Asynchronous Distributed ADMM for Learning with Large-Scale and High-Dimensional Sparse Data Set

Dongxia Wang and Yongmei Lei(✉)

School of Computer Engineering and Science of Shanghai University, No. 333, Nanchen Road, Baoshan District, Shanghai 200436, China
wangdongxia1983@126.com, lei@shu.edu.cn

Abstract. The distributed alternating direction method of multipliers is an effective method to solve large-scale machine learning. At present, most distributed ADMM algorithms need to transfer the entire model parameter in the communication, which leads to high communication cost, especially when the features of model parameter is very large. In this paper, an asynchronous distributed ADMM algorithm (GA-ADMM) based on general form consensus is proposed. First, the GA-ADMM algorithm filters the information transmitted between nodes by analyzing the characteristics of high-dimensional sparse data set: only associated features, rather than all features of the model, need to be transmitted between workers and the master, thus greatly reducing the communication cost. Second, the bounded asynchronous communication protocol is used to further improve the performance of the algorithm. The convergence of the algorithm is also analyzed theoretically when the objective function is non-convex. Finally, the algorithm is tested on the cluster supercomputer "Ziqiang 4000". The experiments show that the GA-ADMM algorithm converges when appropriate parameters are selected, the GA-ADMM algorithm requires less system time to reach convergence than the AD-ADMM algorithm, and the accuracy of these two algorithms is approximate.

Keywords: GA-ADMM · General form consensus · Bounded asynchronous · Non-convex

1 Introduction

Information processing is an important research field of science and technology. With the advent of the era of big data, information processing has become more and more complex, and the classification and systematization of big data has become a research hotspot. The classification problem of big data can

Supported by the National Natural Science Foundation of China under grant No. U1811461.

G. Gui and L. Yun (Eds.): ADHIP 2019, LNICST 302, pp. 259–274, 2019.
https://doi.org/10.1007/978-3-030-36405-2_27

be abstracted to solve the optimization problem, which can be described by formula (1):

$$minf_x(x, D), \tag{1}$$

where $D \in R^{m*n}$ represents the training samples, $x \in R^n$ is the model parameter. m represents the number of samples and n represents the number of features. Due to the large amount of data, in formula (1), sometimes not only is the data set size large but also the number of features is large. Therefore, it is often difficult to solve the above problem in a valid time with a single node, and how to solve large-scale optimization problems by the distributed environment is the main problem at present. The distributed alternating direction multiplier method (ADMM) is an effective method to solve the optimization problems. Moreover, its decomposability makes it suitable for distributed systems. The main idea of the distributed ADMM algorithm is to transform large global problem into multiple small, local sub-problems, and derive the solution of global problem by coordinating the solutions of sub-problems [1]. We can transform formula (1) into a global consensus optimization problem, as shown in formula (2):

$$\begin{aligned} &min \sum_{i=1}^{M} f_i(x_i, D_i), \\ &s.t.\ x_i = z, i = 1, 2, ..., M, \end{aligned} \tag{2}$$

where $D_i \in R^{m_i*n}$, $\sum_{i=1}^{M} m_i = m$, M represents the number of nodes, $x_i \in R^n$ represents the local variable, $z \in R^n$ represents the global variable, and $f_i : R^n \to R$ is the loss function. In the formula (2), the data set D is decomposed into $D_i(i = 1, 2, ..., M)$, which can be distributed and stored in different nodes. Moreover, each sub-problem $f_i(x_i, D_i)$ can be solved by one node in parallel. This method of data parallelism can effectively solve the problems caused by the large number of samples. [11] uses MapReduce to implement the distributed ADMM algorithm, and in [15], the distributed ADMM algorithm is implemented by MPI and applied to large-scale neural networks. As the number of working nodes increases, the convergence speed of the distributed ADMM algorithm will slow down. At the same time, due to the difference between nodes and network delay, each node may update variables at a different rate. So, the performance of the system is determined by the slowest node, which is called the "straggler" problem [19]. In order to speed up the convergence of distributed ADMM algorithms, [16] proposes a group-based ADMM algorithm (GADMM), which accelerates the convergence speed of the algorithm by grouping working nodes, but it will cause a certain precision loss. In order to solve the "straggler" problem, [19] proposes an asynchronous distributed ADMM algorithm (async-ADMM), which uses bounded delay and partial obstacles to ensure the convergence of the algorithm. All distributed ADMM algorithms introduced in [11,15,16,19] need to transmit the whole model parameter in each iteration, thus the communication cost is high. Especially, when the features of the model parameter is very large, the communication efficiency becomes the bottleneck of the distributed ADMM algorithm. However, for large-scale and high-dimensional sparse data set,

some features are only related to partial data blocks. For example, when dealing with text classification problems, some words appear only in part of the text, so each processor can only handle words that appear in the local corpus. Similarly, when solving global consensus optimization problem, each worker can only process the model parameter associated with its local data set, and only need to interact the associated model parameter with the master. In this case, we can convert formula (2) to formula (3):

$$\begin{aligned} &min \sum_{i=1}^{M} f_i(x_i, D_i), \\ &s.t.\ x_i = z_{G_i}, i = 1, 2, ..., M, \end{aligned} \tag{3}$$

where $x_i \in R^{n_i}$ is the local variable, $z \in R^n$ is the global variable, $z_{G_i} \in R^{n_i}$ represents global variables associated with the local variable x_i and it is a linear function of the global variable z. The problem described by formula (3) is also called the general form consensus optimization problem [1]. To solve the problem (3), we only need to transfer the n_i features of the model parameter. So when $n_i \ll n$, the communication cost will be greatly reduced compared to transmitting the entire model parameter.

In this paper, an asynchronous distributed ADMM algorithm (GA-ADMM) for general form consensus optimization is proposed by analyzing the characteristics of large-scale sparse data set and distributed systems. The distributed ADMM framework is used to solve the problem of general form consensus with regularization optimization, and the communication efficiency of the algorithm is improved by filtering features. The bounded asynchronous communication protocol is adopted to improve the scalability of the algorithm. Moreover, the GA-ADMM algorithm is not only applicable to convex optimization, but also applies to non-convex application.

The rest of the paper is arranged as follows: In Sect. 2, we introduce the related work of distributed algorithms, and in Sect. 3 we describe the asynchronous distributed ADMM algorithm for general form consensus with regularization optimization (GA-ADMM). The convergence of the GA-ADMM algorithm is analyzed in Sect. 4. In Sect. 5, the GA-ADMM algorithm is used to solve the sparse logistic regression problem with L1 regularization, and the public data set is used to test the performance of the algorithm on the "Ziqiang 4000" of Shanghai University, and the performance of the algorithm is compared with the AD-ADMM algorithm [2]. Finally, we summarize the work of this paper in Sect. 6.

2 Related Work of Distributed Algorithms

Memory and I/O resources are the bottlenecks of solving large-scale optimization problems with a single machine. A large number of distributed algorithms have emerged to solve these problems. [12] proposes a distributed SGD algorithm.

[18] proposes a distributed ADMM algorithm for linear classification, which has a faster convergence rate than the distributed SGD algorithm. [16] introduces a group-based ADMM algorithm (GADMM), which speeds up the convergence of the algorithm by relaxing global consensus constraints. But the accuracy of the algorithm decreases. These algorithms are all based on synchronous communication protocol. All nodes must wait for other nodes to complete the calculation before the next iteration, so the performance of the system is determined by the slowest node. Compared with synchronous algorithms, asynchronous algorithms can better adapt to heterogeneous distributed systems. [4] and [8] introduce the distributed asynchronous SGD algorithm, [10] implementes an asynchronous random coordinate descent algorithm. [19] proposes an asynchronous ADMM algorithm, which is mainly for the case where the objective function is convex. [2] and [3] propose an asynchronous AD-ADMM algorithm for non-convex functions. [5] and [17] use hierarchical communication structure to reduce the communication between nodes, thus improving the communication efficiency of distributed ADMM. Although asynchronous communication protocol can effectively solve the problem of slow nodes in the system, it is still necessary to pass the entire model parameter in each iteration. When the dimension of the data set is very high, the communication efficiency of the system is still the bottleneck of the algorithm. This is also a common problem of distributed algorithms based on data parallelism.

Another type of distributed algorithm is the distributed algorithm based on model parallel. This kind of algorithms divide the data set by features, and each node only processes the features associated with it, which can reduce the communication cost. [4] proposes an asynchronous random coordinate descent algorithm, which achieves the linear acceleration ratio in the multi-core systems. However, in [4], the shared memory method is used to implement model parallelism, which is not suitable for large-scale distributed systems. In [14], the model parallel method is used to implement the distributed dual coordinate descent algorithm. [7] proposes an stochastic coordinate descent algorithm (DF-DSCD) that supports both data parallelism and model parallelism. Each node of the DF-DSCD algorithm only needs to handle part of samples and part of features. It is not only suitable for environments with large-scale data set, but also for the scenes with high-dimensional data set. However, the main research object of this algorithm is the logistic regression problem, and there is no further research on the case that the objective function is non-convex. Moreover, stochastic coordinate descent algorithm is not applicable to optimization problems with constraints. Compared with the distributed stochastic coordinate descent algorithm, the distributed ADMM algorithm can be better adapt to the distributed environment, and has a faster convergence speed. In [13,20], the stochastic ADMM is proposed to solve the convex optimization problems. [6] proposes an incremental asynchronous distributed ADMM algorithm, which can solve the non-convex and non-smooth optimization problems. However, its workers are only responsible for calculating the gradient information, and it does not make full use of the advantages of distributed

nodes. At the same time, the algorithm uses approximate calculations for solving sub-problems, which may require more iterations to achieve convergence, thus bringing greater communication cost.

This paper proposes an asynchronous distributed ADMM algorithm based on general form consensus (GA-ADMM) for large-scale and high-dimensional sparse data set. This GA-ADMM algorithm is based on data parallelism. Unlike [2], each worker in the GA-ADMM algorithm only needs to transmit associated features to the master, which can effectively reduce the communication cost. Different from the algorithm in [6], the worker in the GA-ADMM algorithm not only calculate the local variable, but also calculate the dual variable. The master is only responsible for the update of the global variable, and the solving process of the sub-problem can be controlled flexibly. Moreover, the algorithm is applicable not only to convex optimization problem, but also to non-convex optimization problem.

3 The Distributed Asynchronous ADMM Algorithm Based on General Form Consensus with Regularization

In this section, we first introduce how to use the ADMM algorithm to solve the general form consensus with regularization optimization problem, and further proposes an asynchronous distributed ADMM algorithm to solve this kind of problems.

3.1 General Form Consensus with Regularization Optimization and ADMM

For the optimization problem of large-scale and high-dimensional sparse data set, although the features of the model parameter is very large, there are fewer features associated with each data block. This kind of problems is abstracted into the general form consensus optimization problem in [1] (as shown in formula (3)). Formula (4) adds a regularization term based on formula (3), called the general form consensus with regularization optimization [1], which is more general than (3).

$$\begin{aligned} &min \sum_{i=1}^{M} f_i(x_i, D_i) + g(z), \\ &s.t.\ x_i = z_{G_i}, i = 1, 2, ..., M, \end{aligned} \tag{4}$$

where $g : R \rightarrow R \cap \infty$ is the regularization term. In formula (4), the local variable x_i only needs to be consistent with the associated global variable z_{G_i}, so when solving the local variable, only the associated global variable z_{G_i} to be used, but not the entire global variable. For the sake of presentation, we give the following definition:

Definition 1. *$\Gamma(i)$ represents the set of features associated with the i-th (i=1,. . ., M) worker, and $|\Gamma(i)|$ represents the size of the set $\Gamma(i)$. $\Phi(j)$ represents the set of all workers associated with the j-th (j=1, . . ., n) feature, and $|\Phi(j)|$ represents the size of the set $\Phi(j)$. z_j represents the j-th feature of the global variable z. x_{ij} represents the local variable associated with z_j in the i-th node, and y_{ij} is the corresponding dual variable. The set Ω represents all the (i, j) pairs, and the pair (i,j) represent the i-th node associated with the j-th feature.*

By Definition 1, the set Ω can be expressed as shown in formula (5) and we can convert formula (4) into formula (6):

$$\Omega = \{(i,j)|\sum_{j=1}^{n} i \in \Phi(j)\} = \{(i,j)|\sum_{i=1}^{M} j \in \Gamma(i)\}, \tag{5}$$

$$\begin{aligned} &min \sum_{j=1}^{n}\sum_{i=1}^{M} f(x_{ij}, D_i) + g(z), \\ &s.t.\ x_{ij} = z_j, \forall (i,j) \in \Omega. \end{aligned} \tag{6}$$

Using the ADMM algorithm framework to solve formula (6), the iterative formulae are shown in (7)–(10):

$$x_i^{k+1} := argmin(f_i(x_i, D_i) + \sum_{j \in \Gamma(i)} \frac{\rho}{2}\|x_{ij} + \frac{y_{ij}^k}{\rho} - z_j^k\|^2)), \tag{7}$$

$$z^{k+1} := argmin_z(\sum_{j=1}^{n}\sum_{i \in \Phi(j)} (\frac{\rho}{2}\|x_{ij}^{k+1} + \frac{y_{ij}^k}{\rho} - z_j\|^2) + g(z)), \tag{8}$$

$$z_{G_i}^{k+1} := \{z_j | \forall j \in \Gamma(i)\}, \tag{9}$$

$$y_{ij}^{k+1} := y_{ij}^k + \rho(x_{ij}^{k+1} - z_j^{k+1}), \forall j \in \Gamma(i). \tag{10}$$

In the distributed system, the master updates the global variable, and M workers update the local variables and dual variables respectively. The topology diagram is shown in Fig. 1. The worker first updates the local variable, and then sends the local variable and dual variable to the master. After collecting the local variables and dual variables from all workers, the master updates the global variable. Finally, the master sends the associated global variable to each worker. After the worker receives the associated global variable, it updates the dual variable and loops until the stop condition is satisfied. The description of the synchronous ADMM algorithm based on general consensus is shown in Algorithm 1.

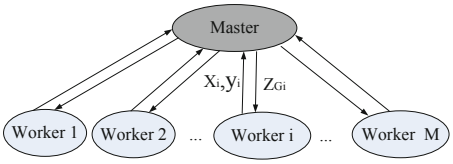

Fig. 1. The topology diagram of the ADMM algorithm based on general form consensus with regularization.

Algorithm 1. The synchronous distributed ADMM based on general form consensus with regularization

Algorithm of the Master:
Initialize z^0 and set $k = 0$.
repeat
  if *k==0* **then**
    wait until receiving x_{ij}^{k+1}, y_{ij}^k and $\Gamma(i)$ from all workers.
  end
  else
    wait until receiving x_{ij}^{k+1}, y_{ij}^k from all workers.
  end
  update z^{k+1} using (8)
  update z_{G_i} using (9)
  send z_{G_i} to all workers.
  set $k \leftarrow k + 1$
until *the stopping conditions are satisfied*;
Algorithm of the i-th worker:
Initialize x_i^0,y_i^0 and set k_i=0.
repeat
  update $x_{ij}^{k_i+1}$ using (7)
  if $k_i == 0$ **then**
    save the set $\Gamma(i)$ and send $x_{ij}^{k+1}, y_{ij}^k, \Gamma(i)$ to the master
  end
  else
    send x_{ij}^{k+1}, y_{ij}^k to the master
  end
  wait until receive z_{G_i} from the master
  update $y_{ij}^{k_i+1}$ using (10)
  set $k_i \leftarrow k_i + 1$
until *the stopping conditions are satisfied*;

3.2 Asynchronous ADMM Algorithm Based on General Form Consensus

In Algorithm 1, the master cannot update the global variable z until it receives information from all workers. This makes the system performance determined by the slowest worker. In this paper, the algorithm is further improved, and the

asynchronous ADMM algorithm is used to solve the general consensus optimization problem. In each iteration, the master only needs to receive part of the workers' information to update the global variable. In addition, we first update the dual variable, and then update the global variable, so that the latest dual variable is used to update the global variable, which can reach convergence faster. The iterative formulae are shown in (11)–(14):

$$x_i^{k+1} := argmin(f_i(x_i, D_i) + \sum_{j \in \Gamma(i)} \frac{\rho}{2} \| x_{ij} + \frac{y_{ij}^k}{\rho} - \widetilde{z_j} \|^2)), \tag{11}$$

$$y_{ij}^{k+1} := y_{ij}^k + \rho(x_{ij}^{k+1} - \widetilde{z_j}), \forall(j) \in \Gamma(i), \tag{12}$$

$$z^{k+1} := argmin_z(g(z) + \sum_{j=1}^{n} \sum_{i \in \Phi(j)} (\frac{\rho}{2} \| x_{ij}^{k+1} + \frac{y_{ij}^{k+1}}{\rho} - z_j \|^2) + \frac{\gamma}{2} \| z - z^k \|^2), \tag{13}$$

$$\widetilde{z_{Gi}} := \{z_j | \forall j \in \Gamma(i)\}, \tag{14}$$

in which $\widetilde{z_j}$ is the latest associated global variable received by the worker. In Eq. (13), we add the term $\frac{\gamma}{2} \| z - z^k \|^2$ to ensure the convergence of the algorithm. According to the characteristics of Eq. (13), we need to sum local variables and dual variables when updating the global variable z. Therefore, in the communication process, the worker send w (whose definition is shown in Eq. (15)) to the master, instead of sending local variable and dual variable respectively. This will further reduces the amount of information that the worker sends to the master:

$$w_{ij}^{k+1} = \rho x_{ij}^k + y_{ij}^{k+1}. \tag{15}$$

Since the global variable associated with each worker is different and there is only one master to update the global variable in the algorithm. Therefore, after updating the global variable z, we need to update the global variable $\widetilde{z_{Gi}}$ associated with each worker. The update formula of $\widetilde{z_{Gi}}$ is as shown in Eq. (14).

Algorithm 2 describes the GA-ADMM algorithm: the worker independently updates its local variable x_i and the dual variable y_i, and computes w_i. Then sends w_i to the master. After receiving the parameter information of A ($A \leq M$) workers, the master updates the global variable and the associated global variable $\widetilde{z_{Gi}}$, and then sends $\widetilde{z_{Gi}}$ to the corresponding worker. This loop does not stop until all the stop conditions are met. In order to avoid the mater using too old parameter information, when updating z, the algorithm requires that each worker be updated at least once in the fixed period τ. We set an independent counter t_i for each worker and save it in the master. When the worker's information reaches the master, the corresponding counter t_i is set to 0, otherwise the counter t_i is incremented by 1. All counters must be less than τ in the iterations. Since the global variables associated with each worker are different, at the first iteration, each worker needs to mark the global variable associated with

it and sends this information to the master. In the Algorithm 2, B_k represents the index subset of workers from which the master receives information during iteration k, and B_k^c represents the complementary set of B_k.

Different from the literature [6], in each iteration, we get the exact solution of the local variable x_i. The time required for solving the sub-problem is higher in the system running process. Therefore, different solving algorithms can be selected according to the specific problems in the system implementation, such as the dual coordinate descent method and the Trust Region Newton method(TRON) [9]. These two algorithms are implemented in the system. Users can select different solutions by passing parameters.

4 Convergence Analysis

In this section we analyze the convergence of the GA-ADMM algorithm. First, we make the following assumptions:

Assumption 1. *t_i is the delay number of the i-th worker, and $T(i,j)$ represents the maximum difference of the delay numbers of all workers associated with z_j (Namely, $T(i,j) = \max|t_i - t_k|, \forall i,k \in \Phi(j)$). The maximum delay number τ satisfies the condition $0 < \tau \leq T(i,j), \forall(i,j) \in \Omega$. And there is a constant $A \in [1,M]$ such that the condition $|B^k| < A$ is satisfied in each iteration.*

Assumption 2. *Each function $f_{ij}(x_{ij})$ is twice differentiable and there is a constant $L_{ij} \geq 0$ such that the gradient of $f_{ij}(x_{ij})$ satisfies the Lipschitz continuous condition.*

Assumption 3. *The regularization function g is a convex function and the domain of g is compact. Moreover, the solution of formula (4) is bounded below and there is an optimal value $\hat{f} > -\infty$.*

Theorem 1. *If Assumptions 1–3 are true, and appropriate parameters are selected to satisfy the formula (20)–(22):*

$$\infty > L_\rho(x^0, y^0, z^0) - \hat{f} \geq 0, \tag{20}$$

$$\rho \geq \frac{(1 + L_{ij} + L_{ij}^2) + \sqrt{(1 + L_{ij} + L_{ij}^2)^2 + 8L_{ij}^2}}{2}, \forall(i,j) \in \Omega, \tag{21}$$

$$\gamma > \frac{A(1+\rho^2)(\tau-1)\max(T(i,j)) - \max|\Phi(j)|\rho}{2}, \forall(i,j) \in \Omega, \tag{22}$$

then, the sequence of $(\{x_i^k\}_{i=1}^M, \{y_i^k\}_{i=1}^M, \{z_j^k\}_{j=1}^n)$ generated by the GA-ADMM is bounded and has limit points which satisfy the KKT conditions of problem (4).

It is implied by Theorem 1 that the GA-ADMM is guaranteed to converge to the set of KKT points so long as ρ and γ are large enough. It can be seen from formula (22) that A should be increased as the number of workers M increases

Algorithm 2. GA-ADMM: The asynchronous distributed ADMM based on general form consensus with regularization

Algorithm of the Master:

Initialize z^0 and set $k = 0, t_1 = t_2 = ... = t_M = 0$.

repeat

  if *k==0* **then**

    wait until receiving $\widetilde{w_{ij}}$ and $\Gamma(i)$ from all workers, set $B_k^c = \phi$.

$$w_{ij}^{k+1} = \widetilde{w_{ij}}. \tag{16}$$

  end

  else

    wait until receiving $\widetilde{w_{ij}}$ from all workers $i \in B_k$ such that $|B_k| \geq A$ and $(t_1, t_2...t_M) < \tau$, and then update

$$t_i = \begin{cases} 0 : \forall i \in B_k, \\ t_{i+1} : \forall i \in B_k^c, \end{cases} \tag{17}$$

$$w_{ij}^{k+1} = \begin{cases} \widetilde{w}_{ij} : \forall i \in B_k, \\ w_{ij}^k : \forall i \in B_k^c. \end{cases} \tag{18}$$

  end

  update

$$z^{k+1} := argmin_z(g(z) + \sum_{j=1}^{n} \sum_{i \in \Phi(j)} (\frac{\rho}{2} \| \frac{w_{ij}^{k+1}}{\rho} - z_j \|^2) + \frac{\gamma}{2} \| z - z^k \|^2). \tag{19}$$

  update $\widetilde{z_{Gi}}$ using (14)

  send $\widetilde{z_{Gi}}$ to the workers in B_k.

  set $k \leftarrow k + 1$

until *the stopping conditions are satisfied*;

Algorithm of the i-th worker:

Initialize x_i^0,y_i^0 and set k_i=0.

repeat

  update $x_{ij}^{k_i+1}, y_{ij}^{k_i+1}$ using (11) and (12)

  if $k_i == 0$ **then**

    save the set $\Gamma(i)$

  end

  compute w_{ij}^{k+1} using (15)

  if $k_i == 0$ **then**

    send $w_{ij}^{k+1}, \Gamma(i)$ to the master

  end

  else

    send w_{ij}^{k+1} to the master

  end

  wait until receive $\widetilde{z_{Gi}}$ from the master

  set $k_i \leftarrow k_i + 1$

until *the stopping conditions are satisfied*;

if the maximum delay number $\tau(\tau > 1)$ remains unchanged. This is because the larger M, the larger $|\Phi(j)|$, and the more outdated information is used for each update, so the threshold A should be increased in order to ensure the convergence speed. On the other hand, when $\tau = 0$, the GA-ADMM algorithm is equivalent to the synchronization algorithm, and γ can be set to 0. Otherwise, γ should increase as τ increases.

5 Experiments and Discussion

In this section, we use the GA-ADMM algorithm to solve the sparse logistic regression problem, test its convergence and performance, and compare the algorithm with the asynchronous distributed ADMM (AD-ADMM) algorithm proposed in [2]. The sparse logistic regression problem can be described as shown in formula (23):

$$\min \frac{1}{M} \sum_{i=1}^{M} \log(1 + \exp(-b_i D_i^T x)) + \beta \|x\|_1 \tag{23}$$

where $D_i \in R^n$ is the sample dataset, $x \in R^n$ is the model parameter, $b_i \in \{-1, 1\}$ is the label of the sample and $\beta > 0$ is the scalar regularization parameter.

5.1 Experimental Environment and Parallelization Implementation

We tested the algorithms on the cluster supercomputer "Ziqiang 4000" of Shanghai University. Each node of the cluster has an Intel E5-2690 CPU (2.9 GHz/8-core) processor and 64 GB of random access memory. The network bandwidth of the cluster is 5.6 GB. We use the KDDb (raw)[1] and KDDa[2] as the test data sets. The KDDb (raw) has more than 19 million samples and one million features, the KDDa has more than eight million samples and 20 million features. We implement the algorithm use MPICH v3.2.1 as the inter-processor communication and use C++ as the programming language.

In the system, we use nine computing nodes and 65 processes, one processes is selected as the master and the others as workers. The initial residual r^k and the dual residual s^k, which are defined as formula (24), are used to set the stopping criterion of the GA-ADMM algorithm. The algorithm doesn't stop until the initial residual and the dual residual satisfy formula (25) and (26):

$$\|r^k\|_2^2 = \frac{1}{M} \Sigma_{i=1}^{M} \sum_{j \in \Gamma(i)} \|x_{ij}^k - z_j^k\|_2^2, \|s^k\|_2^2 = \rho^2 \|z^k - z^{k-1}\|_2^2, \tag{24}$$

[1] https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary.html#kdd2010 raw version (bridge to algebra).

[2] https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary.html#kdd2010 (algebra).

$$\|r^k\|_2 \leq ABS * \sqrt{m} + REL * \max\{\frac{1}{M}\sum_{i=1}^{M}\|x_i^k\|_2, \|z^k\|_2\}, \tag{25}$$

$$\|s^k\|_2 \leq ABS * \sqrt{m} + REL * \frac{1}{M}\sum_{i=1}^{M}\|y_i^k\|_2, \tag{26}$$

where m represents the total number of samples, both the absolute error ABS and the relative error REL are set to 0.001.

5.2 Convergence Test of the GA-ADMM Algorithm

In this section we test the convergence of the GA-ADMM algorithm. 64 workers are used in the experiment, and different thresholds are selected to test the algorithm. When testing with the data set KDDb (raw), the threshold A is taken as 64, 32, and 8, respectively. While the data set KDDa is used for testing, the threshold A is taken as 64, 16, and 4, respectively. The maximum delay number τ is set to 5, and the penalty term parameter ρ is set to 6. The sub-problem is solved by the TRON. When the threshold A is set to 64, the algorithm is synchronous, and the maximum delay number has no effect on the algorithm. Figures 2 and 3 respectively show the convergence of the algorithm when the data set is KDDb (raw) and KDDa.

Figures 2 and 3 show that when the other parameters are the same, the total number of iterations required for the GA-ADMM algorithm to converge increases as the threshold decreases, but the total system time decreases. This is because the smaller the threshold, the more outdated information the master uses to update the global variable, so the total number of iterations increases. However, since the waiting time and the sending time of the master decrease, the total time decreases instead. Figures 2 and 3 also show that the convergence of the GA-ADMM algorithm can be guaranteed when appropriate parameters are selected.

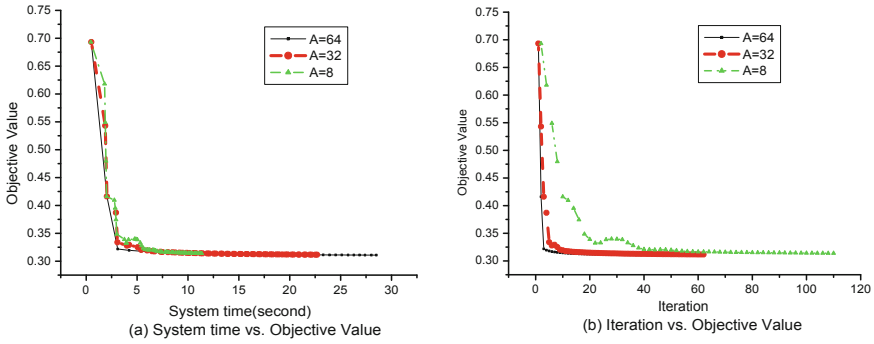

Fig. 2. Convergence of the GA-ADMM algorithm: the data set is KDDb (raw) and A represents the threshold

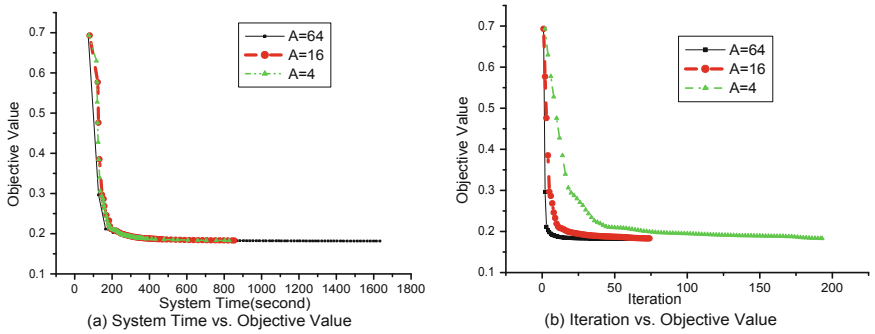

Fig. 3. Convergence of the GA-ADMM algorithm: the data set is KDDa and A represents the threshold

5.3 Performance Test of the GA-ADMM Algorithm

We test the system time cost and accuracy of the GA-ADMM algorithm in this section, and compare it with the AD-ADMM algorithm. The setting of relevant parameters is the same as in Sect. 5.2.

The system time is the running time of the master, which includes communication time and computation time. The communication time includes the time when the master waits to receive w from the workers and the time when the associated model parameters are sent to the workers, and the computation time includes the update time of the global variable z and the associated global variable z_{Gi}. Figure 4 shows the system time cost of the GA-ADMM algorithm and the AD-ADMM algorithm with different thresholds.

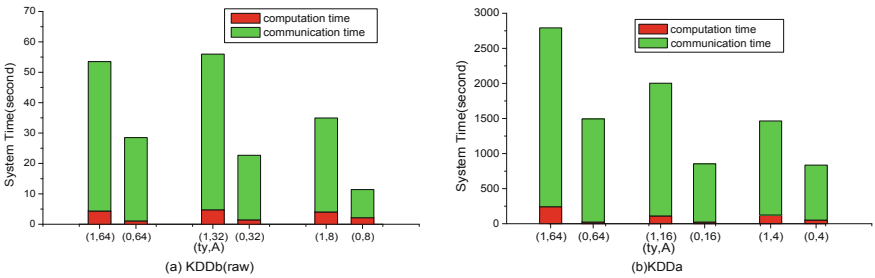

Fig. 4. The system time of the GA-ADMM and the AD-ADMM: ty represents the algorithm type, $ty = 0$ represents the GA-ADMM algorithm, and $ty = 1$ represents the AD-ADMM algorithm. A represents the threshold

It can be seen from Fig. 4 that the system time cost of the GA-ADMM algorithm is less than that of the AD-ADMM algorithm with the same parameters. On the one hand, in the GA-ADMM algorithm, only the associated parameters

need to be transmitted between nodes, so the communication time is greatly reduced. On the other hand, when updating the global variable, the AD-ADMM algorithm needs to compute all features of the model, and the GA-ADMM algorithm only needs to compute the association features of the corresponding worker, so the computation time required for each iteration is reduced. This advantage becomes more apparent as the number of features of the data set increases. As shown in Fig. 4, since the features of data set KDDa is much larger than that of data set KDDb (raw), the comparison of the computation time is more obvious.

Finally, we test the accuracy of the GA-ADMM and the AD-ADMM. The accuracy is defined as the proportion of correctly predicted samples to the total number of samples, and can be described by Eq. (27):

$$Accuracy = (N_{tp} + N_{tn})/N_{total}, \tag{27}$$

where N_{tp} represents the number of the predicted correct positive sample, N_{tn} represents the number of predicted correct negative sample, and N_{total} represents the total number of samples.

Figure 5 shows the accuracy of the GA-ADMM and the AD-ADMM at different thresholds. As can be seen from Fig. 5, the difference in accuracy between these two algorithms is not obvious. In the same situation, when the data set KDDb (raw) is used for testing, the accuracy of the GA-ADMM algorithm is slightly higher than that of the AD-ADMM algorithm, while when the data set KDDa is used for testing, the accuracy of the AD-ADMM algorithm is higher. However, the accuracy of the GA-ADMM algorithm decreases with the decrease of the threshold. This is because the global variable associated with each worker in the GA-ADMM algorithm is different, the update time of each feature of global variable is also different. When the threshold is too small, the difference will increase, which will lead to a decrease in accuracy. As shown in the analysis in Sect. 4, when the number of workers increases, the threshold should also increase in order to ensure the convergence of the algorithm.

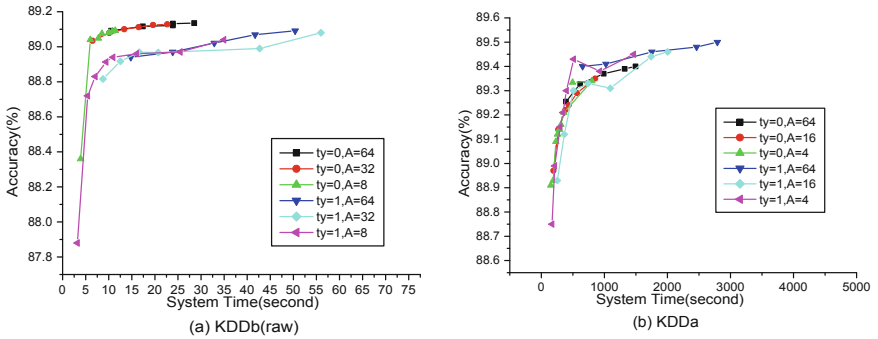

Fig. 5. The accuracy of the GA-ADMM and the AD-ADMM: ty represents the algorithm type, $ty = 0$ represents the GA-ADMM algorithm, and $ty = 1$ represents the AD-ADMM algorithm. A represents the threshold.

6 Conclusion

In order to reduce the communication cost of large-scale distributed algorithms, this paper first proposes an asynchronous distributed ADMM algorithm based on general consensus (GA-ADMM) by analyzing the characteristics of high-dimensional sparse data set and distributed ADMM algorithms. In the GA-ADMM algorithm, each worker only needs to process the associated features of the model parameter, and only the associated features need to be passed between nodes, but not all the features, thus greatly reducing the communication cost. Then, bounded asynchronous and partial obstacles are used to ensure the convergence of the algorithm, and the convergence of the algorithm is analyzed. This algorithm is not only suitable for convex optimization problem, but also for non-convex optimization problem. Finally, the algorithm is used to solve the logistic regression problem with L1 regularization on the high-performance parallel platform "Ziqiang 4000" of Shanghai University, and the algorithm is tested by KDDb(raw) and KDDa data sets. The experimental results show that the algorithm converges with reasonable parameter settings. Moreover, we compare the performance of the GA-ADMM algorithm to the AD-ADMM algorithm. Experiments also show that under the same conditions, the GA-ADMM algorithm requires less system time than the AD-ADMM algorithm, and the accuracy of these two algorithms is approximate. In this paper, the distributed ADMM algorithm is optimized mainly by reducing the communication cost between nodes. The time proportion of solving sub-problems of the algorithm is also large. We will further study the optimization strategy of solving sub-problems in the future.

Acknowledgements. This work is partially supported by the National Natural Science Foundation of China under grant No. U1811461.

References

1. Boyd, S., Parikh, N., Chu, E., Peleato, B., Eckstein, J., et al.: Distributed optimization and statistical learning via the alternating direction method of multipliers. Found. Trends® in Mach. Learn. **3**(1), 1–122 (2011)
2. Chang, T.H., Hong, M., Liao, W.C., Wang, X.: Asynchronous distributed admm for large-scale optimization–part i: algorithm and convergence analysis. IEEE Trans. Signal Process. **64**(12), 3118–3130 (2016)
3. Chang, T.H., Liao, W.C., Hong, M., Wang, X.: Asynchronous distributed admm for large-scale optimization–part ii: linear convergence analysis and numerical performance. IEEE Trans. Signal Process. **64**(12), 3131–3144 (2016)
4. Chen, T., et al.: Mxnet: a flexible and efficient machine learning library for heterogeneous distributed systems. abs/1512.01274 (2015). https://arxiv.org/abs/1512.01274
5. Fang, L., Lei, Y.: An asynchronous distributed admm algorithm and efficient communication model. In: 14th IntlConf on Pervasive Intelligence and Computing, 2nd International Conference on Big Data Intelligence and Computing and Cyber Science and Technology Congress. IEEE (2016)

6. Hong, M.: A distributed, asynchronous and incremental algorithm for nonconvex optimization: An ADMM based approach. CoRR abs/1412.6058 (2014). http://arxiv.org/abs/1412.6058
7. Kang, D., Lim, W., Shin, K., Sael, L., Kang, U.: Data/feature distributed stochastic coordinate descent for logistic regression. In: CIKM 2014 Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management. dl.acm.org (2014)
8. Li, M., G.Andersen, D., Smola, A.: Distributed delayed proximal gradient methods. In: NIPS Workshop on Optimization for Machine Learning. cs.cmu.edu (2013)
9. Lin, C.J., Weng, R.C., Keerthi, S.S.: Trust region newton method for large scale logistic regression. In: ICML 2007 Proceedings of the 24th International Conference on Machine Learning. dl.acm.org (2007)
10. Liu, J., Wright, S.J.: Asynchronous stochastic coordinate descent: parallelism and convergence properties. SIAM J. Optim. **25**(1), 351–376 (2015)
11. Lubell-Doughtie, P., Sondag, J.: Practical distributed classification using the alternating direction method of multipliers algorithm. In: Proceedings of the 33rd International Conference on Machine Learning, vol. 1. IEEE (2013)
12. Martin, Z., Markus, W., Li, L., Smola, A.J.: Parallelized stochastic gradient descent. In: Advances in Neural Information Processing Systems, vol. 23, pp. 2595–2603. Curran Associates, Inc. (2010)
13. Ouyang, H., He, N., Tran, L.Q., Gray, A.: Stochastic alternating direction method of multipliers. In: Proceedings of the 30th International Conference on Machine Learning. vol. 28. jmlr.org (2013)
14. Richtari, P., Takac, M.: Distributed coordinate descent method for learning with big data. J. Mach. Learn. Res. **17**, 1–15 (2016)
15. Taylor, G., Burmeister, R., Xu, Z., Singh, B., Patel, A., Goldstein, T.: Training neural networks without gradients:a scalable admm approach. In: IEEE International Conferences on Big Data. vol. 48. jmlr.org (2016)
16. 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)
17. Wang, S., Lei, Y.: Fast communication structure for asynchronous distributed ADMM under unbalance process arrival pattern. In: Kůrková, V., Manolopoulos, Y., Hammer, B., Iliadis, L., Maglogiannis, I. (eds.) ICANN 2018. LNCS, vol. 11139, pp. 362–371. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-01418-6_36
18. Zhang, C., Lee, H., Shin, K.G.: Efficient distributed linear classification algorithms via the alternating direction method of multipliers. In: the 15th Artificial Intelligence and Statistic. jmlr.org (2012)
19. Zhang, R., Kwok, J.: Asynchronous distributed admm for consensus optimization. In: International Conference on Machine Learning, pp. 1701–1709. jmlr.org (2014)
20. Zhong, L.W., Kwok, J.T.: Fast stochastic alternating direction method of multipliers. In: Proceedings of the 31st International Conference on Machine Learning, vol. 32. jmlr.org (2014)