Fast and Secure Distributed Nonnegative Matrix Factorization

Yuqiu Qian, Conghui Tan, Danhao Ding, Hui Li, and Nikos Mamoulis

Abstract—Nonnegative matrix factorization (NMF) has been successfully applied in several data mining tasks. Recently, there is an increasing interest in the acceleration of NMF, due to its high cost on large matrices. On the other hand, the privacy issue of NMF over federated data is worthy of attention, since NMF is prevalently applied in image and text analysis which may involve leveraging privacy data (e.g, medical image and record) across several parties (e.g., hospitals). In this paper, we study the *acceleration* and *security* problems of distributed NMF. Firstly, we propose a *distributed sketched alternating nonnegative least squares* (DSANLS) framework for NMF, which utilizes a matrix sketching technique to reduce the size of nonnegative least squares subproblems with a convergence guarantee. For the second problem, we show that DSANLS with modification can be adapted to the security setting, but only for *one or limited iterations.* Consequently, we propose four efficient distributed NMF methods in both synchronous and asynchronous settings with a security guarantee. We conduct extensive experiments on several real datasets to show the superiority of our proposed methods. The implementation of our methods is available at https://github.com/qianyuqiu79/DSANLS.

Index Terms—Distributed Nonnegative Matrix Factorization, Matrix Sketching, Privacy

1 Introduction

Nonnegative matrix factorization (NMF) is a technique for discovering nonnegative latent factors and/or performing dimensionality reduction. Unlike general matrix factorization (MF), NMF restricts the two output matrix factors to be nonnegative. Specifically, the goal of NMF is to decompose a huge matrix $M \in \mathbb{R}_{+}^{m\times n}$ into the product of two matrices $U \in \mathbb{R}_{+}^{m\times k}$ and $V \in \mathbb{R}_{+}^{n\times k}$ such that $M \approx UV^{\top}$. $\mathbb{R}_{+}^{m\times n}$ denotes the set of $m \times n$ matrices with nonnegative real values, and k is a user-specified dimensionality, where typically $k \ll m, n$. Nonnegativity is inherent in the feature space of many real-world applications, where the resulting factors of NMF can have a natural interpretation. Therefore, NMF has been widely used in a branch of fields including text mining [1], image/video processing [2], recommendation [3], and analysis of social networks [4].

Modern data analysis tasks apply on big matrix data with increasing scale and dimensionality. Examples [5] include community detection in a billion-node social network, background separation on a 4K video in which every frame has approximately 27 million rows, and text mining on a bag-of-words matrix with millions of words. The volume of data is anticipated to increase in the 'big data' era, making it impossible to store the whole matrix in the main memory throughout NMF. Therefore, there is a need for high-performance and scalable distributed NMF algorithms. On the other hand, there is a surge of works on privacy-preserving data mining over federated data [6, 7] in recent years. In contrast to traditional research about privacy which emphasizes protecting individual information from single institution, federated data mining deals with multiple parties. Each party possesses its own confidential dataset(s) and the union of data from all parties is utilized for achieving better performance in the target task. Due to the prevalent use of NMF in image and text analysis which may involve leveraging privacy data (e.g, medical image and record) across several parties (e.g., hospitals), the privacy issue of NMF over federated data is worthy of attention. To address aforementioned challenges of NMF (i.e., high performance and privacy), we study the *acceleration* and *security* problems of distributed NMF in this paper.

First of all, we propose the *distributed sketched alternating nonnegative least squares* (DSANLS) for accelerating NMF. The state-of-the-art distributed NMF is MPI-FAUN [8], a general framework that iteratively solves the nonnegative least squares (NLS) subproblems for U and V. The main idea behind MPI-FAUN is to exploit the independence of local updates for rows of U and V, in order to minimize the communication requirements of matrix multiplication operations within the NMF algorithms. Unlike MPI-FAUN, our idea is to speed up distributed NMF in a new, orthogonal direction: by reducing the problem size of each NLS subproblem within NMF, which in turn decreases the overall computation cost. In a nutshell, we reduce the size of each NLS subproblem, by employing a *matrix sketching* technique: the involved matrices in the subproblem are multiplied by a specially designed random matrix at each iteration, which greatly reduces their dimensionality. As a result, the computational cost of each subproblem significantly drops.

However, applying matrix sketching comes with several issues. First, although the size of each subproblem is sig-

- *Yuqiu Qian is with Tencent, Shenzhen, China. E-mail: yuqiuqian@tencent.com.*
- *Conghui Tan is with WeBank, Shenzhen, China. E-mail: tanconghui@gmail.com.*
- *Danhao Ding is with Department of Computer Science, University of Hong Kong, Hong Kong SAR, China. E-mail: dhding2@cs.hku.hk.*
- *Hui Li is with School of Informatics, Xiamen University, Xiamen, Fujian, China. E-mail: hui@xmu.edu.cn. He is the corresponding author.*
- *Nikos Mamoulis is with Department of Computer Science and Engineering, University of Ioannina, Ioannina, Epirus, Greece. E-mail: nikos@cs.uoi.gr.*

nificantly reduced, sketching involves matrix multiplication which brings computational overhead. Second, unlike in a single machine setting, data is distributed to different nodes in distributed environment. Nodes may have to communicate extensively in a poorly designed solution. In particular, each node only retains part of both the input matrix and the generated approximate matrices, causing difficulties due to data dependencies in the computation process. Besides, the generated random matrices should be the same for all nodes in every iteration, while broadcasting the random matrix to all nodes brings severe communication overhead and can become the bottleneck of distributed NMF. Furthermore, after reducing each original subproblem to a sketched random new subproblem, it is not clear whether the algorithm still converges and whether it converges to stationary points of the original NMF problem.

Our DSANLS overcomes these problems. Firstly, the extra computation cost due to sketching is reduced with a proper choice of the random matrices. Then, the same random matrices used for sketching are generated independently at each node, thus there is no need for transferring them among nodes during distributed NMF. Having the complete random matrix at each node, an NMF iteration can be done locally with the help of a matrix multiplication rule with proper data partitioning. Therefore, our matrix sketching approach reduces not only the computational overhead, but also the communication cost. Moreover, due to the fact that sketching also *shifts* the optimal solution of each original NMF subproblem, we propose subproblem solvers paired with theoretical guarantees of their convergence to a stationary point of the original subproblems.

To provide solutions to the problem of secure distributed NMF over federated data, we first show that DSANLS with modification can be adapted to this security setting, but only for *one or limited iterations*. Therefore, we design new methods called *Syn-SD* and *Syn-SSD* in synchronous setting. They are later extended to *Asyn-SD* and *Asyn-SSD* in asynchronous setting (i.e., client/server), respectively. Syn-SSD improves the convergence rate of Syn-SD, without incurring much extra communication cost. It also reduces computational overhead by *sketching*. All proposed algorithms are secure with a guarantee. Secure distributed NMF problem is hard in nature. All parties involved should not be able to infer the confidential information during the process. To the best of our knowledge, we are the first to study NMF over federated data.

In summary, our contributions are as follows:

- DSANLS is the first distributed NMF algorithm that leverages matrix sketching to reduce the problem size of each NLS subproblem and can be applied to both dense and sparse input matrices with a convergence guarantee.
- We propose a novel and specially designed subproblem solver (*proximal coordinate descent*), which helps DSANLS converge faster. We also discuss the use of *projected gradient descent* as subproblem solver, showing that it is equivalent to stochastic gradient descent (SGD) on the original (non-sketched) NLS subproblem.
- For the problem of secure distributed NMF, we propose efficient methods, Syn-SD and Syn-SSD, in synchronous setting and later extend them to asynchronous setting. Through sketching, their computation cost is significantly reduced. They are the first secure distributed NMF methods for federated data.
- We conduct extensive experiments using several (dense and sparse) real datasets, which demonstrates the efficiency and scalability of our proposals.

Algorithm 1 Two-Block Coordinate Descent: Framework of Most NMF Algorithms

```
Input: $M$
Parameter: Iteration number $T$
1: initialize $U^0 \geq 0$, $V^0 \geq 0$
2: for $t = 0$ to $T - 1$ do
3:     $U^{t+1} \leftarrow$ update($M$, $U^t$, $V^t$)
4:     $V^{t+1} \leftarrow$ update($M$, $U^{t+1}$, $V^t$)
5: return $U^T$ and $V^T$
```

The remainder of the paper is organized as follows. Sec. 2 provides the background and discusses the related work. Our DSANLS algorithm with detailed theoretical analysis is presented in Sec. 3. Our proposed algorithms for secure distributed NMF problem in both synchronous and asynchronous settings are presented in Sec. 4. Sec. 5 evaluates all algorithms. Finally, Sec. 6 concludes the paper.

Notations. For a matrix A, we use $A_{i:j}$ to denote the entry at the i-th row and j-th column of A. Besides, either i or j can be omitted to denote a column or a row, i.e., $A_{i:}$ is the i-th row of A, and $A_{:j}$ is its j-th column. Furthermore, i or j can be replaced by a subset of indices. For example, if $I \subset \{1, 2, \ldots, m\}$, $A_{I:}$ denotes the sub-matrix of A formed by all rows in I, whereas $A_{:J}$ is the sub-matrix of A formed by all columns in a subset $J \subset \{1, 2, \ldots, n\}$.

2 Background and Related Work

In Sec. 2.1, we first illustrate NMF and its security problem in a distributed environment. Then we elaborate on previous works which are related to this paper in Sec. 2.2.

2.1 Preliminary

2.1.1 NMF Algorithms

Generally, NMF can be defined as an optimization problem [9] as follows:

$$\min_{U \in \mathbb{R}_+^{m \times k}, V \in \mathbb{R}_+^{n \times k}} \left\| M - UV^\top \right\|_F, \tag{1}$$

where $||X||_F = \left(\sum_{ij} x_{ij}^2\right)^{1/2}$ is the Frobenius norm of X. Problem (1) is hard to solve directly because it is non-convex. Therefore, almost all NMF algorithms leverage two-block coordinate descent schemes (shown in Alg. 1): they optimize over one of the two factors, U or V, while keeping the other fixed [10]. By fixing V, we can optimize U by solving a nonnegative least squares (NLS) subproblem:

$$\min_{U \in \mathbb{R}_+^{m \times k}} \left\| M - UV^\top \right\|_F. \tag{2}$$

Similarly, if we fix U, the problem becomes:

$$\min_{V \in \mathbb{R}_+^{n \times k}} \left\| M^\top - VU^\top \right\|_F. \tag{3}$$

Within two-block coordinate descent schemes (exact or inexact), different subproblem solvers are proposed. The first widely used update rule is Multiplicative Updates (MU) [9, 11]. MU is based on the majorization-minimization framework and its application guarantees that the objective function monotonically decreases [9, 11]. Another extensively studied method is alternating nonnegative least squares (ANLS), which represents a class of methods where the subproblems for U and V are solved exactly following the framework described in Alg. 1. ANLS is guaranteed to converge to a stationary point [12] and has been shown to perform very well in practice with active set [13, 14], projected gradient [15], quasi-Newton [16], or accelerated gradient [17] methods as the subproblem solver. Therefore, we focus on ANLS in this paper.

2.1.2 Secure Distributed NMF

Secure distributed NMF problem is meaningful with practical applications. Suppose two hospitals A and B have different clinical records, M_1 and M_2 (i.e., matrices), for same set of phenotypes. For legal or commercial concerns, it is required that none of the hospitals can reveal personal records to another directly. For the purpose of phenotype classification, NMF task can be applied independently (i.e., $M_1 \approx U_1 V_1^\top$ and $M_2 \approx U_2 V_2^\top$). However, since M_1 and M_2 have the same schema for phenotypes, the concatenated matrix $M = [M_1, M_2]$ can be taken as input for NMF and results in better user (i.e., patients) latent representations V_1 and V_2 by sharing the same item (i.e., phenotypes) latent representation U:

$$M = \begin{bmatrix} M_1 & M_2 \end{bmatrix} \approx \begin{bmatrix} UV_1^\top & UV_2^\top \end{bmatrix} = U \cdot \begin{bmatrix} V_1^\top & V_2^\top \end{bmatrix}. \quad (4)$$

Throughout the factorization process, a *secure* distributed NMF should guarantee that party A only has access to M_1, U and V_1 and party B only has access to M_2, U and V_2. It is worth noting that the above problem of distributed NMF with two parties can be straightforwardly extended to N parties. The requirement of all parties over federated data in secure distributed NMF is actual the so-called *t-private protocol* (shown in Definition 1 with $t = N-1$) which derives from secure function evaluation [18]. In this paper, we will use it to assess whether a distributed NMF is *secure*.

Definition 1. *(t-private protocol). All N parties follow the protocol honestly, but they are also curious about inferring other party's private information based on their own data (i.e., honest-but-curious). A protocol is t-private if any t parties who collude at the end of the protocol learn nothing beyond their own outputs.*

Note that a single matrix transpose operation transforms a column-concatenated matrix to a row-concatenated matrix. Without loss of generality, we only consider the scenario that matrices are concatenated along rows in this paper.

Secure distributed NMF problem is hard in nature. Firstly, party A needs to solve the NMF problem to get U and V_1 together with party B. At the same time, party A should not be able to infer V_2 or M_2 during the whole process. Such secure requirement makes it totally different from traditional distributed NMF problem, whose data partition already incurs secure violation.

2.2 Related Work

In the sequel, we briefly review three research areas which are related to this paper.

2.2.1 Accelerating NMF

Parallel NMF algorithms are well studied in the literature [19, 20]. However, different from a parallel and single machine setting, data sharing and communication have considerable cost in a distributed setting. Therefore, we need specialized NMF algorithms for massive scale data handling in a distributed environment. The first method in this direction [21] is based on the MU algorithm. It mainly focuses on sparse matrices and applies a careful partitioning of the data in order to maximize data locality and parallelism. Later, CloudNMF [22], a MapReduce-based NMF algorithm similar to [21], was implemented and tested on large-scale biological datasets. Another distributed NMF algorithm [23] leverages block-wise updates for local aggregation and parallelism. It also performs frequent updates using whenever possible the most recently updated data, which is more efficient than traditional concurrent counterparts. Apart from MapReduce implementations, Spark is also attracting attention for its advantage in iterative algorithms, e.g., using MLlib [24]. Finally, there are implementations using X10 [25] and on GPU [26].

The most recent and related work in this direction is MPI-FAUN [5, 8], which is the first implementation of NMF using MPI for interprocessor communication. MPI-FAUN is flexible and can be utilized for a broad class of NMF algorithms that iteratively solve NLS subproblems including MU, HALS, and ANLS/BPP. MPI-FAUN exploits the independence of local update computation for rows of U and V to apply communication-optimal matrix multiplication. In a nutshell, the full matrix M is split across a two-dimensional grid of processors and multiple copies of both U and V are kept at different nodes, in order to reduce the communication between nodes during the iterations of NMF algorithms.

2.2.2 Matrix Sketching

Matrix sketching is a technique that has been previously used in numerical linear algebra [27], statistics [28] and optimization [29]. Its basic idea is described as follows. Suppose we need to find a solution x to the equation: $Ax = b, \quad (A \in \mathbb{R}^{m \times n},\ b \in \mathbb{R}^m)$. Instead of solving this equation directly, in each iteration of matrix sketching, a random matrix $S \in \mathbb{R}^{d \times m}$ $(d \ll m)$ is generated, and we instead solve the following problem: $(SA)x = Sb$. Obviously, the solution to the first equation is also a solution to the second equation, but not vice versa. However, the problem size has now decreased from $m \times n$ to $d \times n$. With a properly generated random matrix S and an appropriate method to solve subproblem in the second equation, it can be guaranteed that we will progressively approach the solution to the first equation by iteratively applying this sketching technique.

To the best of our knowledge, there is only one piece of previous work [30] which incorporates dual random projection into the NMF problem, in a centralized environment, sharing similar ideas as SANLS, the centralized version of our DSANLS algorithm. However, Wang et al. [30] did not provide an efficient subproblem solver, and their method was less effective than non-sketched methods in practical experiments. Besides, data sparsity was not taken into consideration in their work. Furthermore, no theoretical guarantee was provided for NMF with dual random projection. In short, SANLS is not same as [30] and DSANLS is much more than a distributed version of [30]. The methods that we propose in this paper are efficient in practice and have strong theoretical guarantees.

2.2.3 Secure Matrix Computation on Federated Data

In federated data mining, parties collaborate to perform data processing task on the union of their unencrypted data, without leaking their private data to other participants [31]. A surge of work in the literature studies federated matrix computation algorithms, such as privacy-preserving gradient descent [32, 33], eigenvector computation [34], singular value decomposition [35, 36], *k*-means clustering [37], and spectral clustering [38] over partitioned data on different parties. Secure multi-party computation (MPC) are applied to preserve the privacy of the parties involved (e.g. secure addition, secure multiplication and secure dot product) [37, 39]. These Secure MPC protocols compute arbitrary function among n parties and tolerate up to $t < (1/2)n$ corrupted parties, at a cost $\Omega(n)$ per bit [40, 41]. These protocols are too generic when it comes to a specific task like secure NMF. Our proposed protocol does not incorporate costly MPC multiplication protocols while tolerates up to n-1 corrupted (static, honest but curious) parties. Recently, Kim et al. [6] proposed a federated method to learn phenotypes across multiple hospitals with alternating direction method of multipliers (ADMM) tensor factorization; and Feng et al. [7] developed a privacy-preserving tensor decomposition framework for processing encrypted data in a federated cloud setting.

3 DSANLS: Distributed Sketched ANLS

In this section, we illustrate our DSANLS method for accelerating NMF in general distributed environment.

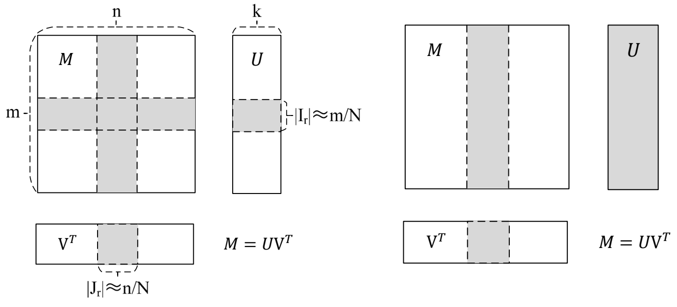
3.1 Data Partitioning

Assume there are N computing nodes in the cluster. We partition the row indices $\{1, 2, \dots, m\}$ of the input matrix M into N disjoint sets $I_1, I_2, \dots, I_N$, where $I_r \subset \{1, 2, \dots, m\}$ is the subset of rows assigned to node r, as in [21]. Similarly, we partition the column indices $\{1, 2, \dots, n\}$ into disjoint sets $J_1, J_2, \dots, J_N$ and assign column set J_r to node r. The number of rows and columns in each node are near the same in order to achieve load balancing, i.e., $|I_r| \approx m/N$ and $|J_r| \approx n/N$ for each node r. The factor matrices U and V are also assigned to nodes accordingly, i.e., node r stores and updates $U_{I_r:}$ and $V_{J_r:}$ as shown in Fig. 1(a).

Data partitioning in distributed NMF differs from that in parallel NMF. Previous works on parallel NMF [19, 20] choose to partition U and V along the long dimension, but we adopt the row-partitioning of U and V as in [21]. To see why, take the U-subproblem (2) as an example and observe that it is row-independent in nature, i.e., the r-th row block of its solution $U_{I_r:}$ is given by:

$$U_{I_r:} = \underset{U_{I_r:} \in \mathbb{R}_+^{|I_r| \times k}}{\arg\min} \left\| M_{I_r:} - U_{I_r:} V^\top \right\|_F^2, \tag{5}$$

and thus can be solved independently without referring to any other row blocks of U. The same holds for the V-subproblem. In addition, no communication is needed concerning M when solving (5) because $M_{I_r:}$ is already present in node r.



(a) DSANLS

(b) Secure NMF

Fig. 1. Partitioning data to N nodes with node r's data shaded.

On the other hand, solving (5) requires the entire V of size $n \times k$, meaning that every node needs to gather V from all other nodes. This process can easily be the bottleneck of a naive distributed ANLS implementation. As we will explain shortly, our DSALNS algorithm alleviates this problem, since we use a sketched matrix of reduced size instead of the original complete matrix V.

3.2 SANLS: Sketched ANLS

To better understand DSANLS, we first introduce the Sketched ANLS (SANLS), i.e., a centralized version of our algorithm. Recall that in Sec. 2.1.1, at each step of ANLS, either U or V is fixed and we solve a nonnegative least square problem (2) or (3) over the other variable. Intuitively, it is unnecessary to solve this subproblem with high accuracy, because we may not have reached the optimal solution for the fixed variable so far. Hence, when the fixed variable changes in the next step, this accurate solution from the previous step will not be optimal anymore and will have to be re-computed. Our idea is to apply matrix sketching for each subproblem, in order to obtain an approximate solution for it at a much lower computational and communication cost.

Specifically, suppose we are at the t-th iteration of ANLS, and our current estimations for U and V are U^t and V^t respectively. We must solve subproblem (2) in order to update U^t to a new matrix U^{t+1}. We apply matrix sketching to the residual term of subproblem (2). The subproblem now becomes:

$$\min_{U \in \mathbb{R}_+^{m \times k}} \left\| M S^t - U \left(V^{t\top} S^t \right) \right\|_F^2, \tag{6}$$

where $S^t \in \mathbb{R}^{n \times d}$ is a randomly-generated matrix. Hence, the problem size decreases from $n \times k$ to $d \times k$. d is chosen to be much smaller than n, in order to sufficiently reduce the computational cost[1]. Similarly, we transform the V-subproblem into:

$$\min_{V \in \mathbb{R}_{+}^{n \times k}} \left\| M^{\top} S'^{t} - V \left(U^{t\top} S'^{t} \right) \right\|_{F}^{2}, \tag{7}$$

where $S'^{t} \in \mathbb{R}^{m \times d'}$ is also a random matrix with $d' \ll m$.

3.3 DSANLS: Distributed SANLS

Now, we come to our proposal: the distributed version of SANLS called DSANLS. Since the U-subproblem (6) is the same as the V-subproblem (7) in nature, here we restrict our attention to the U-subproblem. The first observation about subproblem (6) is that it is still row-independent, thus node r only needs to solve:

$$\min_{U_{I_r:} \in \mathbb{R}_{+}^{|I_r| \times k}} \left\| \left(M S^{t} \right)_{I_r:} - U_{I_r:} \left(V^{t\top} S^{t} \right) \right\|_{F}^{2}. \tag{8}$$

For simplicity, we denote:

$$A_r^t \triangleq \left(M S^t \right)_{I_r:} \quad \text{and} \quad B^t \triangleq V^{t\top} S^t, \tag{9}$$

and the subproblem (8) can be written as:

$$\min_{U_{I_r:} \in \mathbb{R}_{+}^{|I_r| \times k}} \left\| A_r^t - U_{I_r:} B^t \right\|_{F}^{2}. \tag{10}$$

Thus, node r needs to know matrices A_r^t and B^t in order to solve the subproblem.

For A_r^t, by applying matrix multiplication rules, we get $A_r^t = \left(M S^t \right)_{I_r:} = M_{I_r:} S^t$. Therefore, if S^t is stored at node r, A_r^t can be computed without any communication. On the other hand, computing $B^t = \left(V^{t\top} S^t \right)$ requires communication across the whole cluster, since the rows of V^t are distributed across different nodes. Fortunately, if we assume that S^t is stored at all nodes again, we can compute B^t in a much cheaper way. Following block matrix multiplication rules, we can rewrite B^t as:

$$B^t = V^{t\top} S^t = \left[\left(V_{J_1:}^t \right)^{\top} \cdots \left(V_{J_N:}^t \right)^{\top} \right] \begin{bmatrix} S_{J_1:}^t \\ \vdots \\ S_{J_N:}^t \end{bmatrix} = \sum_{r=1}^{N} \left(V_{J_r:}^t \right)^{\top} S_{J_r:}^t. \tag{11}$$

Note that the summand $\bar{B}_r^t \triangleq \left(V_{J_r:}^t \right)^{\top} S_{J_r:}^t$ is a matrix of size $k \times d$ and can be computed locally. As a result, communication is only needed for summing up the matrices $\bar{B}_r^t$ of size $k \times d$ by using MPI all-reduce operation, which is much cheaper than transmitting the whole V_t of size $n \times k$.

Algorithm 2 Distributed SANLS on Node r

```
Input: $M_{I_r:}$ and $M_{:J_r}$
Parameter: Iteration number $T$
Initialize $U_{I_r:}^0 \geq 0$, $V_{J_r:}^0 \geq 0$
Broadcast the random seed
for $t = 0$ to $T - 1$ do
    Generate random matrix $S^t \in \mathbb{R}^{n \times d}$
    Compute $A_r^t \leftarrow M_{I_r:} S^t$
    Compute $\bar{B}_r^t \leftarrow \left( V_{J_r:}^t \right)^{\top} S_{J_r:}^t$
    All-Reduce: $B^t \leftarrow \sum_{i=1}^{N} \bar{B}_i^t$
    Update $U_{I_r:}^{t+1}$ by solving $\min_{U_{I_r:}} \| A_r^t - U_{I_r:} B^t \|$

    Generate random matrix $S'^t \in \mathbb{R}^{m \times d'}$
    Compute $A_r'^t \leftarrow \left( M_{:J_r} \right)^{\top} S'^t$
    Compute $\bar{B}_r'^t \leftarrow \left( U_{I_r:}^t \right)^{\top} S_{I_r:}'^t$
    All-Reduce: $B'^t \leftarrow \sum_{i=1}^{N} \bar{B}_i'^t$
    Update $V_{J_r:}^{t+1}$ by solving $\min_{V_{J_r:}} \| A_r'^t - V_{J_r:} B'^t \|$
return $U_{I_r:}^T$ and $V_{J_r:}^T$
```

Now, the only remaining problem is the transmission of S^t. Since S^t can be dense, even larger than V^t, broadcasting it across the whole cluster can be quite expensive. However, it turns out that we can avoid this. Recall that S^t is a randomly-generated matrix; each node can generate exactly the same matrix, if we use the same pseudo-random generator and the same seed. Therefore, we only need to broadcast the random seed, which is just an integer, at the beginning of the whole program. This ensures that each node generates exactly the same random number sequence and hence the same random matrices S^t at each iteration.

In short, the communication cost of each node is reduced from $\mathcal{O}(nk)$ to $\mathcal{O}(dk)$ by adopting our sketching technique for the U-subproblem. Likewise, the communication cost of each V-subproblem is decreased from $\mathcal{O}(mk)$ to $\mathcal{O}(d'k)$. The general framework of our DSANLS algorithm is listed in Alg. 2.

3.4 Generation of Random Matrices

A key problem in Alg. 2 is how to generate random matrices $S^t \in \mathbb{R}^{n \times d}$ and $S'^t \in \mathbb{R}^{m \times d'}$. Here we focus on generating a random $S^t \in \mathbb{R}^{d \times n}$ satisfying Assumption 1. The reason for choosing such a random matrix is that the corresponding sketched problem would be equivalent to the original problem on expectation; we will prove this in Sec. 3.5.

Assumption 1. *Assume the random matrices are normalized and have bounded variance, i.e., there exists a constant σ^2 such that $\mathbb{E}\left[S^t S^{t\top} \right] = I$ and $\mathbb{V}\left[S^t S^{t\top} \right] \leq \sigma^2$ for all t, where I is the identity matrix.*

Different options exist for such matrices, which have different computation costs in forming sketched matrices $A_r^t = M_{I_r:} S^t$ and $\bar{B}_r^t = \left(V_{J_r:}^t \right)^{\top} S_{J_r:}^t$. Since $M_{I_r:}$ is much larger than $V_{J_r:}^t$ and thus computing A_r^t is more expensive, we only consider the cost of constructing A_r^t here.

The most classical choice for a random matrix is one with i.i.d. Gaussian entries having mean 0 and variance $1/d$. It is easy to show that $\mathbb{E}\left[S^t S^{t\top} \right] = I$. Besides, Gaussian random matrix has bounded variance because Gaussian distribution has finite fourth-order moment. However, since each entry of such a matrix is totally random and thus no special structure exists in S^t, matrix multiplication will be expensive. That is, when given $M_{I_r:}$ of size $|I_r| \times n$, computing its sketched matrix $A_r^t = M_{I_r:} S^t$ requires $\mathcal{O}(|I_r| nd)$ basic operations.

A seemingly better choice for S^t would be a *subsampling* random matrix. Each column of such random matrix is uniformly sampled from $\{e_1, e_2, \ldots, e_n\}$ without replacement,

[1] However, we should not choose an extremely small d, otherwise the the size of sketched subproblem would become so small that it can hardly represent the original subproblem, preventing NMF from converging to a good result. In practice, we can set $d = 0.1n$ for medium-sized matrices and $d = 0.01n$ for large matrices if $m \approx n$. When m and n differ a lot, e.g., $m \ll n$ without loss of generality, we should not apply sketching technique to the V subproblem (since solving the U subproblem is much more expensive) and simply choose $d = m \ll n$.

where $e_i \in \mathbb{R}^n$ is the i-th canonical basis vector (i.e., a vector having its i-th element 1 and all others 0). We can easily show that such an S^t also satisfies $\mathbb{E}\left[S^t S^{t\top}\right] = I$ and the variance $\mathbb{V}\left[S^t S^{t\top}\right]$ is bounded, but this time constructing the sketched matrix $A_r^t = M_{I_r:}S^t$ only requires $\mathcal{O}\left(|I_r|d\right)$. Besides, subsampling random matrix can preserve the sparsity of original matrix. Hence, a subsampling random matrix would be favored over a Gaussian random matrix by most applications, especially for very large-scale or sparse problems. On the other hand, we observed in our experiments that a Gaussian random matrix can result in a faster per-iteration convergence rate, because each column of the sketched matrix A_r^t contains entries from multiple columns of the original matrix and thus is more informative. Hence, it would be better to use a Gaussian matrix when the sketch size d is small and thus a $\mathcal{O}(|I_r|nd)$ complexity is acceptable, or when the network speed of the cluster is poor, hence we should trade more local computation cost for less communication cost.

Although we only test two representative types of random matrices (i.e., Gaussian and subsampling random matrices), our framework is readily applicable for other choices, such as subsampled randomized Hadamard transform (SRHT) [42, 43] and count sketch [44, 45]. The choice of random matrices is not the focus of this paper and left for future investigation.

3.5 Solving Subproblems

Before describing how to solve subproblem (10), let us make an important observation. As discussed in Sec. 2.2.2, the sketching technique has been applied in solving linear systems before. However, the situation is different in matrix factorization. Note that for the distributed matrix factorization problem we usually have:

$$\min_{U_{I_r:} \in \mathbb{R}_+^{|I_r| \times k}} \left\| M_{I_r:} - U_{I_r:}V^{t\top} \right\|_F^2 \neq 0. \tag{12}$$

So, for the sketched subproblem (10), which can be equivalently written as:

$$\min_{U_{I_r:} \in \mathbb{R}_+^{|I_r| \times k}} \left\| \left(M_{I_r:} - U_{I_r:}V^{t\top} \right) S^t \right\|_F^2, \tag{13}$$

where the non-zero entries of the residual matrix $\left(M_{I_r:} - U_{I_r:}V^{t\top}\right)$ will be scaled by the matrix S^t at different levels. As a consequence, the optimal solution will be shifted because of sketching. This fact alerts us that for SANLS, we need to update U^{t+1} by exploiting the sketched subproblem (10) to step towards the true optimal solution and avoid convergence to the solution of the sketched subproblem.

3.5.1 Projected Gradient Descent

A natural method is to use *one step*[2] of projected gradient descent for the sketched subproblem:

$$\begin{aligned} U_{I_r:}^{t+1} &= \max\left\{ U_{I_r:}^t - \eta_t \, \nabla_{U_{I_r:}} \left\| A_r^t - U_{I_r:}B^t \right\|_F^2 \Big|_{U_{I_r:} = U_{I_r:}^t}, \, 0 \right\} \\ &= \max\left\{ U_{I_r:}^t - 2\eta_t \left[U_{I_r:}^t B^t B^{t\top} - A_r^t B^{t\top} \right], \, 0 \right\}, \end{aligned} \tag{14}$$

where $\eta_t > 0$ is the step size and $\max\{\cdot, \cdot\}$ denotes the entry-wise maximum operation. In the gradient descent step (14), the computational cost mainly comes from two matrix multiplications: $B^t B^{t\top}$ and $A_{t,r} B^{t\top}$. Note that A_r^t and B^t are of sizes $|I_r| \times d$ and $k \times d$ respectively, thus the gradient descent step takes $\mathcal{O}\left(kd(|I_r| + k)\right)$ in total.

To exploit the nature of this algorithm, we further expand the gradient:

$$\begin{aligned} &\nabla_{U_{I_r:}} \left\| A_r^t - U_{I_r:}B^t \right\|_F^2 = 2\left[U_{I_r:}B^t B^{t\top} - A_r^t B^{t\top} \right] \\ &\overset{(9)}{=} 2\left[U_{I_r:}\left(V^{t\top}S^t\right)\left(V^{t\top}S^t\right)^\top - \left(M_{I_r:}S^t\right)\left(V^{t\top}S^t\right)^\top \right] \\ &= 2\left[U_{I_r:}V^{t\top}\left(S^t S^{t\top}\right)V^t - M_{I_r:}\left(S^t S^{t\top}\right)V^t \right]. \end{aligned} \tag{15}$$

By taking the expectation of the above equation, and using the fact $\mathbb{E}\left[S^t S^{t\top}\right] = I$, we have:

$$\begin{aligned} &\mathbb{E}\left[\nabla_{U_{I_r:}} \left\| A_r^t - U_{I_r:}B^t \right\|_F^2 \right] = 2\left[U_{I_r:}V^{t\top}V^t - M_{I_r:}V^t \right] \\ &= \nabla_{U_{I_r:}} \left\| M_{I_r:} - U_{I_r:}V^{t\top} \right\|_F^2 \end{aligned} \tag{16}$$

which means that the gradient of the sketched subproblem is equivalent to the gradient of the original problem on expectation. Therefore, such a step of gradient descent can be interpreted as a (generalized) *stochastic gradient descent* (SGD) [46] method on the original subproblem. Thus, according to the theory of SGD, we naturally require the step sizes $\{\eta_t\}$ to be diminishing, i.e., $\eta_t \to 0$ as t increases.

3.5.2 Proximal Coordinate Descent

However, it is well known that the gradient descent method converges slowly, while the coordinate descent method, namely the HALS method for NMF, is quite efficient [10]. Still, because of its very fast convergence, HALS should not be applied to the sketched subproblem directly because it shifts the solution away from the true optimal solution. Therefore, we would like to develop a method which resembles HALS but will not converge towards the solutions of the sketched subproblems.

To achieve this, we add a regularization term to the sketched subproblem (10). The new subproblem becomes:

$$\min_{U_{I_r:} \in \mathbb{R}_+^{|I_r| \times k}} \left\| A_r^t - U_{I_r:}B^t \right\|_F^2 + \mu_t \left\| U_{I_r:} - U_{I_r:}^t \right\|_F^2, \tag{17}$$

where $\mu_t > 0$ is a parameter. Such regularization is reminiscent to the proximal point method [47] and parameter μ_t controls the step size as $1/\eta_t$ in projected gradient descent. We therefore require $\mu_t \to +\infty$ to enforce the convergence of the algorithm, e.g., $\mu_t = t$.

At each step of proximal coordinate descent, only one column of $U_{I_r:}$, say $U_{I_r,j}$ where $j \in \{1, 2, \ldots, k\}$, is updated:

$$\min_{U_{I_r:j} \in \mathbb{R}_+^{|I_r|}} \left\| A_r^t - U_{I_r:j}B_{j:}^t - \sum_{l \neq j} U_{I_r:l}B_{l:}^t \right\|_F^2 + \mu_t \left\| U_{I_r:j} - U_{I_r:j}^t \right\|_2^2. \tag{18}$$

It is not hard to see that the above problem is still row-independent, which means that each entry of the row vector

[2] Note that we only apply one step of projected gradient descent here to avoid solution shifted.

Algorithm 3 Proximal Coordinate Descent for Local Subproblem (10) on Node r

```
Parameter: $\mu_t > 0$
for $j = 1$ to $k$ do
    $T \leftarrow \mu_t U^t_{I_r:j} + A^t_r B^{t\top}_{j:}$
    for $l = 1$ to $j - 1$ do
        $T \leftarrow T - \left(B^t_{l:} B^{t\top}_{j:}\right) U^{t+1}_{I_r:l}$
    for $l = j + 1$ to $k$ do
        $T \leftarrow T - \left(B^t_{l:} B^{t\top}_{j:}\right) U^{t}_{I_r:l}$
    $U^{t+1}_{I_r:j} \leftarrow \max\left\{T/\left(B^t_{j:} B^{t\top}_{j:} + \mu_t\right), 0\right\}$
return $U^{t+1}_{I_r:}$
```

$U_{I_r:j}$ can be solved independently at each node. For example, for any $i \in I_r$, the solution of $U^{t+1}_{i:j}$ is given by:

$$\begin{aligned} U^{t+1}_{i:j} &= \underset{U_{i:j} \geq 0}{\arg\min} \left\| \left(A^t_r\right)_{i:} - U_{i:j} B^t_{j:} - \sum_{l \neq j} U_{i:l} B^t_{l:} \right\|_2^2 \\ &\quad + \mu_t \left\| U_{i:j} - U^t_{i:j} \right\|_2^2 \\ &= \max \left\{ \frac{\mu_t U^t_{i:j} + \left(A^t_r\right)_{i:} B^{t\top}_{j:} - \sum_{l \neq j} U_{i:l} B^t_{l:} B^{t\top}_{j:}}{B^t_{j:} B^{t\top}_{j:} + \mu_t}, 0 \right\}. \end{aligned} \tag{19}$$

At each step of coordinate descent, we choose the column j from $\{1, 2, \ldots, k\}$ successively. When updating column j at iteration t, the columns $l < j$ have already been updated and thus $U_{I_r:l} = U^{t+1}_{I_r:l}$, while the columns $l > j$ are old so $U_{I_r:l} = U^t_{I_r:l}$.

The complete proximal coordinate descent algorithm for the U-subproblem is summarized in Alg. 3. When updating column j, computing the matrix-vector multiplication $A^t_r B^{t\top}_{j:}$ takes $\mathcal{O}(d|I_r|)$. The whole inner loop takes $\mathcal{O}\left(k\left(d + |I_r|\right)\right)$ because one vector dot product of length d is required for computing each summand and the summation itself needs $\mathcal{O}\left(k|I_r|\right)$. Considering that there are k columns in total, the overall complexity of coordinate descent is $\mathcal{O}\left(k((k+d)\,|I_r| + kd)\right)$. Typically, we choose $d > k$, so the complexity can be simplified to $\mathcal{O}\left(kd\left(|I_r| + k\right)\right)$, which is the same as that of gradient descent.

Since proximal coordinate descent is much more efficient than projected gradient descent, we adopt it as the default subproblem solver within DSANLS.

3.6 Theoretical Analysis

3.6.1 Complexity Analysis

We now analyze the computational and communication costs of our DSANLS algorithm, when using subsampling random sketch matrices. The computational complexity at each node is:

$$\begin{aligned} &\mathcal{O}\Big(\overbrace{d}^{\text{generating } S^t} + \overbrace{|I_r|d}^{\text{constructing } A^t_r \text{ and } B^t} + \overbrace{kd(|I_r| + k)}^{\text{solving subproblem}} \Big) \\ &= \mathcal{O}\left(kd(|I_r| + k)\right) \approx \mathcal{O}\left(kd\left(\frac{m}{N} + k\right)\right). \end{aligned} \tag{20}$$

Moreover, as we have shown in Sec. 3.3, the communication cost of DSANLS is $\mathcal{O}\left(kd\right)$.

On the other hand, for a classical implementation of distributed HALS [48], the computational cost is:

$$\mathcal{O}\left(kn\left(|I_r| + k\right)\right) \approx \mathcal{O}\left(kn\left(\frac{m}{N} + k\right)\right) \tag{21}$$

and the communication cost is $\mathcal{O}\left(kn\right)$ due to the all-gathering of V^t's.

Comparing the above quantities, we observe an $n/d \gg 1$ speedup of our DSANLS algorithm over HALS in both computation and communication. However, we empirically observed that DSANLS has a slower per-iteration convergence rate (i.e., it needs more iterations to converge). Still, as we will show in the next section, in practice, DSANLS is superior to alternative distributed NMF algorithms, after taking all factors into account.

3.6.2 Convergence Analysis

Here we provide theoretical convergence guarantees for the proposed SANLS and DSANLS algorithms. We show that SANLS and DSANLS converge to a stationary point.

To establish convergence result, Assumption 2 is needed first.

Assumption 2. *Assume all the iterates U^t and V^t have uniformly bounded norms, which means that there exists a constant R such that $\|U^t\|_F \leq R$ and $\|V^t\|_F \leq R$ for all t.*

We experimentally observed that this assumption holds in practice, as long as the step sizes used are not too large. Besides, Assumption 2 can also be enforced by imposing additional constraints, such as:

$$U_{i:l} \leq \sqrt{2\|M\|_F} \quad \text{and} \quad V_{j:l} \leq \sqrt{2\|M\|_F} \quad \forall i, j, l, \tag{22}$$

with which we have $R = \max\{m, n\} k \sqrt{2\|M\|_F}$. Such constraints can be very easily handled by both of our projected gradient descent and regularized coordinate descent solvers. Lemma 1 shows that imposing such extra constraints does not prevent us from finding the global optimal solution.

Lemma 1. *If the optimal solution to the original problem* (1) *exists, there is at least one global optimal solution in the domain* (22).

Based on Assumptions 1 (see Sec. 3.4) and Assumption 2, we now can formally show our main convergence result:

Theorem 1. *Under Assumptions 1 and 2, if the step sizes satisfy $\sum_{t=1}^{\infty} \eta_t = \infty$ and $\sum_{t=1}^{\infty} \eta_t^2 < \infty$, for projected gradient descent, or $\sum_{t=1}^{\infty} 1/\mu_t = \infty$ and $\sum_{t=1}^{\infty} 1/\mu_t^2 < \infty$, for regularized coordinate descent, then SANLS and DSANLS with either subproblem solver will converge to a stationary point of problem* (1) *with probability 1.*

The proofs of Lemma 1 and Theorem 1 can be found in Appendices A and B.

4 Secure Distributed NMF

In this section, we provide our solutions to the problem of secure distributed NMF over federated data.

4.1 Extend DSANLS to Secure Setting

DSANLS and all lines of works discussed in Sec. 2.2.1 store copies of M across two-dimensional (shown in Fig. 1(a)), and exploit the independence of local update computation for rows of U and V to apply communication-optimal matrix multiplication. They cannot be applied directly to secure distributed NMF setting. The reason is that, in secure distributed NMF setting (shown in Fig. 1(b)), only one column copy is stored in each node, while the others cannot be disclosed.

Nevertheless, DSANLS can be adapted to this secure setting with modification, but only for *one or limited iterations*. The reason is illustrated in Theorem 2. In modified DSANLS algorithm, each node still takes charge of updating $U_{I_r:}$ and $V_{J_r:}$ as before, but only one copy $M_{:J_r}$ of $M = [M_1, M_2, ..., M_N]$ will be stored in node r. Thus, V-subproblem is exactly the same as in DSANLS. Differently, we need to use MPI-AllReduce function to gather $M_{:J_r}S^t$ from all nodes before each iteration of U-subproblem, so that each node has access to fully sketched matrix MS^t to solve sketched U-subproblem. Note that here random matrix S^t not only helps reduce the communication cost from $\mathcal{O}(mn)$ to $\mathcal{O}(md)$ with a smaller NLS problem, but also conceals the full matrix M in each iteration.

Theorem 2. *M cannot be recovered only using information about MS (or SM) and S.*

Proof. Assume S is a square matrix. Given MS (or SM) and S, we are able to get M by $M = MSS^{-1}$ (or $M = S^{-1}$SM). However, the numbers of row and column are highly imbalanced in S and it is not a square matrix. Therefore M cannot be recovered only using information about MS (or SM) and S. □

However, NMF is an iterative algorithm (shown in Alg. 1). Secure computation in limited iterations cannot guarantee an acceptable accuracy for practical use due to the following reason:

Theorem 3. *M can be recovered after enough iterations.*

Proof. If we view $M \cdot S = MS$ as a system of linear equations with a variable matrix M and constant matrices S and MS. Each row of M can be solved by a standard Gaussian Elimination solver, given a sufficient number of (S, MS) pairs. □

Theorem 3 suggests that DSANLS algorithm suffers from the dilemma of choosing between information disclosure and unacceptable accuracy, making it impractical to real applications. Therefore, we need to propose new practical solutions to secure distributed NMF.

4.2 Synchronous Framework

A straightforward solution to secure distributed NMF is that each node solves a local NMF problem with a local copy of U (denoted as $U_{(r)}$ for node r). Periodically, nodes communicate with each other, and update local copy of U to the aggregation of all local copies $U_{(j)}, j \in \{1, \cdots, N\}$ by All-Reduce operation. We name this method as *Syn-SD* under synchronous setting. The detailed algorithm is shown in Alg. 4. Within inner iterations, every node maintains its own copy of U (i.e., $U_{(r)}$) by solving the regular NMF problem. Every T_2 rounds, different local copies of U will be averaged through nodes by using $\sum_{j=1}^{N} U_{(j)}/N$. Note that, $U_{(r)}$ is one copy of the whole matrix U stored locally in node r, while $V_{J_r:}$ is the corresponding part of the matrix $V = [V_{J_1:}; V_{J_2:}; ..., V_{J_N:}]$ stored in node r.

Algorithm 4 Syn-SD: Secure Distributed NMF on node r

```
Input: $M_{:J_r}$
Parameter: Iteration numbers $T_1, T_2$
initialize $U^0_{(r)} \geq 0$, $V^0_{J_r:} \geq 0$
for $t_1 = 0$ to $T_1 - 1$ do
    for $t_2 = 1$ to $T_2$ do
        $t \leftarrow t_1 \times T_2 + t_2$
        $U^t_{(r)} \leftarrow$ update($M_{:J_r}$, $U^{t-1}_{(r)}$, $V^{t-1}_{J_r:}$)
        $V^t_{J_r:} \leftarrow$ update($M_{:J_r}$, $U^t_{(i)}$, $V^{t-1}_{J_r:}$)
    All-Reduce: $U^t_{(r)} \leftarrow \frac{\sum_{j=1}^{N} U^t_{(j)}}{N}$
return $U^t_{(r)}$ and $V^t_{J_r:}$
```

Algorithm 5 Syn-SSD: Secure Sketched Distributed NMF on node r

```
Input: $M_{:J_r}$
Parameter: Iteration numbers $T_1, T_2$
initialize $U^0_{(i)} \geq 0$, $V^0_{J_r:} \geq 0$
for $t_1 = 0$ to $T_1 - 1$ do
    for $t_2 = 1$ to $T_2$ do
        $t \leftarrow t_1 \times T_2 + t_2$
        Generate random matrix $S^t_1$
        $U^t_{(r)} \leftarrow$ update($M_{:J_r}S^t_1$, $U^{t-1}_{(r)}$, $V^{t-1}_{J_r:}S^t_1$)
        Generate random matrix $S^t_2$
        All-Reduce: $\overline{SU}^t \leftarrow \frac{\sum_{j=1}^{N} S^t_2 U^t_{(j)}}{N}$
        $V^t_{J_r:} \leftarrow$ update($S^t_2 M_{:J_r}$, $\overline{SU}^t$, $V^{t-1}_{J_r:}$)
    All-Reduce: $U^t_{(r)} \leftarrow \frac{\sum_{j=1}^{N} U^t_{(j)}}{N}$
return $U^t_{(r)}$ and $V^t_{J_r:}$
```

In Syn-SD, the local copy $U_{(r)}$ in node r will be updated to a uniform aggregation of local copies from all nodes periodically. Small number of inner iteration T_2 incurs large communication cost caused by All-Reduce. Larger T_2 may lead to slow convergence, since each node does not share any information of its local copy $U_{(r)}$ inside the inner iterations.

To improve the efficiency of data exchange, we incorporate matrix sketching to *Syn-SD*, and propose an improved version called *Syn-SSD*. In Syn-SSD, information of local copies is shared across cluster nodes more frequently, with communication overhead roughly the same as Syn-SD. As shown in Alg. 5, the sketched version $S^tU_{(r)}$ of the local copy $U_{(r)}$ is exchanged within each inner iteration. There are two advantages of applying matrix sketching: (1) Since the sketched matrix has a much smaller size, All-Reduce operation causes much less communication cost, making it affordable with higher frequency. (2) Solving a sketched NLS problem can also reduce the computation cost due to a reduced problem size of solving $U_{(r)}$ and $V_{J_r:}$ for each node. It is worth noting that S^t_1 is exactly the same for each node by using the same seed and generator. The same for S^t_2. But S^t_1 and S^t_2 are not necessarily equivalent. With such a constraint, the algorithm is equivalent to NMF in single-machine environment and the convergence can be guaranteed.

It is straightforward to see that Syn-SD and Syn-SSD satisfy Definition 1 and they are $(N-1)$-private protocols, since $V_{J_r:}$ and $M_{:J_r}$ are only seen by node r.

Algorithm 6 *Asyn-SD*, *Asyn-SSD*: Server part

```
Parameter: Relaxation parameter $\rho$
1: initialize $U^0 \geq 0$
2: $t \leftarrow 0$                                   ▷ $t$ is the update counter.
3: while not stopping do
4:     Receive $U^t_{(r)}$ from client node $r$
5:     $\omega^t \leftarrow \frac{\rho}{\rho+t}$                ▷ $\omega^t$ is the relaxation weight.
6:     $U^t \leftarrow (1-\omega^t)U^t + \omega^t U^t_{(r)}$
7:     Send $U^t$ back to client node $r$
8:     $t \leftarrow t+1$
9: return $U^t$
```

Algorithm 7 *Asyn-SD*, *Asyn-SSD*: Client part of node r

```
Input: $M_{:J_r}$
Parameter: Iteration number $T$
1: initialize $V^0_{J_r:} \geq 0$
2: while Server not stopping do
3:     Receive $U$ from server
4:     $U^0_{(r)} \leftarrow U$
5:     for $t = 1$ to $T$ do
6:         $V^t_{J_r:} \leftarrow$ update($M_{:J_r}$, $U^{t-1}_{(r)}$, $V^{t-1}_{J_r:}$)
7:         $U^t_{(r)} \leftarrow$ update($M_{:J_r}$, $U^{t-1}_{(r)}$, $V^t_{J_r:}$)    ▷ For Asyn-SSD,
           replace it with Lines 5-6 of Alg. 5.
8:     Send $U^T_{(r)}$ to server
9: return $V^T_{J_r:}$
```

4.3 Asynchronous Framework

In Syn-SD and Syn-SSD, each node must stall until all participating nodes reach the synchronization barrier before the All-Reduce operation. However, highly imbalanced data in real scenario of federated data mining may cause severe workload imbalance problem. The synchronization barrier will force nodes with low workload to halt, making synchronous algorithms less efficient. In this section, we study secure distributed NMF in an asynchronous (i.e., server/client architecture) setting and propose corresponding asynchronous algorithms.

First of all, we extend the idea of Syn-SD to asynchronous setting and name the new method *Asyn-SD*. In Asyn-SD, the server (in Alg. 6) takes full charge of updating and broadcasting U^t. Once received $U^t_{(r)}$ from the client node r, the server would update U^t locally, and return the latest version of U^t back to the client node r for further computing. Note that the server may receive local copies of U^t from clients in an arbitrary order. Consequently, we cannot use the same operation of All-Reduce as Syn-SD any more. Instead, U^t in server side is updated by the weighted sum of current U^t and newly received local copy $U^t_{(r)}$ from client node r. Here the relaxation weight ω^t asymptotically converges to 0. Thus a converged U^t is guaranteed on server side. Our experiments in Sec. 5 suggest that this relaxation has no harm to factorization convergence.

On the other hand, client nodes of Asyn-SD (in Alg. 7) behave similarly as nodes in Syn-SD. Clients locally solve the standard NMF problem for T iterations, and then update local $U^t_{(r)}$ by communicating only with the server node. Unlike Syn-SD, Asyn-SD does not have a global synchronization barrier. Client nodes in Asyn-SD independently exchange their local copy $U^t_{(r)}$ with the server without an All-Reduce operation.

TABLE 1
Statistics of datasets

Dataset	#Rows	#Columns	Non-zero values	Sparsity
BOATS	216,000	300	64,800,000	0%
MIT CBCL FACE	2,429	361	876,869	0%
MNIST	70,000	784	10,505,375	80.86%
GISETTE	13,500	5,000	8,770,559	87.01%
Reuters (RCV1)	804,414	47,236	60,915,113	99.84%
DBLP	317,080	317,080	2,416,812	99.9976%

Similarly, Syn-SSD can be extended to its asynchronous version *Asyn-SSD*. However, the algorithm for clients is more constrained and conservative in sketching. Note that the random sketching matrices S_1 and S_2 (in Alg. 5) should be the same across the nodes in the same summation in order to have a meaningful summation of sketched matrices. However, enforcing the same S^t_2 for updating sketched U will result in a synchronous All-Reduce operation. Therefore, U cannot be sketched in asynchronous algorithms and we only consider sketching $V_{J_r:}$ in Asyn-SSD (Line 7 in Alg. 7). The server part of Asyn-SSD is the same as Asyn-SD in Alg. 6.

Similar to synchronous versions, Asyn-SD and Asyn-SSD satisfy Definition 1 and they are $(N-1)$-private protocols, since $V_{J_r:}$ and $M_{:J_r}$ are only seen by node r.

5 Experimental Evaluation

This section includes an experimental evaluation of our algorithms on both dense and sparse real data matrices. The implementation of our methods is available at https://github.com/qianyuqiu79/DSANLS.

5.1 Setup

We use several (dense and sparse) real datasets as Qian et al. [49] for evaluation. They corresponds to different NMF tasks, including video analysis, image processing, text mining and community detection. Their statistics are summarized in Tab. 1.

We conduct our experiments on a Linux cluster with 16 nodes. Each node contains 8-core Intel® Core$^{\text{TM}}$ i7-3770 CPU @ 1.60GHz cores and 16 GB of memory. Our algorithms are implemented in C++ using the Intel® Math Kernel Library (MKL) and Message Passing Interface (MPI). By default, we use 10 nodes and set the factorization rank k to 100. We also report the impact of different node number (2-16) and k (20-500). We use $\mu_t = \alpha + \beta t$ [50], do the grid search for α and β in the range of $\{0.1, 1, 10\}$ for each dataset and report the best results. Because the use of Gaussian random matrices is too slow on large datasets RCV1 and DBLP, we only use subsampling random matrices for them.

For the general acceleration of NMF, we assess DSANLS with subsampling and Gaussian random matrices, denoted by DSANLS/S and DSANLS/G, respectively, using proximal coordinate descent as the default subproblem solver. As mentioned in [5, 8], it is unfair to compare with a Hadoop implementation. We only compare DSANLS with MPI-FAUN[3] (MPI-FAUN-MU, MPI-FAUN-HALS, and MPI-FAUN-ABPP implementations), which is the first and the

[3] https://github.com/ramkikannan/nmflibrary

state-of-the-art C++/MPI implementation with MKL and Armadillo. For parameters pc and pr in MPI-FAUN, we use the optimal values for each dataset, according to the recommendations in [5, 8].

For the problem of secure distributed NMF, we evaluate all proposed methods: Syn-SD, Syn-SSD with sketch on U (denoted as Syn-SSD-U), Syn-SSD with sketching on V (denoted as Syn-SSD-V), Syn-SSD with sketching on both U and V (denoted as Syn-SSD-UV), Asyn-SD, Asyn-SSD with sketching on V (denoted as Asyn-SSD-V), using proximal coordinate descent as the default subproblem solver. We do not list secure building block methods as baselines, since communication overhead is heavy in these multi-round handshake protocols and it is unfair to compare them with MPI based methods. For example, a matrix sum described by Duan and Canny [39] results in 5X communication overhead compared to a MPI all-reduce operation.

We use the relative error of the low rank approximation compared to the original matrix to measure the effectiveness of different NMF approaches. This error measure has been widely used in previous work [5, 8, 51] and is formally defined as $\left\|M - UV^\top\right\|_F / \|M\|_F$.

5.2 Evaluation on Accelerating General NMF

5.2.1 Performance Comparison

Since the time for each iteration is significantly reduced by our proposed DSANLS compared to MPI-FAUN, in Fig. 2, we show the relative error over time for DSANLS and MPI-FAUN implementations of MU, HALS, and ANLS/BPP on the 6 real public datasets. Observe that DSANLS/S performs best in all 6 datasets, although DSANLS/G has faster per-iteration convergence rate. MU converges relatively slowly and usually has a bad convergence result; on the other hand HALS may oscillate in the early rounds[4], but converges quite fast and to a good solution. Surprisingly, although ANLS/BPP is considered to be the state-of-art NMF algorithm, it does not perform well in all 6 datasets. As we will see, this is due to its high per-iteration cost.

5.2.2 Scalability Comparison

We vary the number of nodes used in the cluster from 2 to 16 and record the average time for 100 iterations of each algorithm. Fig. 3 shows the reciprocal of per-iteration time as a function of the number of nodes used. All algorithms exhibit good scalability for all datasets (nearly a straight line), except for FACE (i.e., Fig. 3(a)). FACE is the smallest dataset, whose number of columns is 300, while k is set to 100 by default. When n/N is smaller than k, the complexity is dominated by k, hence, increasing the number of nodes does not reduce the computational cost, but may increase the communication overhead. In general, we can observe that DSANLS/Subsampling has the lowest per-iteration cost compared to all other algorithms, and DSANLS/Gaussian has similar cost to MU and HALS. ANLS/BPP has the highest per-iteration cost, explaining the bad performance of ANLS/BPP in Fig. 2.

5.2.3 Performance Varying the Value of k

Although tuning the factorization rank k is outside the scope of this paper, we compare the performance of DSANLS with MPI-FAUN varying the value of k from 20 to 500 on RCV1. Observe from Fig. 4 and Fig. 2(e) ($k = 100$) that DSANLS outperforms the state-of-art algorithms for all values of k. Naturally, the relative error of all algorithms decreases with the increase of k, but they also take longer to converge.

5.2.4 Comparison with Projected Gradient Descent

In Sec. 3.5, we claimed that our proximal coordinate descent approach (denoted as DSANLS-RCD) is faster than projected gradient descent (also presented in the same section, denoted as DSANLS-PGD). Fig. 5 confirms the difference in the convergence rate of the two approaches regardless of the random matrix generation approach.

5.3 Evaluation on Secure Distributed NMF

5.3.1 Performance Comparison for Uniform Workload

In Fig. 6, we show the relative error over time for secure distributed NMF algorithms on the 4 real public datasets, with a uniformly partition of columns. *Syn-SSD-UV* performs best in BOAT, FACE and GISETTE. As we will see in the next section, this is due to the fact that per-iteration cost of *Syn-SSD-UV* is significantly reduced by sketching. On MNIST, *Syn-SSD-U* and *Syn-SSD-V* has a better convergence in terms of relative error. *Syn-SD* and *Asyn-SD* converge relatively slowly and usually have a bad convergence result; on the other hand *Asyn-SSD-V* converges slowly but consistently generates better results than *Syn-SD* and *Asyn-SD*.

5.3.2 Performance Comparison for Imbalanced Workload

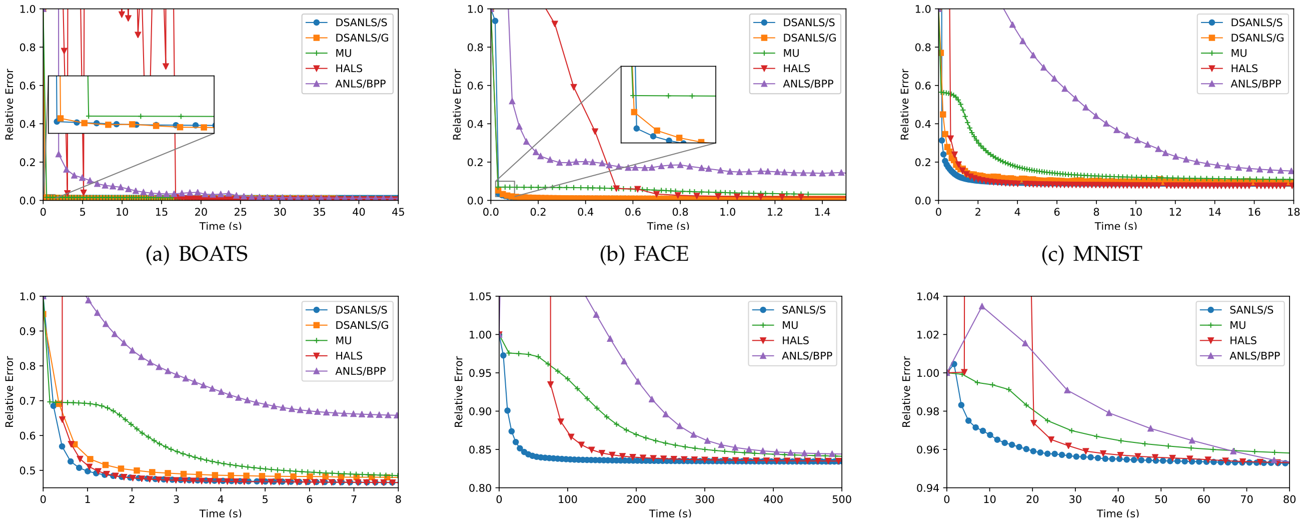
To evaluate the performance of different methods when the workload is imbalanced, we conduct experiments on skewed partition of input matrix. Among 10 worker nodes, node 0 is assigned with 50% of the columns, while other nodes have a uniform partition of the rest of columns. The measure for error is the same as the case of uniform workload.

It can be observed that in imbalanced workload, asynchronous algorithms generally outperform synchronous algorithms. *Asyn-SSD-V* gives the best result in terms of relative error over time, except dataset FACE. In FACE, *Asyn-SD* slowly converges to the best result. Unlike the case of uniform workload in Fig. 6, the sketching method *Syn-SSD-UV* does not perform well in imbalanced workload. *Syn-SD* are basically inapplicable in BOATS, MNIST and GISETTE datasets due to its slow speed. In sparse datasets MNIST and GISETTE, *Syn-SSD-V* and *Syn-SSD-U* can converge to a good result, but they do not generate satisfactory results on dense dataset BOATS and FACE.

5.3.3 Scalability Comparison

We vary the number of nodes used in the cluster from 2 to 16 and record the average time for 100 iterations of each algorithm. Fig. 8 shows the reciprocal of per-iteration time as a function of the number of nodes for uniform workload. All

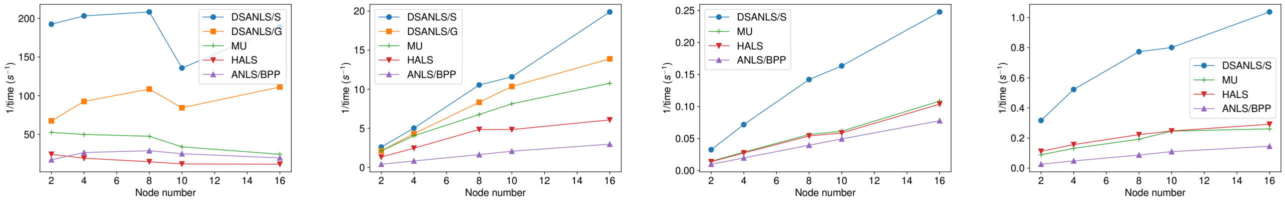
[4] HALS does not guarantee the objective function to decrease monotonically.



(a) BOATS (b) FACE (c) MNIST

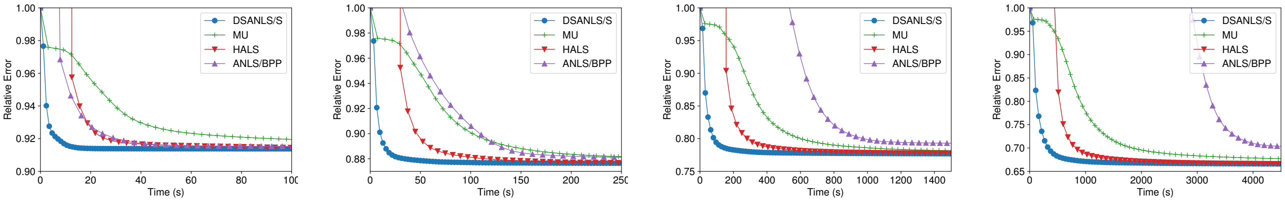
(d) GISETTE (e) RCV1 (f) DBLP

Fig. 2. Relative error over time for general distributed NMF



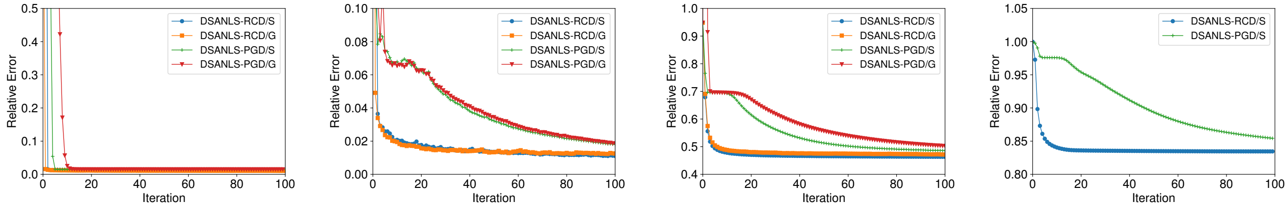
(a) FACE (b) MNIST (c) RCV1 (d) DBLP

Fig. 3. Reciprocal of per-iteration time as a function of cluster size for general distributed NMF



(a) k=20 (b) k=50 (c) k=200 (d) k=500

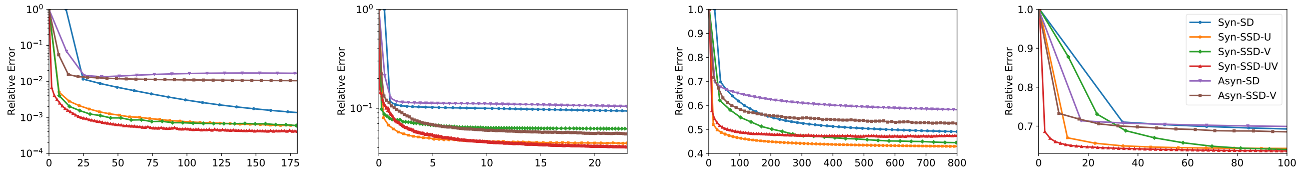
Fig. 4. Relative error over time for general distributed NMF, varying k value



(a) BOATS (b) FACE (c) GISETTE (d) RCV1

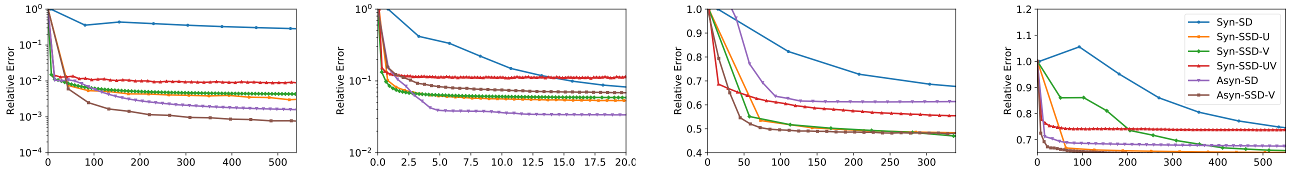
Fig. 5. Relative error per-iteration of different subproblem solvers for general distributed NMF

algorithms exhibit good scalability for all datasets (nearly a straight line), except for FACE (i.e., Fig. 8(b)). FACE is the smallest dataset, whose number of columns is 361 and number of row is 2,429. When n/N is smaller than $k = 100$, the time consumed by subproblem solvers is dominated by the communication overhead. Hence, increasing the number of nodes is does not reduce per-iteration time. In general, we can observe that *Syn-SSD-UV* has the lowest per-iteration time compared to all other algorithms, and also has the best scalability as we can see from the steepest



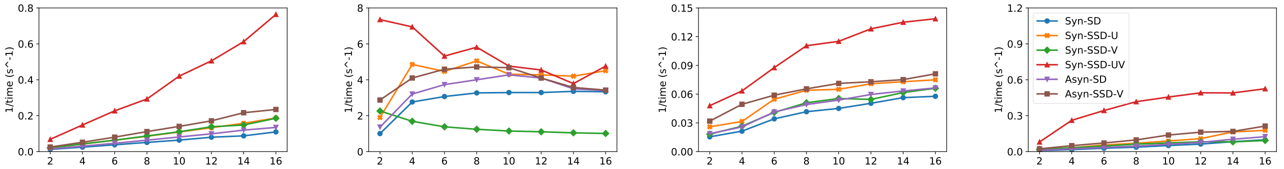
(a) BOATS (b) FACE (c) MNIST (d) GISETTE

Fig. 6. Relative error over time for uniform workload in secure distributed NMF



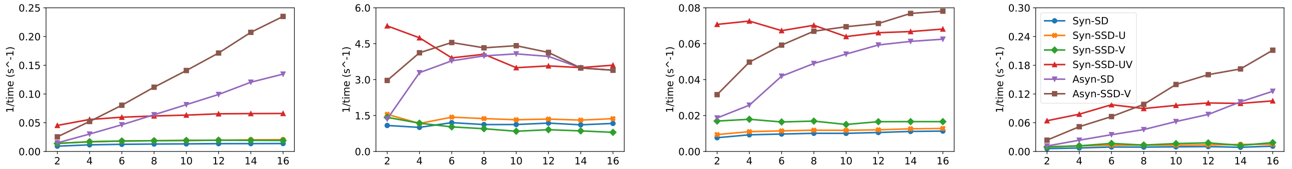
(a) BOATS (b) FACE (c) MNIST (d) GISETTE

Fig. 7. Relative error over time for imbalanced workload in secure distributed NMF



(a) BOATS (b) FACE (c) MNIST (d) GISETTE

Fig. 8. Reciprocal of per-iteration time for uniform workload in secure distributed NMF



(a) BOATS (b) FACE (c) MNIST (d) GISETTE

Fig. 9. Reciprocal of per-iteration time for imbalanced workload in secure distributed NMF

slope. Synchronous averaging has the highest per-iteration cost, explaining the bad performance in uniform workload experiments in Fig. 6.

In imbalanced workload settings, it is not surprising that asynchronous algorithms outperform synchronous algorithms with respect to scalability, as shown in Fig. 9. Synchronization barriers before All-Reduce operations severely affect the scalability of synchronous algorithms, resulting in a nearly flat curve for per-iteration time. The per-iteration time of *Syn-SSD-UV* is satisfactory when cluster size is small. However, it does not get significant improvements when more nodes are deployed. On the other hand, asynchronous algorithms demonstrate decent scalability as number of nodes grows. The short average iteration time of *Asyn-SD* and *Asyn-SSD-V*, shown in Fig. 9, also explains their superior performance over their synchronous counterparts in Fig. 7.

In conclusion, with an overall evaluation of convergence and scalability, *Syn-SSD-UV* should be adopted for secure distributed NMF under uniform workload, while *Asyn-SSD-V* is a more reasonable choice for secure distributed NMF under imbalanced workload.

6 Conclusion

In this paper, we studied the acceleration and security problems for distributed NMF. Firstly, we presented a novel distributed NMF algorithm DSANLS that can be used for scalable analytics of high dimensional matrix data. Our approach follows the general framework of ANLS, but utilizes matrix sketching to reduce the problem size of each NLS subproblem. We discussed and compared two different approaches for generating random matrices (i.e., Gaussian and subsampling random matrices). We presented two subproblem solvers for our general framework, and theoretically proved that our algorithm is convergent. We analyzed the per-iteration computational and communication cost of our approach and its convergence, showing its superiority compared to the state-of-the-art. Secondly, we designed four efficient distributed NMF methods in both synchronous and asynchronous settings with a security guarantee. They are the first distributed NMF methods over federated data, where data from all parties are utilized together in NMF for better performances and the data of each party remains confidential without leaking any individual information to other parties during the process. Finally, we conducted extensive experiments on several real datasets to show the superiority of our proposed methods. In the future, we plan to study the applications of DSANLS in dense or sparse tensors and consider more practical designs of asynchronous algorithm for secure distributed NMF.

7 Acknowledgment

This work was supported by the National Natural Science Foundation of China (no. 61972328).

Appendix A
Proof of Lemma 1

Proof of Lemma 1. Suppose (U^*, V^*) is the global optimal solution but fails to satisfy Eq. 22 in the paper. If there exist indices i, j, l such that $U^*_{i:l} \cdot V^*_{j:l} > 2\|M\|_F$, then

$$\begin{aligned}\left\|M - U^*V^{*\top}\right\|_F^2 &\geq \left(U^*_{i:l} \cdot V^*_{j:l} - M_{i:j}\right)^2 > \left(2\|M\|_F - \|M\|_F\right)^2 \\ &\geq \|M\|_F^2.\end{aligned} \tag{23}$$

However, simply choosing $U = 0$ and $V = 0$ will yield a smaller error $\|M\|_F^2$, which contradicts the fact that (U^*, V^*) is optimal. Therefore, if we define $\alpha_l = \max_i U^*_{i:l}$ and $\beta_l = \max_j V^*_{j:l}$, we must have $\alpha_l \cdot \beta_l \leq 2\|M\|_F$ for each l. Now we construct a new solution $(\overline{U}, \overline{V})$ by:

$$\overline{U}_{i:l} = U^*_{i:l} \cdot \sqrt{\beta_l/\alpha_l} \quad \text{and} \quad \overline{V}_{j:l} = V^*_{j:l} \cdot \sqrt{\alpha_l/\beta_l}. \tag{24}$$

Note that

$$\begin{aligned}\overline{U}_{i:l} &\leq \alpha_l \cdot \sqrt{\beta_l/\alpha_l} = \sqrt{\alpha_l \cdot \beta_l} \leq \sqrt{2\|M\|_F}, \\ \overline{V}_{j:l} &\leq \beta_l \cdot \sqrt{\alpha_l/\beta_l} = \sqrt{\alpha_l \cdot \beta_l} \leq \sqrt{2\|M\|_F},\end{aligned} \tag{25}$$

so $(\overline{U}, \overline{V})$ satisfy Eq. 22 in the paper. Besides,

$$\begin{aligned}&\left\|M - \overline{U}\,\overline{V}^\top\right\|_F^2 = \sum_{i,j}\left(M_{i:j} - \sum_l \overline{U}_{i:l}\overline{V}_{j:l}\right)^2 \\ =&\sum_{i,j}\left(M_{i:j} - \sum_l U^*_{i:l} \cdot \sqrt{\beta_l/\alpha_l} \cdot V^*_{j:l} \cdot \sqrt{\alpha_l/\beta_l}\right)^2 \\ =&\sum_{i,j}\left(M_{i:j} - \sum_l U^*_{i:l} \cdot V^*_{j:l}\right)^2 = \|M - U^*V^{*\top}\|_F^2,\end{aligned} \tag{26}$$

which means that $(\overline{U}, \overline{V})$ is also an optimal solution. In short, for any optimal solution of Eq. 1 outside the domain shown in Eq. 22, there exists a corresponding global optimal solution satisfying the domain shown in Eq. 22, which further means that there exists at least one optimal solution in the domain shown in Eq. 22. □

Appendix B
Proof of Theorem 1

For simplicity, we denote $f(U, V) = \|M - UV^\top\|_F^2$, $\tilde{f}_S = \|MS - U(V^\top S)\|_F^2$, and $\tilde{f}'_{S'} = \|M^\top S' - V(U^\top S')\|_F^2$. Let G^t and $\tilde{G}^t$ denote the gradients of the above quantities, i.e.,

$$\begin{aligned}&G^t \triangleq \nabla_U\ f(U, V^t)\big|_{U=U^t}, \quad \tilde{G}^t \triangleq \nabla_U\ \tilde{f}_{S^t}(U, V^t)\Big|_{U=U^t}, \\ &G'^t \triangleq \nabla_V\ f(U^{t+1}, V)\big|_{V=V^t}, \quad \tilde{G}'^t \triangleq \nabla_V\ \tilde{f}'_{S'^t}(U^{t+1}, V)\Big|_{V=V^t}.\end{aligned} \tag{27}$$

Besides, let

$$\Delta^t \triangleq \frac{1}{\eta_t}\left(U^t - U^{t+1}\right) \quad \text{and} \quad \Delta'^t \triangleq \frac{1}{\eta_t}\left(V^t - V^{t+1}\right). \tag{28}$$

B.1 Preliminary Lemmas

To prove Theorem 1, we need following lemmas (which are proved in Sec. B.3):

Lemma 2. *Under Assumptions 1 and 2, conditioned on U^t and V^t, $\tilde{G}^t$ and $\tilde{G}'^t$ are unbiased estimators of G^t and G'^t respectively with uniformly bounded variance.*

Lemma 3. *Assume X is a nonnegative random variable with mean μ and variance σ^2, and $c \geq 0$ is a constant. Then we have*

$$\mathbb{E}\left[\min\{X, c\}\right] \geq \min\left\{c, \frac{\mu}{2}\right\} \cdot \left(1 - \frac{4\sigma^2}{4\sigma^2 + \mu^2}\right). \tag{29}$$

Lemma 4. *Define the function*

$$\phi(x, y, z) = \min\left\{|xy|, y^2/2\right\} \cdot \left(1 - \frac{4z^2}{4z^2 + y^2}\right) \geq 0. \tag{30}$$

Conditioned on U^t and V^t, there exists an uniform constant $\sigma'^2 > 0$ such that

$$\mathbb{E}[G^t_{i:l} \cdot \Delta^t_{i:l}] \geq \phi\left(U^t_{i:l}/\eta_t, G^t_{i:l}, \sigma'^2\right) \tag{31}$$

and

$$\mathbb{E}[G'^t_{j:l} \cdot \Delta'^t_{j:l}] \geq \phi\left(V^t_{j:l}/\eta_t, G'^t_{j:l}, \sigma'^2\right) \tag{32}$$

for any i, j, l.

Lemma 5 (Supermartingale Convergence Theorem [52])**.** *Let Y_t, Z_t and W_t, $t = 0, 1, \ldots$, be three sequences of random variables and let $\mathcal{F}_t$, $t = 0, 1, \ldots$, be sets of random variables such that $\mathcal{F}_t \subset \mathcal{F}_{t+1}$. Suppose that*

1) *The random variables Y_t, Z_t and W_t are nonnegative, and are functions of the random variables in $\mathcal{F}_t$.*
2) *For each t, we have*

$$\mathbb{E}[Y_{t+1}|\mathcal{F}_t] \leq Y_t - Z_t + W_t. \tag{33}$$

3) *There holds, with probability 1, $\sum_{t=0}^\infty W_t < \infty$.*

Then we have $\sum_{t=0}^\infty Z_t < \infty$, and the sequence Y_t converges to a nonnegative random variable Y, with probability 1.

Lemma 6 ([53])**.** *For two nonnegative scalar sequences $\{a_t\}$ and $\{b_t\}$, if $\sum_{t=0}^\infty a_k = \infty$ and $\sum_{t=0}^\infty a_t b_t < \infty$, then*

$$\liminf_{t\to\infty} b_t = 0. \tag{34}$$

Furthermore, if $|b_{t+1} - b_t| \leq B \cdot a_t$ for some constant $B > 0$, then

$$\lim_{t\to\infty} b_t = 0. \tag{35}$$

B.2 Proof of Theorem 1

Proof of Theorem 1. Let us first focus on projected gradient descent. By conditioning on U^t and V^t, we have

$$\begin{aligned} f(U^{t+1}, V^t) =& \left\| M - U^{t+1} V^{t\top} \right\|_F^2 = \left\| M - \left(U^t - \eta_t \Delta^t\right) V^{t\top} \right\|_F^2 \\ =& \left\| \left(M - U^t V^{t\top}\right) - \eta_t \Delta^t V^{t\top} \right\|_F^2 \\ =& \left\| M - U^t V^{t\top} \right\|_F^2 - 2\eta_t \left(M - U^t V^{t\top}\right) \cdot \left(\Delta^t V^{t\top}\right) \\ &+ \eta_t^2 \|\Delta^t V^{t\top}\|_F^2 \\ =& f(U^t, V^t) - 2\eta_t \left(M - U^t V^{t\top}\right) \cdot \left(\Delta^t V^{t\top}\right) \\ &+ \eta_t^2 \|\Delta^t V^{t\top}\|_F^2. \end{aligned} \tag{36}$$

For the second term of Eq. 36, note that

$$\begin{aligned} 2\left(M - U^t V^{t\top}\right) \cdot \left(\Delta^t V^{t\top}\right) &= 2\mathrm{tr}\left[\left(M - U^t V^{t\top}\right) V^t \Delta^{t\top}\right] \\ &= \mathrm{tr}\left[G^t \Delta^{t\top}\right] = \sum_{i,l} G^t_{i:l} \cdot \Delta^t_{i:l}. \end{aligned} \tag{37}$$

By taking expectation and using Lemma 4, we obtain:

$$\begin{aligned} \mathbb{E}\left[2\left(M - U^t V^{t\top}\right) \cdot \left(\Delta^t V^{t\top}\right)\right] &= \sum_{i,l} \mathbb{E}\left[G^t_{i:l} \cdot \Delta^t_{i:l}\right] \\ &\geq \sum_{i,l} \phi\left(U^t_{i:l}/\eta_t, G^t_{i:l}, \sigma'^2\right). \end{aligned} \tag{38}$$

For simplicity, we will use the notation

$$\Phi(U^t/\eta_t, G^t) \triangleq \sum_{i,l} \phi\left(U^t_{i:l}/\eta_t, G^t_{i:l}, \sigma'^2\right). \tag{39}$$

For the third term of Eq. 36, we can bound it in the following way:

$$\begin{aligned} \|\Delta^t V^{t\top}\|_F^2 \leq& \|\Delta^t\|_F^2 \cdot \|V^t\|_F^2 \leq \|\tilde{G}^t\|_F^2 \cdot \|V^t\|_F^2 \\ =& \left\| 2(U^t V^{t\top} - M)(S^t S^{t\top}) V^t \right\|_F^2 \cdot \|V^t\|_F^2 \\ \leq& 4\|M - U^t V^{t\top}\|_F^2 \cdot \|S^t S^{t\top}\|_F^2 \cdot \|V^t\|_F^4 \\ \leq& 8\left(\|M\|_F^2 + \|U^t\|_F^2 \cdot \|V^t\|_F^2\right) \cdot \|S^t S^{t\top}\|_F^2 \cdot \|V^t\|_F^4 \\ \leq& 8\left(\|M\|_F^2 + R^4\right) R^4 \cdot \|S^t S^{t\top}\|_F^2, \end{aligned} \tag{40}$$

where in the last inequality we have applied Assumption 2. If we take expectation, we have

$$\begin{aligned} \mathbb{E}\|\Delta^t V^{t\top}\|_F^2 \leq& 8\left(\|M\|_F^2 + R^4\right) R^4 \cdot \mathbb{E}\|S^t S^{t\top}\|_F^2 \\ \leq& 8\left(\|M\|_F^2 + R^4\right) R^4 \cdot \left(\left\|\mathbb{E}[S^t S^{t\top}]\right\|^2 + \mathbb{V}[S^t S^{t\top}]\right) \\ \leq& 8\left(\|M\|_F^2 + R^4\right) R^4 \cdot \left(n + \sigma^2\right), \end{aligned} \tag{41}$$

where mean-variance decomposition have been applied in the second inequality, and Assumption 1 was used in the last line. For convenience, we will use

$$\Gamma \triangleq 8\left(\|M\|_F^2 + R^4\right) R^4 \cdot \left(n + \sigma^2\right) \geq 0 \tag{42}$$

to denote this constant later on.

By combining all results, we can rewrite Eq. 36 as

$$\mathbb{E}\left[f(U^{t+1}, V^t)\right] \leq f(U^t, V^t) - \eta_t \Phi\left(U^t/\eta_t, G^t\right) + \eta_t^2 \Gamma. \tag{43}$$

Likewise, conditioned on U^{t+1} and V^t, we can prove a similar inequality for V:

$$\mathbb{E}\left[f(U^{t+1}, V^{t+1})\right] \leq f(U^{t+1}, V^t) - \eta_t \Phi\left(V^t/\eta_t, G'^t\right) + \eta_t^2 \Gamma', \tag{44}$$

where $\Gamma' \geq 0$ is also some uniform constant. From definition, it is easy to see both $\Phi\left(U^t/\eta_t, G^t\right)$ and $\Phi\left(V^t/\eta_t, G'^t\right)$ are nonnegative. Along with condition the condition $\sum_{t=0}^{\infty} \eta_t^2 < \infty$, we can apply the Supermartingale Convergence Theorem (Lemma 5) with

$$\begin{aligned} &Y_{2t} = f(U^t, V^t), \quad Y_{2t+1} = f(U^{t+1}, V^t), \\ &Z_{2t} = \Phi\left(U^t/\eta_t, G^t\right), \quad Z_{2t+1} = \Phi\left(V^t/\eta_t, G'^t\right), \\ &W_{2t} = \Gamma \eta_t^2, \quad W_{2t+1} = \Gamma' \eta_t^2, \end{aligned} \tag{45}$$

and then conclude that both $\{f(U^{t+1}, V^t)\}$ and $\{f(U^t, V^t)\}$ will converge to a same value, and besides:

$$\sum_{t=0}^{\infty} \eta_t \left[\Phi\left(U^t/\eta_t, G^t\right) + \Phi\left(V^t/\eta_t, G'^t\right)\right] < \infty, \tag{46}$$

with probability 1. In addition, it is not hard to verify that $\left|\Phi\left(U^{t+1}/\eta_{t+1}, G^{t+1}\right) - \Phi\left(U^t/\eta_t, G^t\right)\right| \leq C \cdot \eta_t$ for some constant C because of the boundness of the gradients. Then, by Lemma 6, we obtain that

$$\lim_{t\to\infty} \Phi\left(U^t/\eta_t, G^t\right) = \lim_{t\to\infty} \sum_{i:l} \phi\left(U^t_{i:l}/\eta_t, G^t_{i:l}, \sigma'^2\right) \to 0. \tag{47}$$

Since each summand in the above is nonnegative, this equation further implies

$$\lim_{t\to\infty} \phi\left(U^t_{i:l}/\eta_t, G^t_{i:l}, \sigma'^2\right) \to 0 \tag{48}$$

for all i and l. By looking into the definition of ϕ in Eq. 30, it is not hard to see that $\phi\left(U^t_{i:l}/\eta_t, G^t_{i:l}, \sigma'^2\right) \to 0$ if and only if $\min\left\{U^t_{i:l}/\eta_t,\ \left|G^t_{i:l}\right|\right\} \to 0$. Considering $\eta_t > 0$ and $\eta_t \to 0$, we can conclude that

$$\lim_{t\to\infty} \min\left\{U^t_{i:l},\ \left|G^t_{i:l}\right|\right\} \to 0 \tag{49}$$

for all i, l, which means either the gradient $G^t_{i:l}$ converges to 0, or $U^t_{i:l}$ converges to the boundary 0. In other words, the projected gradient at (U^t, V^t) w.r.t U converges to 0 as $t \to \infty$. Likewise, we can prove

$$\lim_{t\to\infty} \min\left\{V^t_{j:l},\ \left|G'^t_{j:l}\right|\right\} \to 0, \tag{50}$$

in a similar way, which completes the proof of projected gradient descent.

The proof of regularized coordinate descent is similar to that of projected gradient descent, and hence we only include a sketch proof here. The key here is to establish an inequality similar to Eq. 36, but with the difference that just one column rather than whole U or V is changed every time. Take $U_{:1}$ as an example. An important observation is that when projection does not happen, we can rewrite (19) in the paper as $U^{t+1}_{:1} = U^t_{:1} - \tilde{G}_{:1}/(\tau_t + B^t_{j:} B^{t\top}_{j:})$, which means that the moving direction of regularized coordinate descent is the same as that of projected gradient descent, but with step size being $1/(\tau_t + B^t_{j:} B^{t\top}_{j:})$. Since both the expectation and variance of $B^t_{j:} B^{t\top}_{j:}$ are bounded, we will have $1/(\tau_t + B^t_{j:} B^{t\top}_{j:}) \approx 1/\tau_t$ when τ_t is large. Given these two reasons, we can out down a similar inequality as Eq. 36. The remaining proof just follows the one for projected gradient descent. □

B.3 Proof of Preliminary Lemmas

Proof of Lemma 2. Since the proof related to $\tilde{G}'^t$ is similar to $\tilde{G}^t$, here we only focus on the latter one.

First, let us write down the definition of G^t and $\tilde{G}^t$:

$$\begin{aligned} G^t &= 2(U^tV^{t\top} - M)V^t \\ \tilde{G}^t &= 2(U^tV^{t\top} - M)(S^tS^{t\top})V^t. \end{aligned} \tag{51}$$

Therefore,

$$\begin{aligned} \mathbb{E}[\tilde{G}^t] =& \mathbb{E}\left[2(U^tV^{t\top} - M)(S^tS^{t\top})V^t\right] \\ =& 2(U^tV^{t\top} - M)\,\mathbb{E}[S^tS^{t\top}]\,V^t \\ =& 2(U^tV^{t\top} - M)\,I\,V^t = 2(U^tV^{t\top} - M)V^t = G^t, \end{aligned} \tag{52}$$

which means $\tilde{G}^t$ is an unbiased estimator of G^t. Besides, its variance is uniformly bounded because

$$\begin{aligned} \mathbb{V}[\tilde{G}^t] \le& \mathbb{V}\left[2(U^tV^{t\top} - M)_{i:}(S^tS^{t\top})V^t_{:l}\right] \\ \le& 4\|M - U^tV^{t\top}\|_F^2 \cdot \mathbb{V}[S^tS^{t\top}] \cdot \|V^t\|_F^2 \\ \le& 8\left(\|M\|_F^2 + \|U^t\|_F^2\|V^t\|_F^2\right) \cdot \|V^t\|_F^2 \cdot \mathbb{V}[S^tS^{t\top}] \\ \le& 8\left(\|M\|_F^2 + R^4\right) R^2 \cdot \sigma^2, \end{aligned} \tag{53}$$

where both Assumptions 1 and 2 are applied in the last line. □

Proof of Lemma 3. In this proof, we will use Cantelli's inequality:

$$\Pr(X \ge \mu + \lambda) \ge 1 - \frac{\sigma^2}{\sigma^2 + \lambda^2} \quad \forall \lambda < 0. \tag{54}$$

When $\mu = 0$, it is easy to see that the right-hand-side of Eq. 29 is 0. Considering that the left-hand-side is the expectation of a nonnegative random variable, Eq. 29 obviously holds in this case.

When $\mu > 0$ and $\mu \ge 2c$, by using the fact that X is nonnegative, we have

$$\mathbb{E}\left[\min\{X, c\}\right] \ge c \cdot \Pr(X \ge c). \tag{55}$$

Now we can apply Cantelli's inequality to bound $\Pr(X \ge c)$ with $\lambda = c - \mu < c - \mu/2 \le 0$, and obtain:

$$\begin{aligned} \mathbb{E}\left[\min\{X, c\}\right] &\ge c \cdot \left(1 - \frac{\sigma^2}{\sigma^2 + (\mu - c)^2}\right) \\ &\ge c \cdot \left(1 - \frac{\sigma^2}{\sigma^2 + (\mu - \mu/2)^2}\right) \\ &= c \cdot \left(1 - \frac{4\sigma^2}{4\sigma^2 + \mu^2}\right), \end{aligned} \tag{56}$$

where in the second inequality we used the fact $c \le \mu/2$ again.

When $\mu > 0$ but $\mu < 2c$, we have:

$$\mathbb{E}\left[\min\{X, c\}\right] \ge \mathbb{E}\left[\min\{X, \mu/2\}\right]. \tag{57}$$

Now we can apply inequality in Eq. 56 from the previous part with $c = \mu/2$, and thus

$$\mathbb{E}\left[\min\{X, c\}\right] \ge \mathbb{E}\left[\min\{X, \mu/2\}\right] \ge \frac{\mu}{2} \cdot \left(1 - \frac{4\sigma^2}{4\sigma^2 + \mu^2}\right), \tag{58}$$

which completes the proof. □

Proof of Lemma 4. We only focus on G^t and Δ^t. We first show that

$$G^t_{i:l} \cdot \tilde{G}^t_{i:l} \ge 0 \tag{59}$$

for any random matrix S^t. Note that

$$\begin{aligned} G^t_{i:l} &= 2(U^tV^{t\top} - M)_{i:}V^t_{:l} \\ \tilde{G}^t_{i:l} &= 2(U^tV^{t\top} - M)_{i:}(S^tS^{t\top})V^t_{:l}. \end{aligned} \tag{60}$$

Hence it would be sufficient if we can show that there holds $a^\top(S^tS^{t\top})b \cdot a^\top b \ge 0$ for any vectors a and b:

$$\begin{aligned} a^\top(S^tS^{t\top})b \cdot a^\top b &= \operatorname{tr}\left(a^\top(S^tS^{t\top})bb^\top a\right) \\ &= \operatorname{tr}\left(aa^\top(S^tS^{t\top})bb^\top\right) \ge 0, \end{aligned} \tag{61}$$

where the first equality is because $A \cdot B = \operatorname{tr}(AB^\top)$, the second equality is due to cyclic permutation invariant property of trace, and the last inequality is because all of $aa^\top$, $bb^\top$ and $S^tS^{t\top}$ are positive semi-definite matrices.

Now, let us consider the relationship between Δ^t and $\tilde{G}^t$:

$$\Delta^t = \frac{1}{\eta_t}\left(U^t - U^{t+1}\right) = \frac{1}{\eta_t}\left(U^t - \max\left\{U^t - \eta_t\tilde{G}^t,\, 0\right\}\right), \tag{62}$$

from which it can be shown that

$$\Delta^t_{i:l} = \min\left\{U^t_{i:l}/\eta_t,\, \tilde{G}^t_{i:l}\right\}. \tag{63}$$

When $G^t_{i:l} = 0$, it is easy to see that both sides of Eq.31 become 0, and hence Eq.31 holds.

When $G^t_{i:l} > 0$, from Eq.59 we know that $\tilde{G}^t_{i:l} \ge 0$ regardless of the choice of S^t. From Lemma 2 we know that

$$\mathbb{E}[\tilde{G}^t_{i:l}] = G^t_{i:l} \tag{64}$$

and there exists a constant $\sigma'^2 \ge 0$ such that

$$\mathbb{V}[\tilde{G}^t_{i:l}] \le \sigma'^2. \tag{65}$$

Since $U^t_{i:l}$ is a nonnegative constant here, we can apply Lemma 3 to Eq.63 and conclude

$$\begin{aligned} \mathbb{E}[\Delta^t_{i:l}] \ge& \min\left\{U^t_{i:l}/\eta_t, G^t_{i:l}/2\right\} \cdot \left(1 - \frac{4\mathbb{V}[\tilde{G}^t_{i:l}]}{4\mathbb{V}[\tilde{G}^t_{i:l}] + \left(G^t_{i:l}\right)^2}\right) \\ \ge& \min\left\{U^t_{i:l}/\eta_t, G^t_{i:l}/2\right\} \cdot \left(1 - \frac{4\sigma'^2}{4\sigma'^2 + \left(G^t_{i:l}\right)^2}\right), \end{aligned} \tag{66}$$

from which Eq.31 is obvious.

When $G^t_{i:l} < 0$, also from Eq. 59 we know that $\tilde{G}^t_{i:l} \le 0$. Since $U^t_{i:l}$ is a nonnegative constant here, we always have

$$\Delta^t_{i:l} = \min\left\{U^t_{i:l}/\eta_t,\, \tilde{G}^t_{i:l}\right\} = \tilde{G}^t_{i:l}. \tag{67}$$

Therefore, by taking expectation and using Lemma 2, we obtain

$$\mathbb{E}[\Delta^t_{i:l}] = \mathbb{E}[\tilde{G}^t_{i:l}] = G^t_{i:l}, \tag{68}$$

and thus

$$\mathbb{E}\left[G^t_{i:l} \cdot \Delta^t_{i:l}\right] = \left(G^t_{i:l}\right)^2 > \frac{\left(G^t_{i:l}\right)^2}{2} \cdot \left(1 - \frac{4\sigma'^2}{4\sigma'^2 + \left(G^t_{i:l}\right)^2}\right) \tag{69}$$

for any constant σ', which means that Eq. 31 holds. □

References

[1] V. P. Pauca, F. Shahnaz, M. W. Berry, and R. J. Plemmons, “Text mining using non-negative matrix factorizations,” in *SDM*, 2004, pp. 452–456.

[2] I. Kotsia, S. Zafeiriou, and I. Pitas, “A novel discriminant non-negative matrix factorization algorithm with applications to facial image characterization problems,” *IEEE Trans. Information Forensics and Security*, vol. 2, no. 3-2, pp. 588–595, 2007.

[3] Q. Gu, J. Zhou, and C. H. Q. Ding, “Collaborative filtering: Weighted nonnegative matrix factorization incorporating user and item graphs,” in *SDM*, 2010, pp. 199–210.

[4] Y. Zhang and D. Yeung, “Overlapping community detection via bounded nonnegative matrix tri-factorization,” in *KDD*, 2012, pp. 606–614.

[5] R. Kannan, G. Ballard, and H. Park, “A high-performance parallel algorithm for nonnegative matrix factorization,” in *PPOPP*, 2016, pp. 9:1–9:11.

[6] Y. Kim, J. Sun, H. Yu, and X. Jiang, “Federated tensor factorization for computational phenotyping,” in *KDD*, 2017, pp. 887–895.

[7] J. Feng, L. T. Yang, Q. Zhu, and K.-K. R. Choo, “Privacy-preserving tensor decomposition over encrypted data in a federated cloud environment,” *IEEE Trans. Dependable Sec. Comput.*, 2018.

[8] R. Kannan, G. Ballard, and H. Park, “MPI-FAUN: an mpi-based framework for alternating-updating nonnegative matrix factorization,” *IEEE Trans. Knowl. Data Eng.*, vol. 30, no. 3, pp. 544–558, 2018.

[9] D. D. Lee and H. S. Seung, “Algorithms for nonnegative matrix factorization,” in *NIPS*, 2000, pp. 556–562.

[10] N. Gillis, “The why and how of nonnegative matrix factorization,” *arXiv Preprint*, 2014. [Online]. Available: https://arxiv.org/abs/1401.5226

[11] M. E. Daube-Witherspoon and G. Muehllehner, “An iterative image space reconstruction algorthm suitable for volume ect,” *IEEE Trans. Med. Imaging*, vol. 5, no. 2, pp. 61–66, 1986.

[12] L. Grippo and M. Sciandrone, “On the convergence of the block nonlinear gauss-seidel method under convex constraints,” *Oper. Res. Lett.*, vol. 26, no. 3, pp. 127–136, 2000.

[13] H. Kim and H. Park, “Nonnegative matrix factorization based on alternating nonnegativity constrained least squares and active set method,” *SIAM J. Matrix Analysis Applications*, vol. 30, no. 2, pp. 713–730, 2008.

[14] J. Kim and H. Park, “Fast nonnegative matrix factorization: An active-set-like method and comparisons,” *SIAM J. Scientific Computing*, vol. 33, no. 6, pp. 3261–3281, 2011.

[15] C. Lin, “Projected gradient methods for nonnegative matrix factorization,” *Neural Computation*, vol. 19, no. 10, pp. 2756–2779, 2007.

[16] R. Zdunek and A. Cichocki, “Non-negative matrix factorization with quasi-newton optimization,” in *ICAISC*, vol. 4029, 2006, pp. 870–879.

[17] N. Guan, D. Tao, Z. Luo, and B. Yuan, “Nenmf: An optimal gradient method for nonnegative matrix factorization,” *IEEE Trans. Signal Processing*, vol. 60, no. 6, pp. 2882–2898, 2012.

[18] M. Naor and K. Nissim, “Communication preserving protocols for secure function evaluation,” in *STOC*, 2001, pp. 590–599.

[19] K. Kanjani, “Parallel non negative matrix factorization for document clustering,” *CPSC-659 (Parallel and Distributed Numerical Algorithms) course. Texas A&M University, Tech. Rep*, 2007.

[20] S. A. Robila and L. G. Maciak, “A parallel unmixing algorithm for hyperspectral images,” in *Intelligent Robots and Computer Vision XXIV: Algorithms, Techniques, and Active Vision*, vol. 6384, 2006, p. 63840F.

[21] C. Liu, H. Yang, J. Fan, L. He, and Y. Wang, “Distributed nonnegative matrix factorization for web-scale dyadic data analysis on mapreduce,” in *WWW*, 2010, pp. 681–690.

[22] R. Liao, Y. Zhang, J. Guan, and S. Zhou, “Cloudnmf: A mapreduce implementation of nonnegative matrix factorization for large-scale biological datasets,” *Genomics, Proteomics & Bioinformatics*, vol. 12, no. 1, pp. 48–51, 2014.

[23] J. Yin, L. Gao, and Z. M. Zhang, “Scalable nonnegative matrix factorization with block-wise updates,” in *ECML/PKDD (3)*, ser. Lecture Notes in Computer Science, vol. 8726, 2014, pp. 337–352.

[24] X. Meng, J. K. Bradley, B. Yavuz, E. R. Sparks, S. Venkataraman, D. Liu, J. Freeman, D. B. Tsai, M. Amde, S. Owen, D. Xin, R. Xin, M. J. Franklin, R. Zadeh, M. Zaharia, and A. Talwalkar, “Mllib: Machine learning in apache spark,” *J. Mach. Learn. Res.*, vol. 17, pp. 34:1–34:7, 2016.

[25] D. Grove, J. Milthorpe, and O. Tardieu, “Supporting array programming in X10,” in *ARRAY@PLDI*, 2014, pp. 38–43.

[26] E. Mejía-Roa, D. Tabas-Madrid, J. Setoain, C. García, F. Tirado, and A. D. Pascual-Montano, “Nmf-mgpu: non-negative matrix factorization on multi-gpu systems,” *BMC Bioinformatics*, vol. 16, pp. 43:1–43:12, 2015.

[27] R. M. Gower and P. Richtárik, “Randomized iterative methods for linear systems,” *SIAM J. Matrix Analysis Applications*, vol. 36, no. 4, pp. 1660–1690, 2015.

[28] M. Pilanci and M. J. Wainwright, “Iterative hessian sketch: Fast and accurate solution approximation for constrained least-squares,” *J. Mach. Learn. Res.*, vol. 17, pp. 53:1–53:38, 2016.

[29] M. Pilanci and M. J. Wainwright, “Newton sketch: A near linear-time optimization algorithm with linear-quadratic convergence,” *SIAM Journal on Optimization*, vol. 27, no. 1, pp. 205–245, 2017.

[30] F. Wang and P. Li, “Efficient nonnegative matrix factorization with random projections,” in *SDM*, 2010, pp. 281–292.

[31] Y. Lindell and B. Pinkas, “Privacy preserving data mining,” in *CRYPTO*, vol. 1880, 2000, pp. 36–54.

[32] L. Wan, W. K. Ng, S. Han, and V. C. S. Lee, “Privacy-preservation for gradient descent methods,” in *KDD*, 2007, pp. 775–783.

[33] S. Han, W. K. Ng, L. Wan, and V. C. S. Lee, “Privacy-preserving gradient-descent methods,” *IEEE Trans. Knowl. Data Eng.*, vol. 22, no. 6, pp. 884–899, 2010.

[34] M. A. Pathak and B. Raj, “Privacy preserving protocols for eigenvector computation,” in *PSDML*, vol. 6549, 2010, pp. 113–126.

[35] S. Han, W. K. Ng, and P. S. Yu, “Privacy-preserving singular value decomposition,” in *ICDE*, 2009, pp. 1267–1270.

[36] S. Chen, R. Lu, and J. Zhang, “A flexible privacy-preserving framework for singular value decomposition under internet of things environment,” in *IFIPTM*, vol. 505, 2017, pp. 21–37.

[37] J. Sakuma and S. Kobayashi, “Large-scale *k*-means clustering with user-centric privacy-preservation,” *Knowl. Inf. Syst.*, vol. 25, no. 2, pp. 253–279, 2010.

[38] Z. Lin and J. W. Jaromczyk, “Privacy preserving spectral clustering over vertically partitioned data sets,” in *FSKD*, 2011, pp. 1206–1211.

[39] Y. Duan and J. F. Canny, “Practical private computation and zero-knowledge tools for privacy-preserving distributed data mining,” in *SDM*. SIAM, 2008, pp. 265–276.

[40] Z. Beerliová-Trubíniová and M. Hirt, “Perfectly-secure MPC with linear communication complexity,” in *TCC*, vol. 4948, 2008, pp. 213–230.

[41] I. Damgård and J. B. Nielsen, “Scalable and unconditionally secure multiparty computation,” in *CRYPTO*, vol. 4622, 2007, pp. 572–590.

[42] N. Ailon and B. Chazelle, “Approximate nearest neighbors and the fast johnson-lindenstrauss transform,” in *STOC*, 2006, pp. 557–563.

[43] Y. Lu, P. S. Dhillon, D. P. Foster, and L. H. Ungar, “Faster ridge regression via the subsampled randomized hadamard transform,” in *NIPS*, 2013, pp. 369–377.

[44] K. L. Clarkson and D. P. Woodruff, “Low rank approximation and regression in input sparsity time,” in *STOC*, 2013, pp. 81–90.

[45] N. Pham and R. Pagh, “Fast and scalable polynomial kernels via explicit feature maps,” in *KDD*, 2013, pp. 239–247.

[46] A. Nemirovski, A. Juditsky, G. Lan, and A. Shapiro, “Robust stochastic approximation approach to stochastic programming,” *SIAM J. on Optim.*, vol. 19, no. 4, pp. 1574–1609, 2009.

[47] R. T. Rockafellar, “Monotone operators and the proximal point algorithm,” *SIAM J. Control Optim.*, vol. 14, no. 5, pp. 877–898, 1976.

[48] J. P. Fairbanks, R. Kannan, H. Park, and D. A. Bader, “Behavioral clusters in dynamic graphs,” *Parallel Computing*, vol. 47, pp. 38–50, 2015.

[49] Y. Qian, C. Tan, N. Mamoulis, and D. W. Cheung, “DSANLS: accelerating distributed nonnegative matrix factorization via sketching,” in *WSDM*, 2018, pp. 450–458.

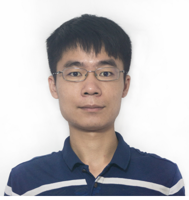
[50] S. Boyd, L. Xiao, and A. Mutapcic, “Subgradient methods,” *Stanford University*, 2003. [Online]. Available: https://web.stanford.edu/class/ee392o/subgrad_method.pdf

[51] J. Kim, Y. He, and H. Park, “Algorithms for nonnegative matrix and tensor factorizations: a unified view based on block coordinate descent framework,” *J. Global Optimization*, vol. 58, no. 2, pp. 285–319, 2014.

[52] J. Neveu, *Discrete-parameter martingales*. Elsevier, 1975, vol. 10.

[53] J. Mairal, “Stochastic majorization-minimization algorithms for large-scale optimization,” in *NIPS*, 2013, pp. 2283–2291.

Yuqiu Qian is currently an applied researcher in Tencent. Her research interests include data engineering and machine learning with applications in recommender systems. She received her B.Eng. degree in Computer Science from University of Science and Technology of China (2015), and her PhD degree in Computer Science from University of Hong Kong (2019).



Conghui Tan is currently a researcher in WeBank. His research interests include machine learning and optimization algorithms. He received his B.Eng. degree in Computer Science from University of Science and Technology of China (2015), and his PhD degree in System Engineering from Chinese University of Hong Kong (2019).

Danhao Ding is currently a PhD candidate in Department of Computer Science, University of Hong Kong. His research interest include high performance computing and machine learning. He received his B.Eng. degree in Computing and Data Analytics from University of Hong Kong (2016).

Hui Li is currently an assistant professor in the School of Informatics, Xiamen University. His research interests include data mining and data management with applications in recommender systems and knowledge graph. He received his B.Eng. degree in Software Engineering from Central South University (2012), and his MPhil and PhD degrees in Computer Science from University of Hong Kong (2015, 2018).

Nikos Mamoulis received his diploma in computer engineering and informatics in 1995 from the University of Patras, Greece, and his PhD in computer science in 2000 from HKUST. He is currently a faculty member at the University of Ioannina. Before, he was professor at the Department of Computer Science, University of Hong Kong. His research focuses on the management and mining of complex data types.