Data Science and Engineering (2023) 8:61–72
https://doi.org/10.1007/s41019-022-00202-7

RESEARCH PAPERS

A Communication Efficient ADMM-based Distributed Algorithm Using Two-Dimensional Torus Grouping AllReduce

Guozheng Wang[1] · Yongmei Lei[1] · Zeyu Zhang[1] · Cunlu Peng[1]

Received: 8 April 2022 / Revised: 5 October 2022 / Accepted: 14 December 2022 / Published online: 2 January 2023

Abstract
Large-scale distributed training mainly consists of sub-model parallel training and parameter synchronization. With the expansion of training workers, the efficiency of parameter synchronization will be affected. To tackle this problem, we first propose 2D-TGA, a **g**rouping **A**llReduce method based on the two-dimensional **t**orus topology. This method synchronizes the model parameters by grouping and makes full use of bandwidth. Secondly, we propose a distributed algorithm, 2D-TGA-ADMM, which combines the 2D-TGA with the alternating direction method of multipliers (ADMM). It focuses on sub-model training and reduces the wait time among workers in the synchronization process. Finally, experimental results on the Tianhe-2 supercomputing platform show that compared with the `MPI_Allreduce`, the 2D-TGA could shorten the synchronization wait time by 33%.

Keywords ADMM · Grouping AllReduce · Two-dimensional torus topology · Synchronous algorithm

1 Introduction

In recent years, with the development of information technology, data are exploding, and we have ushered in a new era of "Big Data". Traditional machine learning focuses on the speed of data processing on a single machine, while it is impossible to store and calculate large amounts of data on a single machine. In addition, the development speed of computing engines has lagged far behind the growth speed of model computing demand. It is a necessary solution to distribute the data or model to multiple machines for computing.

Zeyu Zhang, Cunlu Peng have contributed equally.

✉ Guozheng Wang
gzh.wang@outlook.com

Yongmei Lei
lei@shu.edu.cn

Zeyu Zhang
zeyu_zhang@shu.edu.cn

Cunlu Peng
crystal886@shu.edu.cn

[1] School of Computer Engineering and Science, Shanghai University, 99 Shangda Road BaoShan District, Shanghai 20444, China

The primary purpose of this revolution is to use large amounts of data to enable knowledge discovery and better decision-making. The primitive idea of distributed machine learning (DML) is to parallelize the computing operation across multiple local devices (aka workers and nodes) to solve the following distributed optimization problem

$$\min_{\{\theta_i \in \mathbb{R}^d\}_{i=1}^N} \sum_{i=1}^{N} \mathcal{L}_i(\theta_i), \tag{1}$$

where θ_i presents model parameters vector; $\mathcal{L}_i(\theta_i)$ is the local objective function for worker$i$ ($i \in 1, 2, \cdots, N$). Distributed optimization algorithms are currently one of the most popular research directions, with a specific focus on approaches that try to optimize a performance criterion employing available data stored at local devices [1]. The ADMM combines the decomposability of dual ascent method with good convergence of Lagrange multiplier method. It may be used to solve problem (1) and has a wide range of applications. Wang et al. [2] propose an ADMM-based DML architecture that preserves privacy. Raja et al. [3] design a Secure and Private-Collaborative Intrusion Detection System (SP-CIDS). Steck et al. [4] introduce the Sparse Linear Model, which may be employed in recommendation systems and is based on the ADMM algorithm architecture.

Distributed training utilizes multiple workers to train simultaneously to accelerate the training process by cooperating with each other. Communication becomes an essential part of the cooperation process. The communication mechanism in DML is more challenging. Firstly, training machine learning models usually uses iterative optimization algorithms, and more iterations will lead to high communication frequency. Secondly, DML often trains large models. In order to obtain the updated information of the models, each worker needs to communicate with each other, which determines the large amount of communication data in DML. Thirdly, DML [5] has an important synchronization problem of model parameters. Synchronization based on AllReduce will cause the problem of "slow workers". Workers need to wait for each other and then complete the synchronization. Many methods have been proposed to deal with these challenges. Hasanov et al. [6] utilize the hierarchical idea to design the MPI reduction algorithm, and Wang et al. [7] design an asynchronous ADMM algorithm based on a hierarchical view. Xie et al. [8] design a parameter synchronization architecture for the ADMM algorithm that combines a hierarchical architecture with Ring AllReduce. The architecture adopts a stale synchronous parallel computing model. When the worker size is large, the flat parameter synchronization architecture will bring many problems, such as too small each parameter segment and too many transmission times in the Ring Allreduce architecture. A hierarchical parameter synchronization architecture can effectively alleviate these problems. The computing workers are organized in a hierarchical manner, and all workers are grouped at each level. The same worker can belong to different groups at different levels. Different levels can independently use different parameter synchronization architectures. We propose a grouping AllReduce algorithm based on the two-dimensional torus topology named 2D-TGA to alleviate these problems. The main contributions of this paper can be summarized as follows:

1) This paper proposes a synchronous algorithm 2D-TGA, which uses a grouping mode. In the first phase of the algorithm, intra-group workers perform Ring AllReduce and the groups are parallelized. In the second phase, each group selects the worker with the lowest rank as Leader and acts as the leader group. The workers in the leader group execute Ring AllReduce in a two-dimensional manner. This algorithm can further utilize the network bandwidth and reduce the synchronous time.
2) This paper proposes a distributed algorithm: 2D-TGA-ADMM, which is a distributed algorithm based on ADMM and uses 2D-TGA to synchronize model parameters. This algorithm has good scalability and can be used to solve large-scale DML problems.
3) We use the proposed distributed algorithm to solve the large-scale logistic regression problem on the Tianhe-2 supercomputing platform. The experimental results show that the efficiency of the synchronous algorithm proposed by us is better than that of the baseline algorithm.

The remainder of this paper is organized as follows. Section 2 illuminates the related work. Section 3 outlines the 2D-TGA algorithm in detail. Section 4 describes the design of the ADMM-based distributed algorithm, which uses 2D-TGA to synchronize the local model parameters. In Sect. 5, we first analyze the synchronization overhead of four synchronous AllReduce algorithms theoretically. Then, we use ADMM-based distributed algorithm to solve the logistic regression problem with $\ell 2$-norm on the Tianhe-2 supercomputing platform and verify the efficiency of our proposed 2D-TGA for parameter synchronization. Section 6 concludes the paper and future work.

2 Related Work

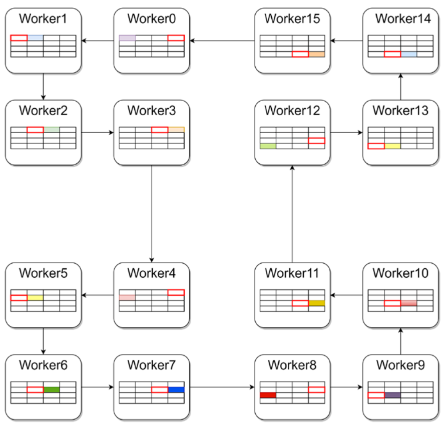
Compared with single-machine training, distributed training introduces additional data communication between workers, so that the speed of distributed training cannot improve linearly with the increase in the number of computing workers. When communication requirements and capabilities are determined, distributed machine learning training can be accelerated by improving communication efficiency.

In a decentralized parameter synchronization architecture, each worker exchanges information only with its own neighbors. The main advantage of a decentralized architecture is that it effectively reduces traffic. The typical AllReduce algorithm is to broadcast the root information to each process after the Reduction operation. This means that the root will have a communication bottleneck. The optimized algorithms are based on a few principles: reduce broadcast, recursive halving and doubling, butterfly, binary blocks, and ring [9].

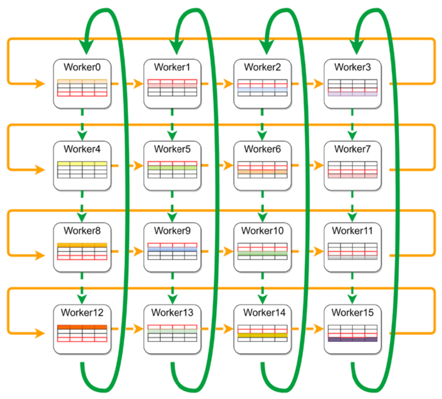
2.1 Ring AllReduce

Ring-based AllReduce is an algorithm with constant communication cost. OpenMPI [10] is implemented in 2007 based on the Ring AllReduce algorithm. Patarasuk et al [11] propose a ring-based AllReduce based on the P2P architecture and prove that a ring-based AllReduce has the optimal bandwidth of the AllReduce algorithms in 2009. Baidu Silicon Valley Lab [12] integrates this strategy with the field of deep learning in 2017 and solves the network bottleneck of synchronous updates.

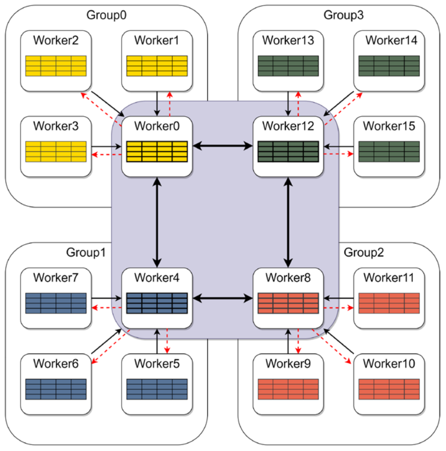
Fig. 1 Topology of AllReduce-based synchronization algorithm



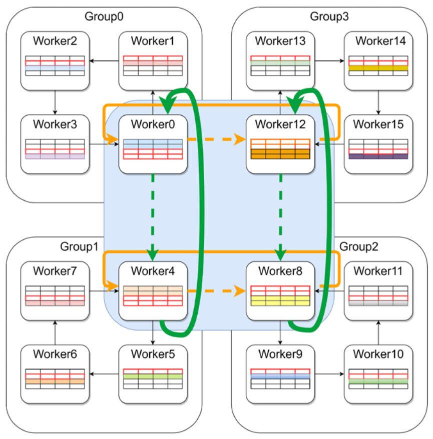
(a) Ring AllReduce: The figure shows a one-step Ring AllReduce process. The coloured block on the worker indicates the location of the data chunk to be sent, and the block framed by the red box represents the location of the data chunk to be received.



(b) 2D-Torus AllReduce: Firstly, form the workers into a regular Cartesian topology. Execute the Reduce-Scatter horizontally in parallel, as shown by the orange arrow. Then execute the AllReduce vertically in parallel, as shown by the green arrow. Lastly, execute AllGather horizontally in parallel.



(c) Hierarchical AllReduce: Firstly, the sixteen workers are divided into four subgroups, and the subgroup performs the Ring AllReduce in parallel. Secondly, each subgroup selects a leader to form the leading group, and the leader-group performs the Ring AllReduce. Thirdly, the leader performs broadcasts in the subgroup.



(d) 2D-TGA: The sixteen workers are divided into four groups, and a leader is chosen from each group. Four subgroups run the Ring AllReduce in parallel, while the leader group runs 2D-Torus allreduce. Finally, the leader broadcasts the parameters to the workers in the group.

The Ring AllReduce algorithm is depicted in Fig. 1a. The algorithm uses two phases of Reduce-Scatter and AllGather for synchronization. When N workers are in a ring topology, $2(N-1)$ steps are required to accomplish the synchronization. The Ring AllReduce can avoid contention on most network topologies [11]. As the number of workers increases, the Ring AllReduce algorithm will encounter issues such as considerable delay and low fault tolerance, which is not conducive to the scalability of distributed algorithms.

2.2 2D-Torus AllReduce

Sony [13] proposes the 2D-Torus AllReduce algorithm in 2018 to reduce the communication overhead during model parameters synchronization. The communication process is divided into two dimensions and three phases. After the two-phase combination, each GPU has some final model parameters.

Figure 1b shows that the algorithm consists of three phases, Reduce-Scatter, AllReduce and AllGather. Although a 2D-Torus AllReduce has one phase more than a Ring AllReduce, its overall communication overhead is still smaller [13]. The algorithm uses a 2D-Torus topology, which synchronizes both horizontally and vertically. In a distributed system with N workers, the algorithm requires $4(N-1)$ communication steps to accomplish the synchronization.

Ying et al. [14] find that, on a TPU Pod, the 1D Ring AllReduce algorithm is limited by the latency of pushing packets in a Hamiltonian circuit across all the nodes in the pod. They develop a 2D mesh algorithm, which performs the Reduction operation in two phases, one phase for each dimension. The 2D-Mesh algorithm has twice the throughput of gradient aggregations than the 1D Ring AllReduce.

2.3 Hierarchical AllReduce

Tencent proposes the hierarchical AllReduce algorithm [15], and the communication topology is shown in Fig. 1c. The algorithm uses a hierarchical ring topology and consists of three phases: intra-groups AllReduce, inter-groups AllReduce, and broadcast within the group. In a distributed system where N workers are divided into L groups, $3(N/L-1)+2$ communication steps are required to complete the synchronization operation. Compared with a Ring AllReduce, this three-phase hierarchical AllReduce decreases the running steps from $2(N-1)$ to $3(N/L-1)+2$. Facebook [16] applies the hierarchical AllReduce algorithm to large-scale model training, which greatly improves model training speed due to the algorithm's ability to reduce latency costs.

Although this hierarchical method reduces the number of running steps, the inter-groups operation still encounters the problem of high latency of 1D Ring AllReduce. Since grouping rings into a few hierarchical collective operations seems to give a better performance, the optimal dimensions of this hierarchical communication depend on multiple aspects. Ueno et al. [17] provide a strategy for choosing the optimal hierarchical communication for deep learning workloads.

3 Synchronous Algorithm 2D-TGA

The effectiveness of the synchronous algorithm is closely related to communication topology. The 2D-TGA method divides workers into groups to decrease communication latency and low fault tolerance when the number of workers increases in a ring topology.

2D-TGA algorithm mainly adopts logical ring Reduce-Scatter operation. As illustrated in Fig. 1d, the sixteen workers are divided into four groups, and a leader is chosen from each group. After executing the Ring AllReduce algorithm, all the workers within the group have the same parameters. By establishing a Cartesian topology for the leaders, the leaders are arranged in the form of a two-dimensional grid. Each leader exchanged parameters with the adjacent leader in different directions (UP, DOWN, LEFT, RIGHT). The mechanism will be described in the following paragraphs.

Let N workers be $W_0, W_1, \dots, W_{N-1}$. Each worker W_i, $0 \le i \le N-1$, has E model parameter elements $a_i^0, a_i^1, \dots, a_i^{E-1}$, and we number the chunks by $[chk_0], [chk_1], \dots, [chk_{N-1}]$. The elements in every worker are partitioned into N chunk, where $\{a_i^0, \dots, a_i^{\lceil \frac{E}{N} \rceil}\} \in chk_0$, $\dots$, $\{a_i^{(N-1)\lceil \frac{E}{N} \rceil}, \dots, a_i^{E-1}\} \in chk_{N-1}$. The logical ring mode includes the following communication process: $W_0 \to W_1 \to W_2 \to \cdots \to W_{N-1} \to W_0$. We use a generic operator $\oplus$ to denote the reduce operator. The Reduce-Scatter operation is performed as follows. In the first iteration, worker W_i sends chunk $[chk_{(i+N)\%N}]$ to $W_{(i+1)\%N}$. After each worker receives the chunk, all workers have all chunk results $W_0[chk_{(N-1)\%N}], \dots, W_{N-1}[chk_{(2N-2)\%N}]$, where $W_i[chk_{(i-1+N)\%N}] = W_i[chk_{(i-1+N)\%N}] \oplus W_{(i-1+N)\%N}[chk_{(i-1+N)\%N}]$. Let j be the number of iterations, $0 \le j \le N-2$. The result of each worker in the j-th iteration can be expressed as:

$$W_i[chk_{(i-1-j+N)\%N}] = W_i[chk_{(i-1-j+N)\%N}] \oplus W_{(i-1-j+N)\%N}[chk_{(i-1-j+N)\%N}]. \tag{2}$$

We illustrate how the 2D-TGA algorithm operates using an example. Figure 2 shows how to divide **sixteen** workers into **four** groups, and each group has four workers. Four distinct colors represent the four groups. In addition, we set the dimension of the parameter vector on each worker to **16**, and each parameter vector value is composed of the serial number of the worker it corresponds to. Through the Ring AllReduce method after grouping, the size of the ring made by the worker can be significantly reduced, the communication delay can be reduced, and the communication efficiency can be improved. According to the number of workers in the group, the **sixteen** parameters are divided into **four** chunks. The data in a red box indicate the data chunk to be sent. The workers in the first group conduct Ring AllReduce, as shown in Fig. 2a. Leaders do Reduce-Scatter in parallel horizontally, as seen in Fig. 2b. The orange arrow shows the transmission direction of the parameters. The LEFT worker transmits parameters to the RIGHT worker, and each worker gets parameters from the LEFT worker. Figure 2c depicts leaders performing Segmented-Ring operations vertically in parallel. The green arrow shows the direction in which the parameters transfer. The UP worker provides parameters to the DOWN worker, while the DOWN worker gets parameters from the UP worker. Leaders conduct the AllGather operation horizontally in parallel, as shown in Fig. 2d. The first group leader executes the broadcast operation, as shown in Fig. 2e. Algorithm 1 depicts the 2D-TGA algorithm.

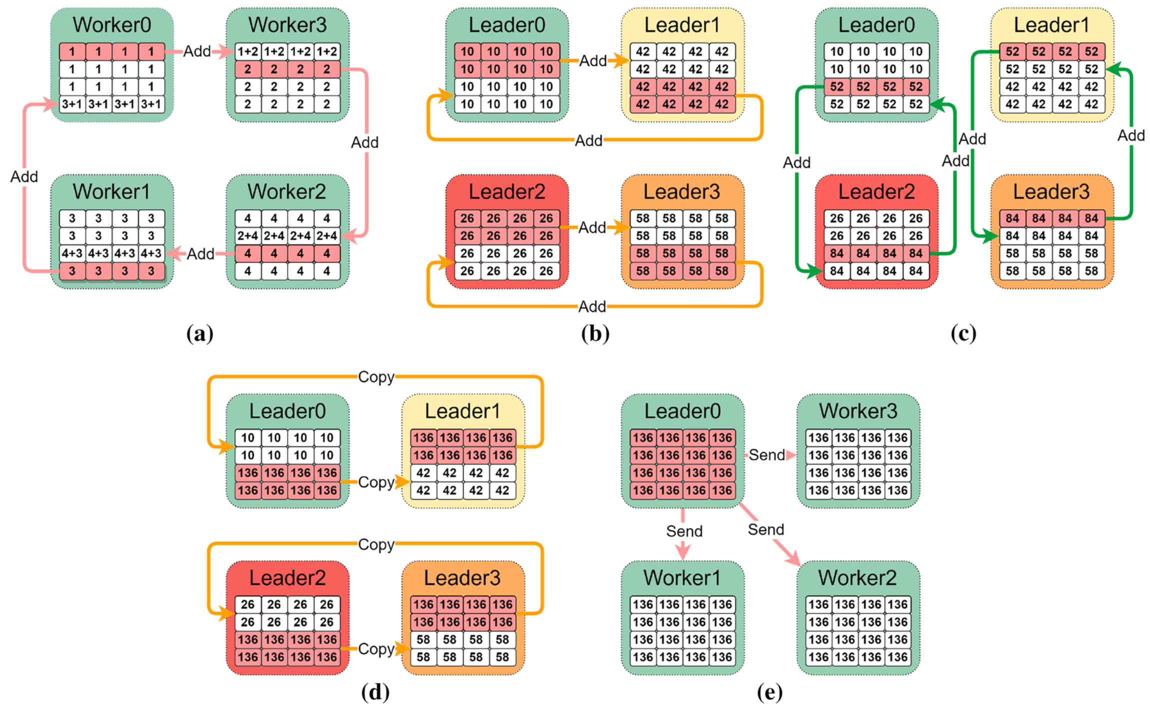


Fig. 2 2D-TGA algorithmic diagram with sixteen workers. **a** The first group performs the Ring AllReduce. **b** Establish Cartesian topology for four leaders and perform Reduce-Scatter operation horizontally. **c** Perform Segmented-Ring operation vertically. **d** Perform AllGather operation horizontally. **e** The first group performs the broadcast operation

```
Algorithm 1 2D-TGA ($W_i$)
Require: Worker numbers $N$, Group numbers $L$, Local parameters of each worker $w_i$
Ensure: Global synchronous parameters $\boldsymbol{w}$
1:  initialize $j$=0;
2:  MPI_Comm_split();/*Hierarchical mode*/
3:  MPI_Group_incl();/*Leader group*/
4:  MPI_Comm_create_group();/*Create subgroup and leadergroup*/
5:  do in parallel
6:    RingAllreduce(); /*Intra-group Ring-Allreduce*/
7:    if $W_i$ ==Leader then
8:      for $j$ < sqrt($L$) do
9:        ReduceScatter($w_i[(i-j-1+N)\%N]$,LEFT,RIGHT,sqrt($L$));/*Split $w_i$ into sqrt($L$) blocks*/
10:     end for
11:     for $j$ < sqrt($L$) do
12:       SegmentedRing($w_i[(i-j-1+N)\%N]$,UP,DOWN,sqrt($L$));
13:     end for
14:     for $j$ < sqrt($L$) do
15:       AllGather($w_i[(i-j-1+N)\%N]$,LEFT,RIGHT,sqrt($L$));
16:     end for
17:     MPI_Bcast();
18:   end if
19: return $\boldsymbol{w}$;
```

4 Algorithm Development

4.1 Distributed ADMM Algorithm

As demonstrated in Equ. (3), a supervised machine learning issue may be abstracted as an optimization problem.

$$\min_{\boldsymbol{x}} l(\boldsymbol{x}) + r(\boldsymbol{x}), \tag{3}$$

where $\boldsymbol{x} \in \mathbb{R}^d$ denotes the model, d denotes the number of sample features , $l(\boldsymbol{x})$ denotes the loss function, and $r(\boldsymbol{x})$ denotes the regularization term. Typically, distributed optimization is usually turned (3) into a consensus problem, as illustrated in (4).

$$\begin{aligned} &\min_{z,\{x_i\}_{i=1}^{N}} \sum_{i=1}^{N} f_i(x_i), \\ &s.t. \quad x_i = z, \forall i. \end{aligned} \tag{4}$$

Write problem (4) in the augmented Lagrangian form:

$$\begin{aligned} L(\{x_i\}, z, \{\lambda_i\}) = &\sum_{i=1}^{N} f_i(x_i) + < \lambda_i, x_i - z > \\ &+ \frac{\rho}{2}\|x_i - z\|_2^2, \end{aligned} \tag{5}$$

where $\{\lambda_i\}$ symbolize Lagrangian multipliers, $\rho > 0$ is the penalty parameter, and $< \cdot, \cdot >$ denotes the inner product. Convergence speed is also influenced by penalty parameter ρ [4]. The value of ρ is verified through experiments [18]. They found that a lower ρ value can make the algorithm converge more quickly. By updating x_i and z iteratively (at the k-th iteration, denoted x_i^k and z^k), $L(\{x_i\}, z, \{\lambda_i\})$ is minimized. In order to integrate the decomposability of dual ascent with the excellent convergence properties of method multipliers, an improved form of optimized ADMM was proposed and use the alternate method to allow the problem to be easily decomposed. The ADMM algorithm update process is as follows:

$$\begin{aligned} x_i^{k+1} := &\arg\min_{x_i}(f_i(x_i) + < \lambda_i^k, x_i - z^k > \\ &+ \frac{\rho}{2}\|x_i - z^k\|_2^2), \end{aligned} \tag{6a}$$

$$z^{k+1} := \frac{1}{N}\sum_{i=1}^{N}(x_i^{k+1} + \frac{1}{\rho}\lambda_i^k), \tag{6b}$$

$$\lambda_i^{k+1} := \lambda_i^k + \rho(x_i^{k+1} - z^{k+1}). \tag{6c}$$

Table 1 Notations

Symbol	Description	Value
N	The total number of workers	32~1280
L	The number of groups	16
α	Delay between two communication workers	$0.7\mu s$
S	The total size of parameters on each worker	3231961
B	The bandwidth capacity of cluster	56 Gb/s
C	Computation time per byte of data	–

4.2 Distributed algorithm 2D-TGA-ADMM

The distributed ADMM algorithm's iterative phase is seen in (6). In distributed systems, the update approach works well. Equations (6a) and (6c) are used parallel by the workers to update the local parameters x_i and λ_i. The global variable z is updated by the total of $(x_i + \frac{\lambda_i}{\rho})$ of all workers, as indicated in Equ. (6b). Thus, we take $(x_i + \frac{\lambda_i}{\rho})$ as a whole and define it as w_i, as shown in Equ. (7)

$$w_i^{k+1} = x_i^{k+1} + \frac{\lambda_i^k}{\rho}. \tag{7}$$

Communication topology is an important factor that affects the scalability of distributed optimization algorithms. In order to minimize the synchronization time of model

Table 2 Global synchronous time of different algorithms

Syn-algorithm	Synchronous time
Ring AllReduce	$2(N-1)(\alpha + \frac{S}{BN}) + (N-1)\frac{SC}{N}$
Hierarchical AllReduce	$2(\frac{N}{L}-1)(\alpha + \frac{LS}{BN}) + (\alpha + \frac{S}{B}) + 2(L-1)(\alpha + \frac{S}{BL})$ $+(\frac{N}{L}-1)\frac{LSC}{N} + (L-1)\frac{SC}{L}$
2D-Torus AllReduce	$2(\sqrt{N}-1)(\alpha + \frac{S}{B\sqrt{N}}) + 2(\sqrt{N}-1)(\alpha + \frac{S}{BN})$ $+(\sqrt{N}-1)\frac{SC}{\sqrt{N}} + (\sqrt{N}-1)\frac{SC}{N}$
2D-TGA	$2(\frac{N}{L}-1)(\alpha + \frac{LS}{BN}) + (\alpha + \frac{S}{B}) + 2(\sqrt{L}-1)(\alpha + \frac{S}{B\sqrt{L}})$ $+2(\sqrt{L}-1)(\alpha + \frac{S}{BL}) + (\sqrt{L}-1)\frac{SC}{\sqrt{L}} + (\sqrt{L}-1)\frac{SC}{L}$

parameters, we adapt the communication-efficient grouping AllReduce based on the two-dimensional torus topology (2D-TGA) proposed in Sect. 3. Next, all w_i in each worker are reduced to $\boldsymbol{w}$ by the synchronous algorithm 2D-TGA. The value of the global variable z is obtained by averaging $\boldsymbol{w}$. The dual variable λ_i is generated by completing (6c).

This paper proposes a distributed algorithm 2D-TGA-ADMM to handle distributed optimization problems, which combines the ADMM algorithm with the communication-efficient 2D-TGA algorithm. Algorithm 2 depicts the algorithm flow. The 2D-TGA-ADMM algorithm can be implemented in a distributed computing environment like MPI or Spark.

Algorithm 2 2D-TGA-ADMM (W_i)

```
Require: The maximum iteration $K$, Group numbers $L$
Ensure: Global parameter $z$ after iterations
1:  initialize $k = 0$, $i = 0$, $w_i^0 = 0$, $x_i^0 = 0$, $\lambda_i^0 = 0$, $z^0 = 0$.
2:  do in parallel
3:    repeat
4:      update $x_i^{k+1}$ by (6a);
5:      update $w_i^{k+1}$ by (7);
6:      excute 2D-TGA($w_i^{k+1}$,$L$);
7:      update $z^{k+1} \leftarrow \frac{\boldsymbol{w}}{N}$;
8:      update $\lambda_i^{k+1}$ by (6c);
9:      $k \leftarrow k + 1$;
10:   until terminal
11: return $z^K$;
```

5 Evaluation and Experiment

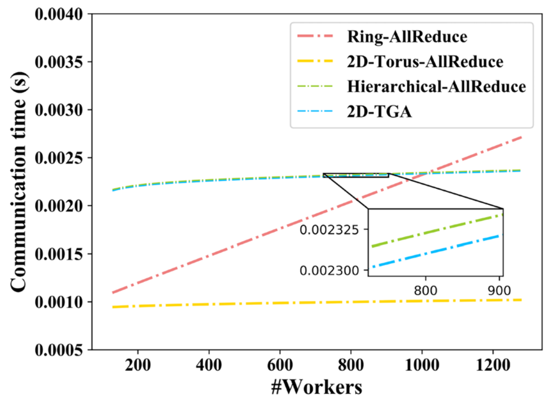
5.1 Evaluation

This paper provides theoretical analysis to compare the AllReduce-based synchronous algorithm's performance, reflecting the theoretical time in a single worker synchronization action. The denotation is provided in Table 1. Based on this denotation, global synchronization time (GST) in different synchronization algorithms can be calculated.

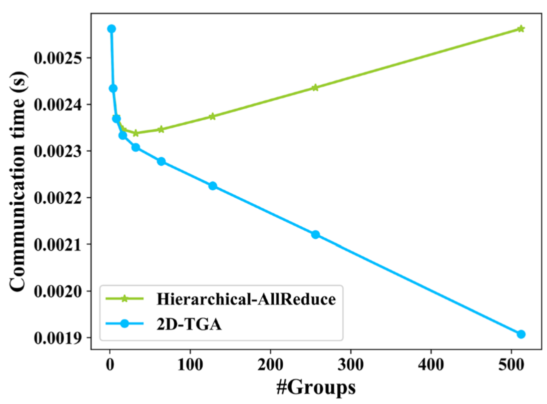
The design of the 2D-TGA algorithm is based on Ring AllReduce, where the synchronization process consists of some Reduction phases (e.g., Scatter-Reduce, and Allgather), each of which is composed of communication steps. The Ring AllReduce algorithm has two phases. The first phase is the Scatter-Reduce, which passes through $N-1$ steps. The communication time of each step is $\alpha + \frac{S}{BN}$, and the calculation time is $\frac{SC}{N}$. The second phase is the Allgather, and the communication time of each step is $\alpha + \frac{S}{BN}$. Overall time consumption is $2(N-1)(\alpha + \frac{S}{BN}) + (N-1)\frac{SC}{N}$. Table 2 shows the synchronous time of different algorithms, including communication time and calculation time. Communication time is shown by the bold formula, whereas the non-bold formula indicates computing time. Figure 3a displays the communication time for Ring-AllReduce, Hierarchical-AllReduce, 2D-Torus-AllReduce, and 2D-TGA algorithms with different #workers. We use the values in Table 1 to get the theoretical analysis results, as shown in Fig. 3a.

In this paper, the global communication time formula is used to calculate the communication time of the corresponding worker, as shown in the bold formula in Table 2. As shown in Fig. 3a, we set #groups to **16**. As #workers increase, the communication time of Ring-AllReduce will grow linearly. The communication time of 2D-TGA and Hierarchical-AllReduce algorithms is comparable and does not rise as #workers increase and maintain steady, which can enhance the scalability of distributed algorithm. Figure 3a also shows that the 2D-TGA method is slightly better than the Hierarchical-AllReduce algorithm.

In addition, both the 2D-TGA and Hierarchical-AllReduce algorithms leverage the concept of grouping. To investigate the impact of grouping on both algorithms, we increase the number of workers to **1024** and examine the impact of varying #groups on the two algorithms over time. Figure 3b shows that when the number of groups increases, the performance of the 2D-TGA is much better than Hierarchical-AllReduce algorithm. The main reason is that the algorithm uses Segmented-Ring operation in the process of vertical AllReduce, which may greatly reduce the communication time. Compared with using the AllReduce algorithm directly, the synchronous time will be

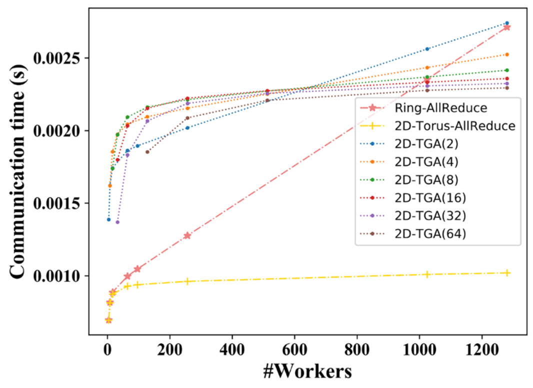


(a) Communication time for different number of workers.

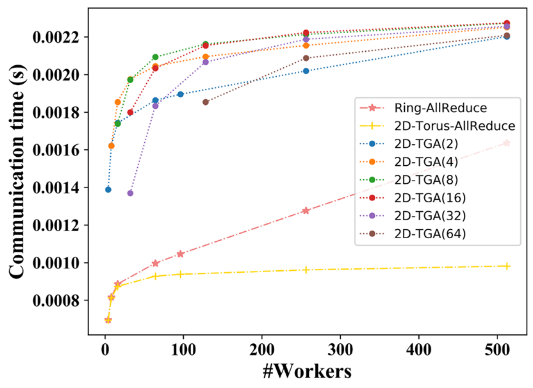


(b) Communication time for different number of groups.

Fig. 3 Communication time of different synchronous algorithm



(a) Communication time in a large-scale clusters.



(b) Communication time in a small-scale clusters.

Fig. 4 Communication time of 2D-TGA algorithm in different groups

reduced, which is also an advantage of the method presented in this paper.

Since grouping impacts the 2D-TGA algorithm, this study estimates the influence of different groups on the algorithm as the number of workers grows. It can be seen from Fig. 4a that grouping is closely related to algorithm performance. As shown in Fig. 4b, the less groups, the better the performance of the algorithm.

5.2 Experiment

Logistic regression is a machine learning method used to solve binary classification problems. To obtain strong generalization abilities, one adds an $\ell 2$ regularization term; in this paper, we consider the following form of regularized logistic regression:

$$\min_{\boldsymbol{x}} f(\boldsymbol{x}) \equiv \sum_{i=1}^{n} log(1 + e^{-b_i \boldsymbol{x}^T D_i}) + \frac{1}{2}\|\boldsymbol{x}\|_2^2, \tag{8}$$

where $\boldsymbol{x} \in \mathbb{R}^d$ represents model parameters, n represents the number of samples, $D_i \in \mathbb{R}^d$ represents the i-th sample, and $b_i \in \{-1, 1\}$ represents the label of the i-th sample.

Experimental Settings. In this section, distributed ADMM algorithm is used to solve the logistic regression problem with the $\ell 2$-norm. Combining the ADMM algorithm with different synchronous algorithms is used to compare the impact of different synchronous algorithms on the scalability of distributed algorithms. To solve sub-problems in distributed ADMM method, we employ the Trust Region Newton method (TRON) [19], and the dataset uses the public dataset `url`[1] and `webspam`[2], as shown in Table 3. The Tianhe-2

[1] https://www.csie.ntu.edu.tw/cjlin/libsvmtools/datasets/binary.html.

[2] https://www.csie.ntu.edu.tw/cjlin/libsvmtools/datasets/binary.html.

Table 3 A summary of datasets

Dataset	#Training samples	#Testing samples	#Features
Url	2156517	239613	3231961
Webspam	337254	37254	16609143

supercomputing cluster serves as the paper's experimental platform. Each node is equipped with two Xeon E5 12-core CPUs and 88 GiB of memory. Our experimental schemes are set as follows: 16 cores×2 nodes, 16 cores×4 nodes, 16 cores×8 nodes, 16 cores×32 nodes. Each node uses 16 cores, and each core runs one process. Each process represents a worker.

Convergence. This paper uses the relative error function (f_{rerr}) to present the convergence of the ADMM algorithm. The definition of f_{rerr} is shown in Equ. (9),

$$f_{rerr} = |f - f^*|/f^*, \tag{9}$$

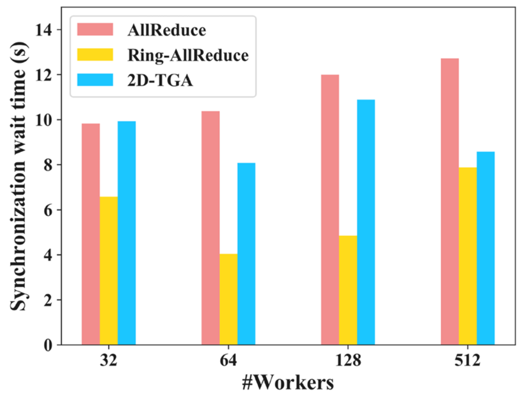
where f represents the value of the objective function in the current state and f^* represents the minimum value of the objective function.

Figures 5b and 6b, respectively, show the convergence of the ADMM algorithm using three different AllReduce-based synchronous algorithms to solve the logistic regression problem under the `url` and `webspam` datasets on 64 workers in 4 nodes. As can be seen from the two figures, the distributed ADMM algorithm based on three synchronous algorithms has the same convergence rate, which means that the three synchronous algorithms do not affect the convergence of the ADMM algorithm. This setting can eliminate interference and test the performance of the three synchronous algorithms more accurately.

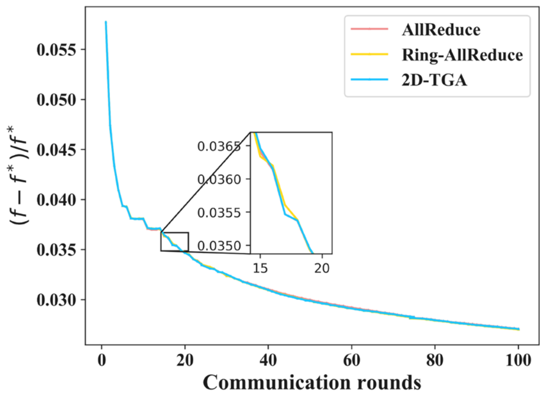
Synchronization Wait Time. Table 4 shows the running time of the experiments, including updating time and synchronization wait time. The updating time refers to the computation time of the TRON method. Due to different calculation speeds of the workers, in a single iteration of the ADMM algorithm, we store the **longest computation time** between workers and accumulate it with the number of iterations of the ADMM algorithm. The synchronization wait time refers to the time for the model parameters w_i to communicate among workers and Reduction operations.

We select different #workers to test the distributed ADMM algorithm based on the three different synchronous algorithms. Figure 5a shows the synchronous overhead of testing the distributed ADMM algorithm on the `url` public dataset. Compared with the `MPI_Allreduce` algorithm in the MPI library, as #workers increase, the 2D-TGA synchronous algorithm can reduce the synchronization wait time by 32.6%. However, this algorithm still has drawbacks compared with the Ring-AllReduce synchronous algorithm on the `url` dataset. As shown in Fig. 5a, we also find that, as #workers increase, the synchronization wait time of the 2D-TGA gradually approaches to the Ring-AllReduce algorithm. This is also the same as the theoretical analysis in Fig. 3a. As #workers increase, the communication time of the Ring-AllReduce algorithm is higher than the 2D-TGA synchronous algorithm proposed in this paper.

In distributed machine learning, the dimension of the model is the decisive factor of the communication volume. In order to evaluate the effectiveness of the 2D-TGA synchronous algorithm, we test it on the high-dimensional `webspam` dataset. As shown in Fig. 6a, comparing the collection communication algorithm `MPI_Allreduce` and Ring-AllReduce algorithms, the synchronization wait time of the 2D-TGA under different #workers has apparent advantages, and the efficiency can be improved by 33.8% compared with the collection communication algorithm `MPI_Allreduce`. Why do the Ring-AllRedcue and 2D-TGA have different performances on datasets with different dimensions? Through the theoretical analysis of Fig. 3a,

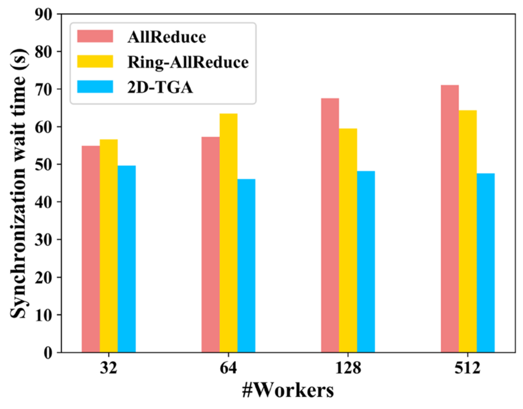


(a) Synchronization wait time of ADMM algorithm based on different synchronous algorithms.

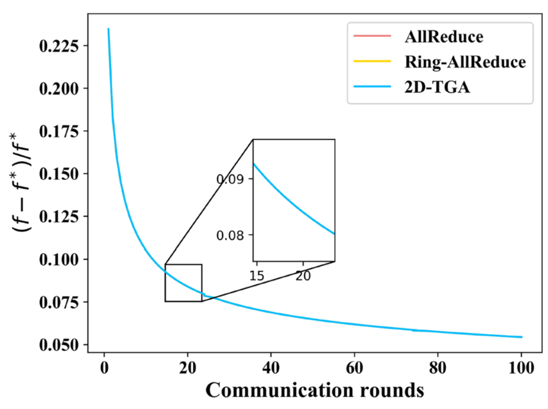


(b) Convergence of the relative error with iterations.

Fig. 5 Experimental results of `url` dataset



(a) Synchronous wait time of ADMM algorithm based on different synchronous algorithms.



(b) Convergence of the relative error with iterations.

Fig. 6 Experimental results of `webspam` dataset

it can be found that as #workers increase, the performance of the Ring-AllReduce algorithm gets worse and worse than the 2D-TGA algorithm. Figure 6a also shows that the 2D-TGA algorithm is more suited to high-dimensional datasets than the Ring-AllReduce technique.

Through the theoretical analysis in Fig. 4, we find that as #workers increase, the number of groups also affects the performance of the 2D-TGA algorithm. As shown in Fig. 4b, theoretical analysis demonstrates that as the increase in # workers, the more #groups, the worse the performance of the algorithm. In this paper, 512 workers are evaluated using the `url` and `webspam` datasets, as illustrated in Fig. 7. 4, 16, and 64 groups are set to test the 2D-TGA algorithm. Four groups are preferable to sixteen, which is supported by the theoretical analysis. Furthermore, Fig. 4a demonstrates that the number of workers also determines the size of the grouping.

6 Conclusion

Synchronization of model parameters is critical in DML. With the increase in model parameters and #workers, the parameter synchronization mechanism will become an important factor that limits the scalability of the DML. In this paper, a new synchronous algorithm 2D-TGA is proposed, which can shorten the synchronous time by effectively utilizing the network bandwidth. Firstly, we introduce the topology of this algorithm. Then, we analyze the synchronous time of four AllReduce-based algorithms. The results show that the performance of 2D-TGA is good. To verify our analysis, we propose an ADMM-based distributed algorithm named 2D-TGA-ADMM, which combines the ADMM algorithm and 2D-TGA. We test it on the Tianhe-2 supercomputing platform to solve logistic regression problems. The synchronization cost of 2D-TGA is verified by selecting different datasets and #workers. Experimental results show that compared with the previous algorithms, this synchronous algorithm can reduce the synchronization wait time by 33%. With the increase in the number of workers, the 2D-TGA algorithm will neither increase the synchronization cost nor affect the convergence of the numerical algorithm. It is well suited to large-scale distributed machine learning.

This paper only studies the distributed ADMM algorithm to solve the logistic regression problem for sparse datasets. The next step is to test the algorithm on new unseen data, so as to improve the generalization ability. Since the distributed

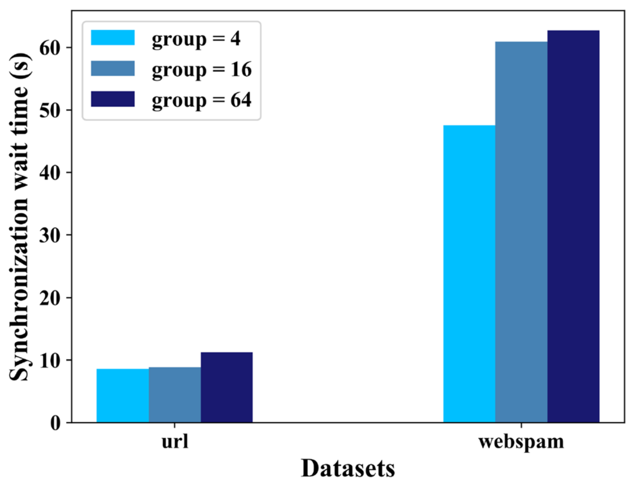


Fig. 7 Synchronization wait time of different #groups on the 2D-TGA-ADMM algorithm

Table 4 Training time of the ADMM algorithm based on various All-Reduce-based synchronous algorithms for different datasets

Dataset	#Workers	Syn-algorithm	Updating time (s)	Synchronous wait time (s)
Url	32	AllReduce	369.93	9.82
		Ring-All-Reduce	396.80	6.58
		2D-TGA	369.84	9.93
	64	AllReduce	348.80	10.38
		Ring-All-Reduce	323.14	4.04
		2D-TGA	331.46	8.08
	128	AllReduce	275.65	12
		Ring-All-Reduce	277.64	4.85
		2D-TGA	279.38	10.89
	512	AllReduce	212.55	12.72
		Ring-All-Reduce	214.33	7.88
		2D-TGA	212.24	8.58
Webs-pam	32	AllReduce	1113.81	54.91
		Ring-All-Reduce	1215.15	56.61
		2D-TGA	1109.43	53.72
	64	AllReduce	819.1	49.64
		Ring-All-Reduce	816.18	63.48
		2D-TGA	818	46.07
	128	AllReduce	862.51	67.6
		Ring-All-Reduce	855.47	59.5
		2D-TGA	859.58	48.2
	512	AllReduce	754.5	71.1
		Ring-All-Reduce	749.23	64.39
		2D-TGA	747.71	47.57

ADMM algorithm has a higher computing and communication ratio, future research will further reduce the synchronization wait time of the distributed ADMM algorithm.

Acknowledgements Thanks to the National Supercomputer Center in Guangzhou for providing computing resources and technical support.

Author Contributions GW involved in conceptualization, methodology, software, formal analysis, writing—original draft, visualization. YL took part in writing—review & editing, project administration, project administration, funding acquisition. ZZ took part in data curation, supervision. CP involved in validation, resources, formal analysis.

Data Availability The data that support the findings of this study are available in LIBSVM at https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary.html.

Declarations

Conflict of interest The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

1. Chen Y, Blum RS, Sadler BM (2022) Communication efficient federated learning via ordered admm in a fully decentralized setting. arXiv preprint arXiv:2202.02580
2. Wang X, Ishii H, Du L, Cheng P, Chen J (2020) Privacy-preserving distributed machine learning via local randomization and admm perturbation. IEEE Trans Signal Proc 68:4226–4241
3. Raja G, Anbalagan S, Vijayaraghavan G, Theerthagiri S, Suryanarayan SV, Wu X-W (2020) Sp-cids: secure and private collaborative ids for vanets. IEEE Trans Int Trans Syst 22(7):4385–4393
4. Steck H, Dimakopoulou M, Riabov N, Jebara T (2020) Admm slim: sparse recommendations for many users. In: Proceedings of the 13th international conference on web search and data mining, pp 555–563
5. Verbraeken J, Wolting M, Katzy J, Kloppenburg J, Verbelen T, Rellermeyer JS (2020) A survey on distributed machine learning. ACM Comput Surv (CSUR) 53(2):1–33
6. Hasanov K, Lastovetsky A (2017) Hierarchical redesign of classic mpi reduction algorithms. J Supercomput 73(2):713–725
7. Wang D, Lei Y, Xie J, Wang G (2021) Hsac-aladmm: an asynchronous lazy admm algorithm based on hierarchical sparse allreduce communication. J Supercomput 77(8):8111–8134
8. Xie J, Lei Y (2019) Admmlib: a library of communication-efficient ad-admm for distributed machine learning. In: IFIP international conference on network and parallel computing. Springer, pp 322–326
9. Sanders P, Speck J, Träff JL (2009) Two-tree algorithms for full bandwidth broadcast, reduction and scan. Parallel Comput 35(12):581–594
10. Graham RL, Barrett BW, Shipman GM, Woodall TS, Bosilca G (2007) Open mpi: A high performance, flexible implementation of mpi point-to-point communications. Parallel Process Lett 17(01):79–88
11. Patarasuk P, Yuan X (2009) Bandwidth optimal all-reduce algorithms for clusters of workstations. J Parallel Distrib Comput 69(2):117–124

12. Research B (2017) baidu-allreduce. [Online]. https://github.com/baidu-research/baidu-allreduce
13. Mikami H, Suganuma H, Tanaka Y, Kageyama Y, et al (2018) Massively distributed sgd: imagenet/resnet-50 training in a flash. arXiv preprint arXiv:1811.05233
14. Ying C, Kumar S, Chen D, Wang T, Cheng Y (2018) Image classification at supercomputer scale. arXiv preprint arXiv:1811.06992
15. Jia X, Song S, He W, Wang Y, Rong H, Zhou F, Xie L, Guo Z, Yang Y, Yu L, et al Highly scalable deep learning training system with mixed-precision: training imagenet in four minutes. arXiv preprint arXiv:1807.11205
16. Goyal P, Dollár P, Girshick R, Noordhuis P, Wesolowski L, Kyrola A, Tulloch A, Jia Y, He K (2017) Accurate, large minibatch sgd: training imagenet in 1 hour. arXiv preprint arXiv:1706.02677
17. Ueno Y, Yokota R (2019) Exhaustive study of hierarchical allreduce patterns for large messages between gpus. In: 2019 19th IEEE/ACM international symposium on cluster, cloud and grid computing (CCGRID). IEEE, pp 430–439
18. Sun DL, Fevotte C (2014) Alternating direction method of multipliers for non-negative matrix factorization with the beta-divergence. In: 2014 IEEE international conference on acoustics, speech and signal processing (ICASSP). IEEE, pp 6201–6205
19. Lin C-J, Weng RC, Keerthi SS (2008) Trust region newton method for large-scale logistic regression. J Mach Learn Res, 9(4):627-650