the baseline for sub-problem optimization, and Pegasos as acceleration to evaluate the local computation. To evaluate the performance, we use the number of outer iterations

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

Table 1. Experimental datasets. d and l are the dimension and the number of examples respectively, 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	7,236	1

((Iter), total time (Ttime), communication time (Ctime), running time (Rtime) and accuracy (Acc(%)) as evaluation metrics. The details of comparison algorithms are described as follows:

- **D-ADMM.D**: D-ADMM is distributed ADMM based the original framework of ADMM [12], and DCD is used for sub-problem optimization.
- **D-ADMM.N** : In the distributed ADMM framework, TRN is used to solve the SVM model with $L2$-regularized squared hinge loss.
- **DS-ADMM**: Ttis based on the framework of distributed ADMM. Pegasos is utilized to solve sub-problems, and a symmetric dual update is used to update dual variables before **z**-update.

Table 2. Performance comparisons on dataset *webspam*

	Iter	Ttime(s)	Ctime(s)	Rtime(s)	Acc(%)
DS-ADMM	8.9	**565.4**	**498.1**	**67.3**	98.83
D-ADMM.D	25.2	872.4	704.2	168.2	99.26
D-ADMM.N	**7.8**	1405.3	789.6	615.7	**99.34**

Table 3. Performance comparisons on dataset *epsilon*

	Iter	Ttime(s)	Ctime(s)	Rtime(s)	Acc(%)
DS-ADMM	**56.3**	**45.7**	**6.5**	**39.2**	89.08
D-ADMM.D	76.4	86.4	10.6	75.8	89.81
D-ADMM.N	65.4	164.5	72.2	92.3	**89.86**

Time Cost of Local Computation. We study the performance of our proposed algorithm with stochastic optimization. Table 2, 3 and 4 show that distributed stochastic ADMM integrated with Pegasos converges faster and improves time efficiency than these algorithms only with DCD and TRN methods, respectively. In the experimental results, we can find that Pegasos significantly reduces the running time of local computation with the acceptable accuracy loss on all datasets. In particular, compared with D-ADMM.D, DS-ADMM

Table 4. Performance comparisons on dataset *rcv1*

	Iter	Ttime(s)	Ctime(s)	Rtime(s)	Acc(%)
DS-ADMM	34.5	**6.2**	**3.4**	**2.8**	97.48
D-ADMM.D	53.2	12.6	3.8	8.8	**97.82**
DS-ADMM.N	**30.4**	18.9	4.3	14.6	97.80

can save up about 2 times of the total time with about 0.4% accuracy loss on dataset *rcv1*. The main possible reason may be that in large-scale data, the number of training samples are very huge, and the local models would be well trained in inner iterations. Therefore, stochastic ADMM can improve the convergence speed of sub-problem optimization.

Accuracy and Efficiency. We find that DS-ADMM integrated with Pegasos can obviously save up the training time with acceptive accuracy loss, which is less than 0.8%, on all datasets in Table 2, 3 and 4. The main reason is that DCD is a batch optimization method, in which all examples should be trained to learn the classification model in each inner iteration. TRN uses a conjugate gradient method to find the Newton-like direction and use the trust region Newton method to iterate. While Pegasos is a min-batch method, which will may introduce the noise in the learning process. Hence, stochastic ADMM with Pegasos reduces the time cost with a little accuracy loss. Moreover, we can find that TRN is more time-consuming compared with DCD and Pegasos when dealing with high-dimensional data, such as *webspam* and *rcv1* datasets. To sum up, DS-ADMM converges faster, and can obviously reduce the time cost with the competitive accuracy compared with D-ADMM.D and D-ADMM.N. Hence, DS-ADMM is can be applied for large-scale machine learning, could be an effective algorithm for big data classification.

5 Conclusion

In this paper, we propose a novel distributed stochastic ADMM algorithm called as DS-ADMM, for big data classification. Specifically, we explore a distributed framework based on ADMM, and divide the global problem into small sub-problems. And then, we integrate a stochastic method, Pegasos with ADMM in the distributed framework for sub-problem optimization. Furthermore, we utilize a symmetric dual update to reduce the difference between local variables and the global variable, which can force to reach global consensus. Finally, experiments on binary classification datasets show that DS-ADMM outperforms distributed ADMM-based classification algorithms. Therefore, it could be used for big data classification.

Acknowledgement. This work is partially supported by the Fundamental Research Funds for the Central Universities, No. 30918014108, and the National Science Foundation of China No. 61806096.

References

1. Yuan, G.-X., Ho, C.-H., Lin, C.-J.: Recent advances of large-scale linear classification. Proc. IEEE **100**(9), 2584–2603 (2012)
2. Zaharia, M., et al.: Apache spark: a unified engine for big data processing. Commun. ACM **59**(11), 56–65 (2016)
3. Boyd, S., Parikh, N., Chu, E., Peleato, B., Eckstein, J.: Distributed optimization and statistical learning via the alternating direction method of multipliers. Found. Trends ® Mach. Learn. **3**(1), 1–122 (2011)
4. Agarwal, N., Suresh, A.T., Yu, F.X.X., Kumar, S., McMahan, B.: cpSGD: Communication-efficient and differentially-private distributed SGD. In: Advances in Neural Information Processing Systems, pp. 7564–7575 (2018)
5. Haddadpour F., Kamani, M.M., Mahdavi M., Cadambe V.: Trading redundancy for communication: speeding up distributed SGD for non-convex optimization. In: International Conference on Machine Learning, pp. 2545–2554 (2019)
6. Reddi, S.J., Hefny, A., Sra, S., Poczos, B., Smola, A.J.: On variance reduction in stochastic gradient descent and its asynchronous variants. In: Advances in Neural Information Processing Systems, pp. 2647–2655 (2015)
7. Suzuki, T.: Stochastic dual coordinate ascent with alternating direction method of multipliers. In: Proceedings of the 31st International Conference on Machine Learning, pp. 736–744 (2014)
8. Wang, H., Gao, Y., Shi, Y., Wang, R.: Group-based alternating direction method of multipliers for distributed linear classification. IEEE Trans. Cybern. **47**(11), 3568–3582 (2017)
9. Zheng, S., Kwok, J.T.: Stochastic variance-reduced admm. arXiv preprint arXiv:1604.07070 (2016)
10. Dajun, D., Xue, L., Wenting, L., Rui, C., Minrui, F., Lei, W.: Admm-based distributed state estimation of smart grid under data deception and denial of service attacks. IEEE Trans. Syst. Man Cybern. Syst. **49**(8), 1698–1711 (2019)
11. Forero, P.A., Cano, A., Giannakis, G.B.: Consensus-based distributed support vector machines. J. Mach. Learn. Res. **11**, 1663–1707 (2010)
12. Zhang, C., Lee, H., Shin, K.: Efficient distributed linear classification algorithms via the alternating direction method of multipliers. In: Artificial Intelligence and Statistics, pages 1398–1406 (2012)
13. Lee, C.P., Chang, K.W., Upadhyay, S., Roth, D.: Distributed training of structured SVM. arXiv preprint arXiv:1506.02620 (2015)
14. Lee, C.P., Roth, D.: Distributed box-constrained quadratic optimization for dual linear SVM. In: International Conference on Machine Learning, pp. 987–996 (2015)
15. Shai, S.-S., Yoram, S., Nathan, S., Andrew, C.: Pegasos: primal estimated subgradient solver for SVM. Math. Program. **127**(1), 3–30 (2011). https://doi.org/10.1007/s10107-010-0420-4
16. Xiaohe, W., Wangmeng, Z., Liang, L., Wei, J., Zhang, D.: F-SVM: Combination of feature transformation and SVM learning via convex relaxation. IEEE Trans. Neural Netw. Learn. Syst. **29**(11), 5185–5199 (2018)
17. Hsieh, C.J., Chang, K.W., Lin, C.J., Keerthi, S.S., Sundararajan, S.: A dual coordinate descent method for large-scale linear SVM. In: Proceedings of the 25th international conference on Machine learning, pp. 408–415 (2008)
18. He, B., Yuan, X.: On the o(1/n) convergence rate of the douglas-rachford alternating direction method. SIAM J. Numer. Anal. **50**(2), 700–709 (2012)

19. Liu Y., Shang F., Cheng J.: Accelerated variance reduced stochastic ADMM. In: Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, pp. 2287–2293 (2017)
20. Yu Y., Huang L.: Fast stochastic variance reduced ADMM for stochastic composition optimization. In: Proceedings of International Joint Conferences on Artifical Intelligence, pp. 3364–3370 (2017)
21. Shi, W., Ling, Q., Yuan, K., Gang, W., Yin, W.: On the linear convergence of the ADMM in decentralized consensus optimization. IEEE Trans. Signal Process. **62**(7), 1750–1761 (2014)
22. Chen, C., He, B., Ye, Y., Yuan, X.: The direct extension of admm for multi-block convex minimization problems is not necessarily convergent. Math. Program. **155**(1–2), 57–79 (2016)
23. Lin, C.J., Weng, R.C., Keerthi, S.S.: Trust region newton method for logistic regression. J. Mach. Learn. Res. **9**, 627–650 (2008)
24. He, B., Ma, F., Yuan, X.: On the step size of symmetric alternating directions method of multipliers (2015). http://www.optimization-online.org