KYBERNETIKA — VOLUME 57 (2021), NUMBER 1, PAGES 60–77

MULTI-AGENT SOLVER FOR NON-NEGATIVE MATRIX FACTORIZATION BASED ON OPTIMIZATION

Zhipeng Tu and Weijian Li

This paper investigates a distributed solver for non-negative matrix factorization (NMF) over a multi-agent network. After reformulating the problem into the standard distributed optimization form, we design our distributed algorithm (DisNMF) based on the primal-dual method and in the form of multiplicative update rule. With the help of auxiliary functions, we provide monotonic convergence analysis. Furthermore, we show by computational complexity analysis and numerical examples that our distributed NMF algorithm performs well in comparison with the centralized NMF algorithm.

Keywords: distributed optimization, non-negative matrix factorization, multiplicative update rules, multi-agent network

Classification: 15A23, 68W15

1. INTRODUCTION

In recent years, distributed computation has attracted considerable attention since the increasing scale of data causes centralized algorithms to be inefficient [1, 19, 24]. Both discrete-time and continuous-time algorithms [23, 26, 28] have been investigated for distributed optimization with various types of constraints. Matrix computation is a promising technique for big data processing, and efficient distributed algorithms for matrix computation are worth of study.

One of the most intriguing matrix computation problems lies in Non-negative Matrix Factorization (NMF), which is to find two low rank factors $X \in \mathbb{R}_+^{p\times d}$ and $Y \in \mathbb{R}_+^{d\times q}$ for a given matrix $M \in \mathbb{R}_+^{p\times q}$, such that $M \approx XY$. To solve it, the following optimization problem is considered

$$\begin{aligned} \min_{X,Y} \quad & \frac{1}{2}\|M - XY\|_F^2 \\ \text{s.t.} \quad & X \geq 0, Y \geq 0 \quad , \end{aligned} \tag{1}$$

which is NP-hard and non-convex. Generally, d is chosen to be far less than p or q, so that the dimensions of X and Y are smaller than that of the original matrix M. This results in a compressed version of the original data matrix. Furthermore, X can be

DOI: 10.14736/kyb-2021-1-0060

regarded as the basis matrix that is optimized for the linear approximation of the data in M. NMF is widely used among matrix computation due to its special property; it can learn a part-representation of the data [10], such as parts of faces and semantic features of text. In fact, formulation (1) arises in many up-to-date applications like joint matrix factorization [5], multi-view clustering [15] and matrix completion [21].

Due to the importance of NMF, centralized algorithms for it have been widely investigated, such as the projected gradient method (PG) [13], the active set method (AS) [9], the multiplicative update rule method (MUR) [11], etc. However, centralized algorithms will become inefficient when the dimension of the matrices increasing, which is known as the curse of dimensionality. Indeed, there comes a few works in distributed solver for NMF. In [14], Liu partitioned the data and rearranged the computation, then factorizing million-by-million matrices with billions of nonzero values became feasible on distributed MapReduce clusters; while Yin leveraged block-wise update functions, which can perform local aggregation and thus have an efficient MapReduce implementation for NMF [25]. Whereas, previous works have not explicitly addressed the issue of information change and communication cost in practice.

In some practical situations, we consider certain agents in a network together to solve a NMF problem. Due to the limited energy, privacy protection (e. g. credit card fraud), and other communication constraints, it is impossible for agents in the network to exchange their sub-datasets. In order to fulfill a given task, each agent over the network needs to work out his sub-task and communicate some information with its local neighbors as well. Thus, multi-agent networks are taken into account [16, 18]. Many algorithms and good results have bee obtained. [20, 27] consider the problem of solving a linear algebraic equation $Ax = b$ in a distributed way by a multi-agent system. [3, 29] solve distributed computation for matrix equations such as $AXB = F$ over multi-agent networks. [12] minimizes the nuclear norm under linear equality constraints over a multi-agent network. Nevertheless, few attention is paid to non-negative matrix factorization.

The main purpose of this paper is to design a distributed solver for non-negative matrix factorization over a multi-agent network. Divide M into n parts, and we focus on decomposing M_is into a common basis matrix X and different coefficient matrices Y_is. Main contributions of our work include:

- To solve NMF of the form $\sum_{i=1}^{n} M_i = \sum_{i=1}^{n} XY_i$, we propose a distributed computation design, where each agent i only knows M_i, works out his sub-task of decomposing M_i into X_i and Y_i, and shares X_i with its neighbors. Then, all agents obtain a consensus public matrix X^* and their private matrices Y_i.

- By using distributed constrained optimization reformulation, we design our distributed algorithm for non-negative matrix factorization (DisNMF) based on the primal-dual method and the multiplicative update rule, which is easy to implement.

- For the distributed algorithm proposed, we provide monotonic convergence analysis by using auxiliary functions. Also, we show by computational complexity analysis and numerical examples that the performance of our distributed algorithm matches that of centralized NMF algorithms.

The remainder of this paper is organized as follows. Section 2 provides necessary preliminary knowledge for our study. Next, the problem is formulated and a distributed algorithm is proposed in Section 3. Section 4 provides the convergence analysis and computational complexity analysis, whereas Section 5 gives numerical examples for illustration. Finally, some concluding remarks are given in Section 6.

2. MATHEMATICAL PRELIMINARIES

In this section, we introduce necessary preliminaries about matrix, graph and optimization.

2.1. Matrix

Denote $\mathbb{R}$, $\mathbb{R}^n$, $\mathbb{R}^{p\times q}$, $\mathbb{R}_+^{p\times q}$ as the set of real numbers, n-dimensional real column vectors, p-by-q real matrices and p-by-q non-negative matrices, respectively. For $M \in \mathbb{R}^{p\times q}$, denote $\mathrm{rank}(M)$, $\ker(M)$, $\mathrm{tr}(M)$, M^{-1}, $M^\top$ as the rank, kernel, trace, inverse, transpose of M, respectively. Denote $M_{s\cdot} \in \mathbb{R}^q$ as the sth row of M, $M_{\cdot t} \in \mathbb{R}^p$ as the tth column of M, and $M_{st} \in \mathbb{R}$ as the (s,t) element of M. A matrix is non-negative (write as $M \geq 0$) means all elements of M is non-negative, that is $M_{st} \geq 0$. For $M_i \in \mathbb{R}^{p\times q}, i \in \{1,\cdots,n\}$, denote $\mathrm{col}\{M_1,\ldots,M_n\} \in \mathbb{R}^{np\times q}$ as the matrix defined by stacking M_i together in columns. Write 1_n for the n-dimensional vector with all elements of 1, and $0_{p\times q}$ for the p-by-q matrix with all elements of 0.

Furthermore, denote $\|\cdot\|_F$ as the Frobenius norm of real matrices, defined by $\|M\|_F = \sqrt{\mathrm{tr}(M^\top M)} = \sqrt{\sum_{i,j} M_{ij}^2}$. Let $\langle\cdot,\cdot\rangle_F$ be the Frobenius inner product of real matrices, defined by $\langle M_1, M_2\rangle_F = \mathrm{tr}\left(M_1^\top M_2\right) = \sum_{i,j} (M_1)_{ij} (M_2)_{ij}$, for $M_1, M_2 \in \mathbb{R}^{p\times q}$. We have $\langle M_1 M_2, M_3\rangle_F = \left\langle M_1, M_3 M_2^\top\right\rangle_F = \left\langle M_2, M_1^\top M_3\right\rangle_F$, for $M_1 \in \mathbb{R}^{p\times r}, M_2 \in \mathbb{R}^{r\times q}, M_3 \in \mathbb{R}^{p\times q}$. Let $M_1 \otimes M_2$ be the Kronecker product of M_1 and M_2.

2.2. Graph theory

A multi-agent network can be described by an undirected graph $\mathcal{G}(\mathcal{V},\mathcal{E})$, where $\mathcal{V} = \{1,\ldots,n\}$ is the set of nodes, representing the set of agents, and $\mathcal{E} \subset \mathcal{V}\times\mathcal{V}$ is the set of edges. Let $A = [a_{ij}] \in \mathbb{R}^{n\times n}$ be the adjacency matrix of $\mathcal{G}$ such that $a_{ij} = a_{ji}$. If $(v_i, v_j) \in \mathcal{E}$, then v_i and v_j can exchange information, and $a_{ij} = 0$ otherwise. We also assume that there are no self-loops, that is $a_{ii} = 0$. The Laplacian matrix is $L = D - A$, where $D \in \mathbb{R}^{n\times n}$ is a diagonal matrix with $D_{ii} = \sum_{j=1}^n a_{ij}$. A path between nodes v_i and v_j is defined as a sequence of edges $(v_i, v_{i1}), (v_{i1}, v_{i2}), \cdots, (v_{ik}, v_j) \in \mathcal{E}$ with distinct nodes $v_{il} \in \mathcal{V}$. The graph is connected if there exists a path between any pair of distinct nodes v_i and v_j. The Laplacian matrix has an important property, which helps to reformulate distributed constraint on optimization later:

Lemma 2.1. (Godsil and Royle [4]) If the undirected graph $\mathcal{G}$ is connected, then $L = L^\top \geq 0$, $\mathrm{rank}(L) = n-1$ and $\ker(L) = \{k1_n : k \in \mathbb{R}\}$.

2.3. Optimization

Since the NMF problem we handle here is non-convex, we adopt some basic optimization concepts [8] for later analysis.

Definition 2.2. *(Joint Convexity and Marginal Convexity)* A continuously differentiable function in two variables $f : \mathbb{R}^p \times \mathbb{R}^q \to \mathbb{R}$ is considered *jointly convex*, if for every $\left(\mathbf{x}^1, \mathbf{y}^1\right), \left(\mathbf{x}^2, \mathbf{y}^2\right) \in \mathbb{R}^p \times \mathbb{R}^q$, we have

$$f\left(\mathbf{x}^2, \mathbf{y}^2\right) \geq f\left(\mathbf{x}^1, \mathbf{y}^1\right) + \left\langle \nabla f\left(\mathbf{x}^1, \mathbf{y}^1\right), \left(\mathbf{x}^2, \mathbf{y}^2\right) - \left(\mathbf{x}^1, \mathbf{y}^1\right) \right\rangle ,$$

where $\nabla f\left(\mathbf{x}^1, \mathbf{y}^1\right)$ is the gradient of f at the point $\left(\mathbf{x}^1, \mathbf{y}^1\right)$. f is considered *marginally convex* in its first variable, if for every value of $\mathbf{y} \in \mathbb{R}^q$, the function $f(\cdot, \mathbf{y}) : \mathbb{R}^p \to \mathbb{R}$ is convex, i. e., for every $\mathbf{x}^1, \mathbf{x}^2 \in \mathbb{R}^p$, we have

$$f\left(\mathbf{x}^2, \mathbf{y}\right) \geq f\left(\mathbf{x}^1, \mathbf{y}\right) + \left\langle \nabla_{\mathbf{x}} f\left(\mathbf{x}^1, \mathbf{y}\right), \mathbf{x}^2 - \mathbf{x}^1 \right\rangle ,$$

where $\nabla_{\mathbf{x}} f\left(\mathbf{x}^1, \mathbf{y}\right)$ is the partial gradient of f with respect to its first variable at the point $(\mathbf{x}^1, \mathbf{y})$. A similar condition is imposed for f to be considered marginally convex in its second variable.

Definition 2.3. *(Marginally Optimum Coordinate and Bistable Point)* Let f be a function of two variables constrained to be in the sets $\mathbb{X}, \mathbb{Y}$ respectively. For any point $\mathbf{y} \in \mathbb{Y}$, we say that $\widetilde{\mathbf{x}}$ is a *marginally optimal coordinate* with respect to $\mathbf{y}$, and use the shorthand $\widetilde{\mathbf{x}} \in mOPT_f(\mathbf{y})$, if $f(\widetilde{\mathbf{x}}, \mathbf{y}) \leq f(\mathbf{x}, \mathbf{y})$ for all $\mathbf{x} \in \mathbb{X}$. Similarly definition for $\widetilde{\mathbf{y}} \in mOPT_f(\mathbf{x})$. A point $(\mathbf{x}, \mathbf{y}) \in \mathbb{X} \times \mathbb{Y}$ is considered a *bistable point* if $\mathbf{x} \in mOPT_f(\mathbf{y})$ and $\mathbf{y} \in mOPT_f(\mathbf{x})$.

We will make use of the auxiliary function similar to that used in [11], which is defined as follows.

Definition 2.4. *(Auxiliary Function)* $G(x, x')$ is an auxiliary function for $F(x)$ if the conditions

$$G(x, x') \geq F(x), \quad G(x', x') = F(x')$$

are satisfied.

Lemma 2.5. If $G(x, x')$ is an auxiliary function of $F(x)$, then $F(x)$ is non-increasing under the update

$$x^{k+1} = \arg\min_x G\left(x, x^k\right). \tag{2}$$

P r o o f. $F\left(x^{k+1}\right) \leq G\left(x^{k+1}, x^k\right) \leq G\left(x^k, x^k\right) = F\left(x^k\right)$. □

The equality $F(x^{k+1}) = F(x^k)$ holds only if x^k is a local minimum of $G(x, x^k)$. By iterating the updates in (2), the sequence will converge to a local minimum $x_{\min} = \arg\min_x F(x)$ [11].

3. PROBLEM FORMULATION AND ALGORITHM DESIGN

In this section, we reformulate the problem of distributed computation for NMF over a multi-agent network as a standard distributed optimization problem, and we propose a distributed algorithm for the reformulation.

3.1. Problem formulation

This paper considers the distributed computation for (1) over a multi-agent network, where each agent just knows partial information of matrix M. As Figure 1 shows, the to-be-factorized matrix M is partitioned into n parts as

$$M = [M_1, M_2, \ldots, M_n] \in \mathbb{R}_+^{p\times q}.$$

where $M_i \in \mathbb{R}_+^{p\times q_i}$ and $\sum_{i=1}^n q_i = q$.

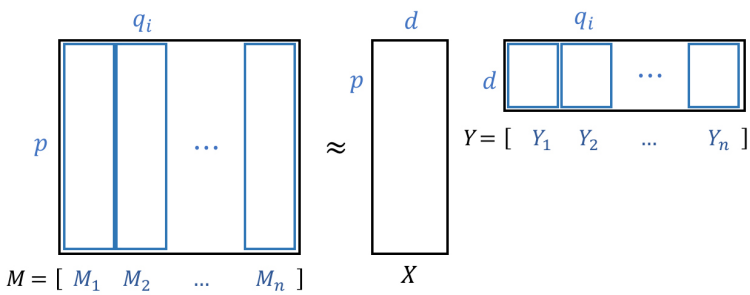


Fig. 1. Distributed Matrix Factorization.

Consider the problem over a multi-agent network of n agents, where each agent i only knows M_i. Here, we introduce a public matrix $X \in \mathbb{R}_+^{p\times d}$ for all agents and a private matrix $Y = [Y_1, Y_2, \ldots, Y_n] \in \mathbb{R}_+^{d\times q}$, where $Y_i \in \mathbb{R}_+^{d\times q_i}$ is held by agent i. M_i and Y_i are not shared between agents for privacy protection. Then (1) can be rewritten as

$$\begin{aligned} \min_{X,Y_i} \quad & \frac{1}{2}\sum_{i=1}^{n} \|M_i - XY_i\|_F^2 \\ \text{s.t.} \quad & X \geq 0, Y_i \geq 0 \quad i \in \{1, \ldots, n\}. \end{aligned} \tag{3}$$

In fact, formulation (3) arises in many up-to-date applications, including the following three problems.

(1) *Joint Matrix Factorization* [5]: Many real-world datasets consist of different views which provide complementary evidence to each other. To find the links among these representations, we can decompose these view matrices into a common basis matrix and different coefficient matrices. Formally, given n views denoted as $\{M_1, \ldots, M_n\}$, each view is factorized as $M_i \approx X_i Y_i$, where X_is are of the same dimension $p \times d$ for all views, while Y_is are of the dimension $d \times q_i$, differing per view. The joint matrix factorization problem can be cast into (3).

(2) *Multi-view Clustering* [15]: Multi-view clustering aims to integrate information from multiple views in the unsupervised setting. The basic idea is that a data point in different views would be assigned to the same cluster with high probability. With the help of matrix factorization, coefficient matrices learnt from different views are required to be a common consensus, which is considered to reflect the latent clustering structure shared by different views. Just as the above joint matrix factorization problem, given n views denoted as $\{M_1, \ldots, M_n\}$, each view is factorized as $M_i \approx X_i Y_i$. Here, Y_is are of the same shape but X_is can differ along the row dimension across multiple views. This problem can be formulated as

$$\begin{aligned} \min_{X_i, Y} \quad & \frac{1}{2} \sum_{i=1}^{n} \|M_i - X_i Y\|_F^2 \\ \text{s.t.} \quad & X_i \geq 0, Y \geq 0 \quad i \in \{1, \ldots, n\}, \end{aligned} \tag{4}$$

which is the same in principle with (3).

(3) *Non-negative Matrix Completion* [21]: This problem can be defined as recovering the missing elements of an incomplete matrix $M \in \mathbb{R}_+^{p \times q}$ with given elements $(M_{ij})_{(i,j)\in\Omega}$, where Ω is the index set of those known elements. Referring to [22], the problem can be formulated as:

$$\begin{aligned} \min_{X} \quad & \operatorname{rank}(X) \\ \text{s.t.} \quad & X_{ij} = M_{ij} \geq 0, \quad (i,j) \in \Omega\,, \end{aligned} \tag{5}$$

which is NP-hard and non-convex. One way to minimize $\operatorname{rank}(X)$ is to minimize the nuclear norm $\|X\|_*$. [12] designs a distributed solver to minimize the nuclear norm under linear equality constraints over a multi-agent network, while it is time-consuming since it involves matrix Singular Value Decomposition (SVD) operation, which is increasingly costly as matrix sizes and increase. Another promising way to minimize rank is through matrix factorization. If M can be represented by the product of two matrices, that is $M = XY, X \in \mathbb{R}_+^{p \times d}, Y \in \mathbb{R}_+^{d \times q}$, then $\operatorname{rank}(XY) \leq d$. d is a latent dimension to be determined. Then the non-negative matrix completion problem can be formulated as:

$$\begin{aligned} \min_{X, Y} \quad & \frac{1}{2} \|P_\Omega(M) - P_\Omega(XY)\|_F^2 \\ \text{s.t.} \quad & X \geq 0, Y \geq 0, \end{aligned} \tag{6}$$

where $P_\Omega(M) - P_\Omega(XY)$ means $M - XY$ in every observed elements, and the objective function is actually equivalent to $\sum_{(i,j)\in\Omega} (M_{ij} - (XY)_{ij})^2$. Problem (6) is similar to problem (1), which can be rewritten as the distributed form like (3).

Obviously, (3) is not a standard distributed structure. To reformulate it, each agent over the network works out his own sub-task of decomposing M_i into X_i and Y_i, and shares X_i with its neighbors, as (7) describes.

$$\begin{aligned} \min_{X_i,Y_i} \quad & \frac{1}{2}\sum_{i=1}^{n}\|M_i - X_iY_i\|_F^2 \\ \text{s.t.} \quad & X_i \geq 0, Y_i \geq 0, \\ & X_i = X_j, \qquad\qquad i,j \in \{1,\ldots,n\}. \end{aligned} \tag{7}$$

A graph $\mathcal{G}(\mathcal{V},\mathcal{E})$ is used to describe the multi-agent network, and its Laplacian matrix is $L = D - A$. To guarantee all agents obtaining a consensus solution X^*, the following assumption is made, which has been widely used in the literatures.

Assumption 3.1. The graph of the multi-agent network is undirected and connected.

Under Assumption 3.1, by Lemma 2.1, $\ker(L) = k1_n$. As a result, $X_i = X_j$ is equivalent to the equation $(L \otimes I_p)\mathcal{X} = 0$, where $\mathcal{X} = \text{col}\{X_1,\ldots,X_n\}$, in other words, $\sum_{j=1}^{n} a_{ij}(X_i - X_j) = 0, i \in \{1,\cdots,n\}$. In consequence, problem (7) is reformulated as

$$\begin{aligned} \min_{X_i,Y_i} \quad & F(\mathcal{X},\mathcal{Y}) = \frac{1}{2}\sum_{i=1}^{n}\|M_i - X_iY_i\|_F^2 \\ \text{s.t.} \quad & X_i \geq 0, Y_i \geq 0, \\ & \sum_{j=1}^{n} a_{ij}(X_i - X_j) = 0, \qquad\qquad i \in \{1,\cdots,n\}, \end{aligned} \tag{8}$$

where $\mathcal{X} = \text{col}\{X_1,\cdots,X_n\}$, $\mathcal{Y} = \text{col}\{Y_1,\cdots,Y_n\}$. The objective function $F(\mathcal{X},\mathcal{Y})$ is not jointly convex in $\mathcal{X}$ and $\mathcal{Y}$, nevertheless, marginally convex in both the variables. Fixing either $\mathcal{X}$ or $\mathcal{Y}$ reduces the above problem to a least squares problem. The following theorem shows the relationship of the optimal solutions between (3) and (8), whose proof is straightforward by the Karush-Kuhn-Tucker optimal conditions and is omitted here.

Lemma 3.2. Under Assumption 3.1, $(X^*,\mathcal{Y}^*)$ is an optimal solution to problem (3) if and only if $(\mathcal{X}^*,\mathcal{Y}^*)$ is an optimal solution to problem (8), where $\mathcal{X}^* = \text{col}\{X_1^*,\ldots,X_n^*\}$ and $\mathcal{Y}^* = \text{col}\{Y_1^*,\ldots,Y_n^*\}$ such that $X_i^* = X^*$ for $i \in \{1,\ldots,n\}$.

3.2. Distributed algorithm design

In this subsection, we propose a distributed algorithm for the reformulation (8). Let $\Psi_i, \Phi_i, \Lambda_i$ be the Lagrange multipliers for the constraints $X_i \geq 0, Y_i \geq 0$ and $\sum_{j=1}^{n} a_{ij}(X_i - X_j) = 0$, $i \in \{1,\cdots,n\}$, respectively. The Lagrange function $\overline{\mathcal{L}}$ of (8) is defined as

$$\begin{aligned} \overline{\mathcal{L}}(\mathcal{X},\mathcal{Y},\Lambda,\Psi,\Phi) = \frac{1}{2}\sum_{i=1}^{n}(&\|M_i - X_iY_i\|_F^2 + \langle\Lambda_i, \sum_{j=1}^{n} a_{ij}(X_i - X_j)\rangle_F \\ &+ \langle\Psi_i, X_i\rangle_F + \langle\Phi_i, Y_i\rangle_F), \end{aligned} \tag{9}$$

where $\Lambda = \text{col}\{\Lambda_1,\cdots,\Lambda_n\}$, $\Psi = \text{col}\{\Psi_1,\cdots,\Psi_n\}$, $\Phi = \text{col}\{\Phi_1,\cdots,\Phi_n\}$, $\Psi_i \leq 0$, $\Phi_i \leq 0$, $i \in \{1,\cdots,n\}$. According to the Karush-Kuhn-Tucker optimal conditions, the

stationarity condition and complementary slackness condition require that

$$\begin{cases} \dfrac{\partial \overline{\mathcal{L}}}{\partial X_i} = -\left(M_i - X_i Y_i\right) Y_i^\top + (L_{i\cdot} \otimes I_p)\Lambda + \Psi_i = 0\,, \\ \dfrac{\partial \overline{\mathcal{L}}}{\partial Y_i} = -X_i^\top \left(M_i - X_i Y_i\right) + \Phi_i = 0\,, \\ (\Psi_i)_{st}(X_i)_{st} = 0\,, \\ (\Phi_i)_{st}(Y_i)_{st} = 0\,. \end{cases} \tag{10}$$

Since $L_{i\cdot} = D_{i\cdot} - A_{i\cdot}$, we get the following equations for $(X_i)_{st}$ and $(Y_i)_{st}$:

$$\begin{aligned} &\left(M_i Y_i^T - X_i Y_i Y_i^T - (D_{i\cdot} \otimes I_p)\Lambda + (A_{i\cdot} \otimes I_p)\Lambda\right)_{st} (X_i)_{st} = 0\,, \\ &\left(X_i^T M_i - X_i^T X_i Y_i\right) (Y_i)_{st} = 0\,. \end{aligned}$$

These equations lead to the following updating rules:

$$(X_i)_{st} \leftarrow (X_i)_{st} \frac{\left(M_i Y_i^\top + (A_{i\cdot} \otimes I_p)\Lambda\right)_{st}}{\left(X_i Y_i Y_i^\top + (D_{i\cdot} \otimes I_p)\Lambda\right)_{st}} = (X_i)_{st} \frac{\left(M_i Y_i^\top + \sum_{j=1}^n a_{ij}\Lambda_j\right)_{st}}{\left(X_i Y_i Y_i^\top + (\sum_{j=1}^n a_{ij})\Lambda_i\right)_{st}}\,, \tag{11}$$

$$(Y_i)_{st} \leftarrow (Y_i)_{st} \frac{\left(X_i^\top M_i\right)_{st}}{\left(X_i^\top X_i Y_i\right)_{st}}\,. \tag{12}$$

As for the Lagrange multiplier Λ, we apply gradient method to each element $(\Lambda_i)_{st}$:

$$\begin{aligned} (\Lambda_i)_{st} \leftarrow (\Lambda_i)_{st} + (\eta_i)_{st} \frac{\partial \mathcal{L}}{\partial (\Lambda_i)_{st}} &= (\Lambda_i)_{st} + (\eta_i)_{st}((L_{i\cdot} \otimes I_p)\mathcal{X})_{st} \\ &= (\Lambda_i)_{st} + (\eta_i)_{st}\left(((D_{i\cdot} \otimes I_p)\mathcal{X})_{st} - ((A_{i\cdot} \otimes I_p)\mathcal{X})_{st}\right)\,, \end{aligned} \tag{13}$$

where $(\eta_i)_{st}$ is a step-size. Particularly, if we take $(\eta_i)_{st} = \frac{(\Lambda_i)_{st}}{((A_{i\cdot} \otimes I_p)\mathcal{X})_{st}}$, then

$$(\Lambda_i)_{st} \leftarrow (\Lambda_i)_{st} \frac{((D_{i\cdot} \otimes I_p)\mathcal{X})_{st}}{((A_{i\cdot} \otimes I_p)\mathcal{X})_{st}} = (\Lambda_i)_{st} \frac{((\sum_{j=1}^n a_{ij})X_i)_{st}}{(\sum_{j=1}^n a_{ij} X_j)_{st}}\,. \tag{14}$$

Our distributed NMF algorithm (written as DisNMF) is formed by the above iterative updating rules (11), (12), (14) and its pseudo-code is shown in 1. In each iteration, every agent updates its X_i, Y_i, Λ_i once. Notice that update rules are multiplicative, when the initial values $(X_i)_{st}^0, (Y_i)_{st}^0, (\Lambda_i)_{st}^0$ are non-negative, X_i, Y_i, Λ_i will be non-negative matrices all the time. This lets us produce a non-negative factorization for M.

Remark 3.3. One can also use the constraint $\langle \mathcal{X}, L\mathcal{X} \rangle_F = 0$ for X_is consensus, since $\langle \mathcal{X}, L\mathcal{X} \rangle_F = \frac{1}{2}\sum_{i,j=1}^n a_{ij} \left\| X_i - X_j \right\|_F^2 \geq 0$, and the equality holds if and only if $X_i =$

Algorithm 1 Distributed Non-negative Matrix Factorization (DisNMF)

Input: $\{M_i\}_{i=1}^n$: data matrices; A: adjacency matrix; d: latent dimension;

initial $\{X_i\}_{i=1}^n, \{Y_i\}_{i=1}^n, \{\Lambda_i\}_{i=1}^n$;
repeat
  for $i = 1 : n$ **do**
    compute $X_i = X_i \,.*\, \frac{M_i Y_i^\top + \sum_{j=1}^n a_{ij}\Lambda_j}{X_i Y_i Y_i^\top + (\sum_{j=1}^n a_{ij})\Lambda_i}$;
    compute $Y_i = Y_i \,.*\, \frac{X_i^\top M_i}{X_i^\top X_i Y_i}$;
    compute $\Lambda_i = \Lambda_i \,.*\, \frac{(\sum_{j=1}^n a_{ij})X_i}{\sum_{j=1}^n a_{ij} X_j}$;
  end for
until $\frac{\sum_{i=1}^n \|M_i - X_i Y_i\|_F^2}{\sum_{i=1}^n \|M_i\|_F^2} \le \epsilon$

Output: Factors $\{X_i\}_{i=1}^n, \{Y_i\}_{i=1}^n$

$X_j, \forall i, j \in \{1, \cdots, n\}$. Using this consensus constraint, similarly we can design a distributed algorithm

$$(X_i)_{st} \leftarrow (X_i)_{st} \frac{\left(M_i Y_i^\top + \lambda(A_{i\cdot} \otimes I_p)\mathcal{X}\right)_{st}}{\left(X_i Y_i Y_i^\top + \lambda(D_{i\cdot} \otimes I_p)\mathcal{X}\right)_{st}}, \tag{15}$$

$$(Y_i)_{st} \leftarrow (Y_i)_{st} \frac{\left(X_i^\top M_i\right)_{st}}{\left(X_i^\top X_i Y_i\right)_{st}}, \tag{16}$$

$$\lambda \leftarrow \lambda + \alpha \left\langle \mathcal{X}, L\mathcal{X} \right\rangle_F, \tag{17}$$

where λ is the Lagrange multiplier for the constraint $\left\langle \mathcal{X}, L\mathcal{X} \right\rangle_F = 0$ and α is a small positive step-size. On the other hand, if we put $\left\langle \mathcal{X}, L\mathcal{X} \right\rangle_F$ into the objective function as the penalty function with a fixed penalty parameter λ, then the problem turns out to be the Graph Regularized NMF [2].

4. MAIN RESULTS

In this section, we provide convergence analysis and computational complexity analysis for the proposed DisNMF algorithm.

4.1. Convergence analysis

Since the non-negative constraints will be satisfied by multiplicative update rules, we consider the problem (8) only with equality constraints, that is

$$\begin{aligned} \min_{X_i, Y_i} \quad & F(\mathcal{X}, \mathcal{Y}) = \frac{1}{2} \sum_{i=1}^n \|M_i - X_i Y_i\|_F^2 \\ \text{s.t.} \quad & \sum_{j=1}^n a_{ij}(X_i - X_j) = 0, \qquad i \in \{1, \cdots, n\}. \end{aligned} \tag{18}$$

Still, let Λ_is be the Lagrange multipliers for the constraints $\sum_{j=1}^{n} a_{ij}(X_i - X_j) = 0$, $i \in \{1, \cdots, n\}$. Then, the Lagrange function $\mathcal{L}$ of (18) is

$$\mathcal{L}(\mathcal{X}, \mathcal{Y}, \Lambda) = \frac{1}{2} \sum_{i=1}^{n} (\|M_i - X_i Y_i\|_F^2 + \langle \Lambda_i, \sum_{j=1}^{n} a_{ij}(X_i - X_j) \rangle_F) \,. \tag{19}$$

It is easy to verify that the solution of the problem (18) coincides with that of the following problem

$$\min_{\mathcal{X}, \mathcal{Y}} \max_{\Lambda} \mathcal{L}(\mathcal{X}, \mathcal{Y}, \Lambda) \tag{20}$$

Inspired by the primal-dual method, the algorithm is designed as

$$\mathcal{X}^{k+1}, \mathcal{Y}^{k+1} \leftarrow \underset{\mathcal{X}, \mathcal{Y}}{\operatorname{argmin}} \mathcal{L}\left(\mathcal{X}, \mathcal{Y}, \Lambda | \mathcal{X}^k, \mathcal{Y}^k, \Lambda^k\right) \tag{21}$$

$$(\Lambda_i)_{st}^{k+1} \leftarrow (\Lambda_i)_{st}^{k} + (\eta_i)_{st}^{k} \frac{\partial \mathcal{L}}{\partial (\Lambda_i)_{st}^{k}} \tag{22}$$

where $(\eta_i)_{st}^{k}$ is a step-size, $\mathcal{X}^k, \mathcal{Y}^k, \Lambda^k$ denote the variables $\mathcal{X}, \mathcal{Y}, \Lambda$ at iteration k. The update of dual variable (22) is easy to carry out, just as the (13) and (14) showed. Given the initial value $(\Lambda_i)_{st}^{0} \geq 0$, Λ_i^k will be non-negative matrices all the time. The difficulty lies in dealing with (21), since $\mathcal{L}$ is non-convex. We will solve this by constructing auxiliary functions.

For any element $(X_i)_{st}$ in $\mathcal{L}$, let $\mathcal{L}_{(X_i)_{st}}$ denotes the part of $\mathcal{L}$ relevant to $(X_i)_{st}$. Let $\mathcal{L}'_{(X_i)_{st}}$ and $\mathcal{L}''_{(X_i)_{st}}$ denote the first and second order partial derivative of $\mathcal{L}$ with respect to $(X_i)_{st}$, that is,

$$\mathcal{L}'_{(X_i)_{st}} = \left(\frac{\partial \mathcal{L}}{\partial X_i}\right)_{st} = \left(-M_i Y_i^\top + X_i Y_i Y_i^\top + (L_{i\cdot} \otimes I_p)\Lambda\right)_{st} \,, \tag{23}$$

$$\mathcal{L}''_{(X_i)_{st}} = (Y_i Y_i^\top)_{tt} \,. \tag{24}$$

The Taylor series expansion of $\mathcal{L}_{(X_i)_{st}}(x)$ at the point $x = (X_i)_{st}^{k}$ is

$$\mathcal{L}_{(X_i)_{st}}(x) = \mathcal{L}_{(X_i)_{st}}\left((X_i)_{st}^{k}\right) + \mathcal{L}'_{(X_i)_{st}}\left((X_i)_{st}^{k}\right)\left(x - (X_i)_{st}^{k}\right) + \frac{1}{2}\mathcal{L}''_{(X_i)_{st}}\left(x - (X_i)_{st}^{k}\right)^2 \,. \tag{25}$$

Since the update is essentially element-wise, it is sufficient to show that each $\mathcal{L}_{(X_i)_{st}}$ is non-increasing under the update step of (11). We prove this by defining the auxiliary function regarding $(X_i)_{st}$ as follows.

Lemma 4.1. The function

$$\begin{aligned} G(x, (X_i)_{st}^{k}) = {} & \mathcal{L}_{(X_i)_{st}}\left((X_i)_{st}^{k}\right) + \mathcal{L}'_{(X_i)_{st}}\left((X_i)_{st}^{k}\right)\left(x - (X_i)_{st}^{k}\right) \\ & + \frac{\left(X_i Y_i Y_i^\top + D_{i\cdot}\Lambda\right)_{st}}{2(X_i)_{st}}\left(x - (X_i)_{st}^{k}\right)^2 \end{aligned} \tag{26}$$

is an auxiliary function for $\mathcal{L}_{(X_i)_{st}}(x)$.

Proof. Obviously, $G((X_i)_{st}^k, (X_i)_{st}^k) = \mathcal{L}_{(X_i)_{st}}((X_i)_{st}^k)$. According to the definition of auxiliary function, we only need to show that $G(x, (X_i)_{st}^k) \geq \mathcal{L}_{(X_i)_{st}}(x)$. Put (24) into (25) and compare it with (26), it is equivalent to show

$$\frac{\left(X_i Y_i Y_i^\top + (D_{i\cdot} \otimes I_p)\Lambda\right)_{st}}{2(X_i)_{st}} \geq \frac{1}{2}\mathcal{L}''_{(X_i)_{st}} = \frac{1}{2}(Y_i Y_i^\top)_{tt} .$$

Since D and Λ are non-negative matrices, $\left((D_{i\cdot} \otimes I_p)\Lambda\right)_{st} \geq 0$. To prove the above inequality, we have

$$(X_i Y_i Y_i^\top)_{st} = \sum_{l=1}^{d} (X_i)_{sl}(Y_i Y_i^\top)_{lt} \geq (X_i)_{st}(Y_i Y_i^\top)_{tt} .$$

□

Then we define auxiliary functions for the update rule (12). Similarly, let $\mathcal{L}_{(Y_i)_{st}}$ denotes the part of $\mathcal{L}$ relevant to $(Y_i)_{st}$. Then we have

Lemma 4.2. The function

$$\begin{aligned} G(y, (Y_i)_{st}^k) = {} & \mathcal{L}_{(Y_i)_{st}}\left((Y_i)_{st}^k\right) + \mathcal{L}'_{(Y_i)_{st}}\left((Y_i)_{st}^k\right)\left(y - (Y_i)_{st}^k\right) \\ & + \frac{(X_i^\top X_i Y_i)_{st}}{2(Y_i)_{st}}\left(y - (Y_i)_{st}^k\right)^2 \end{aligned} \tag{27}$$

is an auxiliary function for $\mathcal{L}_{(Y_i)_{st}}(y)$.

The proof of Lemma 4.2 is essentially similar to that of Lemma 4.1. The key point is

$$\frac{(X_i^\top X_i Y_i)_{st}}{2(Y_i)_{st}} \geq \frac{1}{2}\mathcal{L}''_{(Y_i)_{st}} = \frac{1}{2}(X_i^\top X_i)_{ss} ,$$

since

$$(X_i^\top X_i Y_i)_{st} = \sum_{l=1}^{d} (X_i^\top X_i)_{sl}(Y_i)_{lt} \geq (X_i^\top X_i)_{ss}(Y_i)_{st} .$$

The complete proof is omitted here due to the space limitation.

Then, we have the following lemma regarding the iterative updating rules (11), (12) and (14).

Lemma 4.3. The Lagrange function $\mathcal{L}$ is non-increasing under the update rules (11) and (12), and $\mathcal{L}$ is non-decreasing under the update rules (14). $\mathcal{L}$ is invariant if and only if $\mathcal{X}$, $\mathcal{Y}$ and Λ are at a stationary point.

Proof. Putting $G(x, (X_i)_{st}^k)$ of (26) and $G(y, (Y_i)_{st}^k)$ of (27) into (2), we have:

$$(X_i)_{st}^{k+1} = \arg\min_x G\left(x, (X_i)_{st}^k\right) , \tag{28}$$

$$(Y_i)_{st}^{k+1} = \arg\min_y G\left(y, (Y_i)_{st}^k\right) . \tag{29}$$

To solve above argmin problems, let

$$G'(x,(X_i)_{st}^k) = \frac{\left(X_iY_iY_i^\top + (D_{i\cdot}\otimes I_p)\Lambda\right)_{st}}{(X_i)_{st}}\left(x-(X_i)_{st}^k\right) + \mathcal{L}'_{(X_i)_{st}}\left((X_i)_{st}^k\right) = 0\,, \quad (30)$$

$$G'(y,(Y_i)_{st}^k) = \frac{\left(X_i^\top X_iY_i\right)_{st}}{(Y_i)_{st}}\left(x-(X_i)_{st}^k\right) + \mathcal{L}'_{(Y_i)_{st}}\left((Y_i)_{st}^k\right) = 0\,. \quad (31)$$

We get the close form update rules of $(X_i)_{st}$ and $(Y_i)_{st}$:

$$(X_i)_{st}^{k+1} = (X_i)_{st}^k - \frac{(X_i)_{st}^k\cdot\mathcal{L}'_{(X_i)_{st}}\left((X_i)_{st}^k\right)}{\left(X_iY_iY_i^\top + (D_{i\cdot}\otimes I_p)\Lambda\right)_{st}^k} = (X_i)_{st}^k\frac{(M_iY_i^\top + (A_{i\cdot}\otimes I_p)\Lambda)_{st}^k}{\left(X_iY_iY_i^\top + (D_{i\cdot}\otimes I_p)\Lambda\right)_{st}^k}\,, \quad (32)$$

$$(Y_i)_{st}^{k+1} = (Y_i)_{st}^k - \frac{(Y_i)_{st}^k\cdot\mathcal{L}'_{(Y_i)_{st}}\left((Y_i)_{st}^k\right)}{\left(X_i^\top X_iY_i\right)_{st}^k} = (Y_i)_{st}^k\frac{(X_i^\top M_i)_{st}^k}{\left(X_i^\top X_iY_i\right)_{st}^k}\,. \quad (33)$$

Lemma 4.1 and Lemma 4.2 state that (26) and (27) are auxiliary functions, so $\mathcal{L}_{(X_i)_{st}}(x)$, $\mathcal{L}_{(Y_i)_{st}}(y)$ are non-increasing under update rules (11) and (12), according to Lemma 2.5.

On the other hand, we calculate the change of $\mathcal{L}$ with the update of Λ at iteration k:

$$\begin{aligned}\mathcal{L}(\mathcal{X}^k,\mathcal{Y}^k,\Lambda^{k+1}) - \mathcal{L}(\mathcal{X}^k,\mathcal{Y}^k,\Lambda^k) &= \sum_{i=1}^n\langle(L_{i\cdot}\otimes I_p)\mathcal{X}^k,\Lambda^{k+1}-\Lambda^k\rangle_F\\ &= \sum_{i=1}^n\sum_{s,t}(\eta_i)_{st}^k\left(((L_{i\cdot}\otimes I_p)\mathcal{X})_{st}^k\right)^2 \geq 0\,. \end{aligned} \quad (34)$$

Then $\mathcal{L}$ is non-decreasing under the update rules (14). □

Theorem 4.4. Under Assumption 3.1, the sequence $(\mathcal{X}^k,\mathcal{Y}^k)$ generated by the update rules (11), (12) and (14) will converge to a local minimum of the objective function F.

Proof. Lemma 4.3 indecates that the update rules (11), (12) and (14) stop after they reach a Saddle Point of $\mathcal{L}$, denoted as $(\mathcal{X}^*,\mathcal{Y}^*,\Lambda^*)$. Since F is a continuously differentiable function and is marginally convex in both its variables, its bistable points are exactly its stationary points. Then $(\mathcal{X}^*,\mathcal{Y}^*)$ is a bistable point as well as a local minimum of the objective function F. □

Moreover, for non-convex functions, there is no assurance that all bistable points are global minima. The bistable point to which DisNMF eventually converges to depends on where the procedure was initialized. In order to converge to the globally optimal point, DisNMF should be initialized inside the region of attraction of the global optimum.

In practice, one often use $F(\mathcal{X}^k,\mathcal{Y}^k)/\|M\|_F^2 \leq \epsilon$ as the algorithm stopping criterion. In other works, no matter $(\mathcal{X}^k,\mathcal{Y}^k)$ converges to the globally optimal point or not, we do get a non-negative factorization of M within an acceptable approximation error.

Remark 4.5. Without additional constraints, one can not ensure the uniqueness of the factorization, since, for any diagonal matrix D with its diagonal entries positive, and any permutation matrix P, we have

$$X_iY_i = (X_iPD)((PD)^{-1}Y_i)\,,$$

which allows us to produce many feasible solutions. To put it simply, once we get a factorization $(\mathcal{X},\mathcal{Y})$ of M, a set of solutions $(a\mathcal{X}, a^{-1}\mathcal{Y})$ are also factorizations of M, for any $a \in \mathbb{R}_+$. In this sense, we can properly scale the X_i^k, so that the sequence X_i^k can have a bound, even $\|X_i^1\|_F = \|X_i^2\|_F = \cdots = \|X_i^k\|_F$. Further study of the uniqueness of NMF can go to the reference [7].

4.2. Computational complexity analysis

Here we discuss the computational complexity of our DisNMF algorithm in comparison with the standard NMF algorithm (MUR). We count the numbers of arithmetic operations in Table 1.

	addition	multiplication	division	overall
NMF: X	$dp(q-1)+d^2(q-1)+d(d-1)p$	$dpq+d^2p+d^2q+dq$	dp	$\mathcal{O}(dpq)$
NMF: Y	$d(p-1)q+d^2(p-1)+d(d-1)q$	$dpq+d^2p+d^2q+dq$	dq	$\mathcal{O}(dpq)$
DisNMF: X_i	$dpq_i+d^2p+d^2q_i-d^2+(n-1)dp$	$dpq_i+d^2p+d^2q_i+2dp+ndp$	dp	$\mathcal{O}(dpq_i+ndp)$
DisNMF: Y_i	$d(p-1)q_i+d^2(p-1)+d(d-1)q_i$	$dpq_i+d^2p+d^2q_i+dq_i$	dq_i	$\mathcal{O}(dpq_i)$
DisNMF: Λ_i	$(n-1)dp$	$ndp+2dp$	dp	$\mathcal{O}(ndp)$

Tab. 1. Computational operation counts for each iteration in DisNMF.

It shows that the cost for standard NMF's update in each iteration is $\mathcal{O}(dpq)$. As for DisNMF, in each iteration every agent should update its X_i, Y_i and Λ_i, so the cost for DisNMF's update is

$$\begin{aligned}&\sum_{i=1}^{n}\left(\mathcal{O}(dpq_i+ndp)+\mathcal{O}(dpq_i)+\mathcal{O}(ndp)\right)\\ =&\mathcal{O}(dp\sum_{i=1}^{n}q_i)+\sum_{i=1}^{n}\mathcal{O}(ndp)=\mathcal{O}(dpq+n^2dp)\approx\mathcal{O}(dpq)\,,\end{aligned}\tag{35}$$

since $\sum_{i=1}^{n} q_i = q$ and the agent number $n \ll q$ in practice. Computational complexity analysis shows that our DisNMF is a linear extension of NMF.

5. NUMERICAL EXAMPLE

This section gives numerical examples for illustration. By applying DisNMF algorithm to a small matrix and a big matrix respectively, we show the good performance of DisNMF and make comparison tests.

5.1. A small matrix example

Consider the following 6×8-dimension matrix:

$$M = \begin{bmatrix} 24 & 34 & 35 & 61 & 49 & 34 & 21 & 21 \\ 19 & 23 & 26 & 36 & 23 & 25 & 7 & 18 \\ 36 & 38 & 47 & 58 & 38 & 40 & 9 & 34 \\ 18 & 26 & 26 & 46 & 36 & 26 & 16 & 15 \\ 27 & 29 & 35 & 43 & 26 & 31 & 6 & 25 \\ 15 & 21 & 23 & 34 & 21 & 25 & 6 & 17 \end{bmatrix}.$$

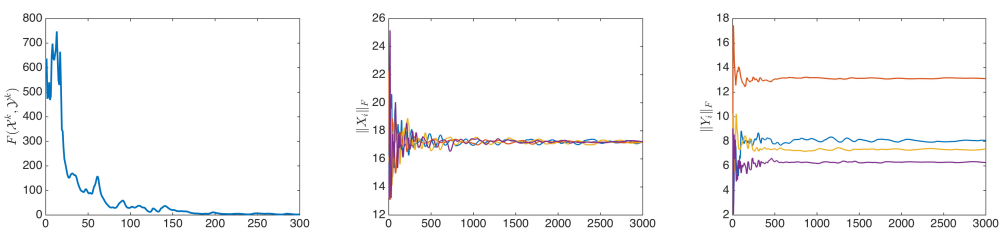
Assume that there are four agents, and each agent only has access to two columns of M. Moreover, the adjacency matrix of the multi-agent network is

$$A = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}.$$

Set latent dimension $d = 4$ and iteration $T = 3000$, we run DisNMF algorithm to factorize M and get the factors $\{X_1, \cdots, X_4\}, \{Y_1, \cdots, Y_4\}$. To see the approximation of the factorization, we multiply factors back, that is $\{X_1Y_1, \cdots, X_4Y_4\}$:

$$\overline{M} = \begin{bmatrix} 24.1700 & 34.1632 & 35.1438 & 60.8913 & 48.8155 & 34.1312 & 21.1868 & 20.6344 \\ 18.8522 & 22.7751 & 25.8614 & 36.0496 & 23.2838 & 24.9087 & 6.7278 & 18.3903 \\ 36.2039 & 38.2503 & 47.1519 & 57.8863 & 37.7320 & 40.1719 & 9.2535 & 33.5023 \\ 17.7934 & 25.7907 & 25.8521 & 46.1318 & 36.2218 & 25.8207 & 15.7747 & 15.4537 \\ 26.7432 & 28.7257 & 34.8197 & 43.1314 & 26.2742 & 30.8090 & 5.7770 & 25.5990 \\ 15.1219 & 21.1537 & 23.0638 & 33.9900 & 20.8694 & 25.0560 & 6.0609 & 16.6820 \end{bmatrix}.$$

$F(\mathcal{X}^T, \mathcal{Y}^T)/\|M\|_F^2$ actually quantifies the quality of the approximation. After 3000 iteration, $F(\mathcal{X}^{3000}, \mathcal{Y}^{3000}) = 2.6132$ and $F(\mathcal{X}^{3000}, \mathcal{Y}^{3000})/\|M\|_F^2 = 5.8034 \times 10^{-5}$. In addition, we draw the trajectories of $F, \|X_i\|_F, \|Y_i\|_F$ versus iteration, as shown in Figure 2.



(a) $F(\mathcal{X}, \mathcal{Y})$ (b) $\|X_i\|_F$ of 4 agents (c) $\|Y_i\|_F$ of 4 agents

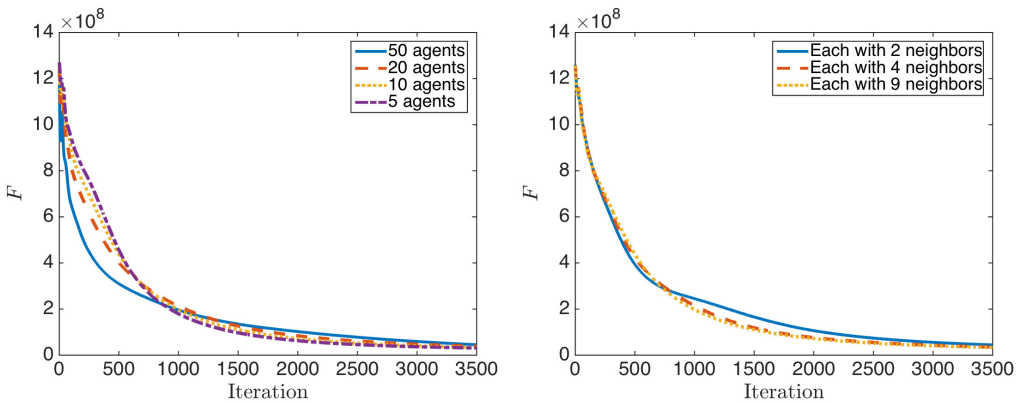
Fig. 2. Trajectory of $F(\mathcal{X}, \mathcal{Y}), \|X_i\|_F, \|Y_i\|_F$ versus iteration.

Figure 2(a) shows that the solutions are feasible because $\lim_{k\to\infty}\sum_{i=1}^{n}\|M_i - X_i^k Y_i^k\|_F^2 \to 0$; Figure 2(b) shows that X_i of each agent will come to consensus and Figure 2(c) shows that Y_i of each agent will converge respectively.

5.2. Comparison tests

It is time to consider a pretty big low-rank matrix $M \in \mathbb{R}^{1000\times 1000}$, which is randomly generated with $\text{rank}(M) = 100$, and we set the latent dimension $d = 100$.

Since our DisNMF algorithm is designed over multi-agent networks, its performance can by affected by the number of agents and the structure of the communication network. Firstly, we consider four kinds of multi-agent networks, which consist of 5 agents, 10 agents, 20 agents and 50 agents, respectively; the structure of the networks are all completely connected graph, that is, agents in the network can communicate with each other. The performance of DisNMF with different agent numbers can be seen in Figure 3(a).

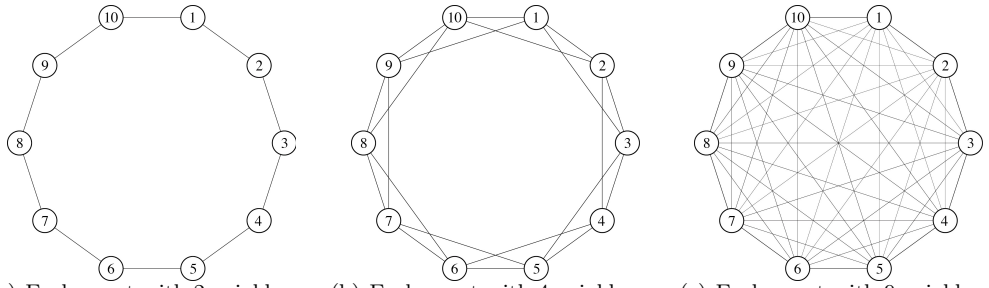


(a) Different agent numbers (b) Different network structures

Fig. 3. The performance of DisNMF with different multi-agent networks.

Then we look into the structure of the communication network. We consider three kinds of multi-agent networks, which all consist of 10 agents. In the first network, each agent can communicate with its nearest 2 neighbors; in the second network, each agent can communicate with its nearest 4 neighbors and in the third network, each agent can communicate with 9 neighbors, that is, they can communicate with each other. The communication graphs are shown in Figure 4, and the performance of DisNMF with different network structures can be seen in Figure 3(b).

Last but not the least, we compare our distributed NMF algorithm with the standard centralized NMF algorithm (MUR)[11]. We adopt a 10-agent completely connected network. After 5000 iterations, our DisNMF algorithm achieves $F(\mathcal{X}^{5000}, \mathcal{Y}^{5000})/\|M\|_F^2 = 1.175 \times 10^{-5}$. Figure 5 indicates that although matrices are distributed stored among agents, DisNMF is comparable with centralized NMF algorithm.



(a) Each agent with 2 neighbors (b) Each agent with 4 neighbors (c) Each agent with 9 neighbors

Fig. 4. Three kinds of network structures.

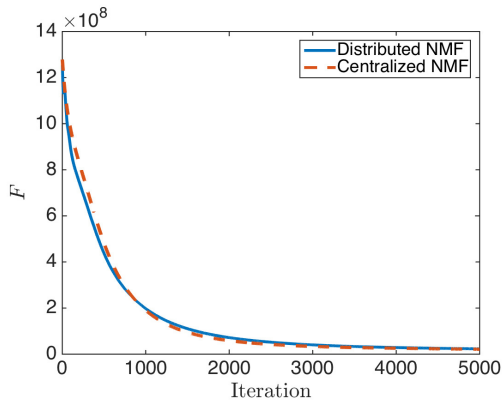


Fig. 5. Comparative result for our distributed NMF algorithm with centralized NMF algorithm.

6. CONCLUSION

In this paper, a distributed algorithm for non-negative matrix factorization over a multi-agent network has been studied. The convergence analysis has been provided by using auxiliary functions, and the algorithm performance has been illustrated via numerical examples. The proposed algorithm can be widely used in many applications, such as joint NMF problem, multi-view clustering problem and distributed matrix completion problem, with minor modification in the update rules.

ACKNOWLEDGEMENT

This work is supported by NSFC (61733018) and the National Key Research and Development Program (2016YFB0901900).

(Received August 5, 2020)

REFERENCES

[1] D. P. Bertsekas and J. W. Tsitsiklis: Parallel and Distributed Computation: Numerical Methods. Prentice hall Englewood Cliffs, NJ 1989. DOI:10.1109/TPAMI.2010.231

[2] D. Cai, X. He, J. Han, and T. S. Huang: Graph regularized nonnegative matrix factorization for data representation. IEEE Trans. Pattern Analysis Machine Intell. *33* (2010), 1548–1560. DOI:10.1109/TPAMI.2010.231

[3] W. Deng, X. Zeng, and Y. Hong: Distributed computation for solving the sylvester equation based on optimization. IEEE Control Systems Lett. *4* (2019), 414–419. DOI:10.1109/LCSYS.2019.2942711

[4] Ch. Godsil and G. F. Royle: Algebraic graph theory. Springer Science Business Media *207* (2013).

[5] X. He, M.-Y. Kan, P. Xie, and X. Chen: Comment-based multi-view clustering of web 2.0 items. In: Proc. 23rd International Conference on World wide web, 2014, pp. 771–782. DOI:10.1145/2566486.2567975

[6] R. Horst and H. Tuy: Global Optimization. Springer–Verlag, Berlin 1996. DOI:10.1007/978-3-662-03199-5

[7] K. Huang, N. D. Sidiropoulos, and A. Swami: Non-negative matrix factorization revisited: Uniqueness and algorithm for symmetric decomposition. IEEE Trans. Signal Process. *62* (2013), 211–224. DOI:10.1109/TSP.2013.2285514

[8] P. Jain and P. Kar: Non-convex optimization for machine learning. arXiv preprint arXiv:1712.07897, 2017. DOI:10.1561/9781680833690

[9] H. Kim and H. Park: Nonnegative matrix factorization based on alternating nonnegativity constrained least squares and active set method. SIAM J. Matrix Analysis Appl. *30* (2008), 713–730. DOI:10.1137/07069239X

[10] D. L. Lee and H. S. Seung: Learning the parts of objects by non-negative matrix factorization. Nature *401* (1999), 788–791. DOI:10.1038/44565

[11] D. S. Lee and H. S. Seung: Algorithms for non-negative matrix factorization. Adv. Neural Inform. Process. Systems *xx* (2001), 556–562.

[12] W. Li, X. Zeng, Y. Hong, and H. Ji: Distributed Design for nuclear norm minimization of linear matrix equation with constraints. IEEE Trans. Automat. Control(2020). DOI:10.1109/TAC.2020.2981930

[13] Ch.-J. Lin: Projected gradient methods for nonnegative matrix factorization. Neural Comput. *19* (2007), 2756–2779. DOI:10.1162/neco.2007.19.10.2756

[14] Ch. Liu, H.-Ch. Yang, J. Fan, L.-W. He, and Y.-M. Wang: Distributed nonnegative matrix factorization for web-scale dyadic data analysis on mapreduce. In: Proc. 19th International Conference on World wide web (2010), pp. 681–690. DOI:10.1145/1772690.1772760

[15] J. Liu, Ch. Wang, J. Gao, and J. Han: Multi-view clustering via joint nonnegative matrix factorization. In: Proc. 2013 SIAM International Conference on Data Mining, SIAM 2013, pp. 252–260. DOI:10.1137/1.9781611972832.28

[16] A. Nedic, A, Ozdaglar, and P. A. Parrilo: Constrained consensus and optimization in multi-agent networks. IEEE Trans. Automat. Control *55* (2010), 922–938. DOI:10.1109/TAC.2010.2041686

[17] V. Peterka: Bayesian system identification. In: Trends and Progress in System Identification (P. Eykhoff, ed.), Pergamon Press, Oxford 1981, pp. 239–304. DOI:10.1016/B978-0-08-025683-2.50013-2

[18] Z. Qiu, S. Liu, and L. Xie: Distributed constrained optimal consensus of multi-agent systems. Automatica *68* (2016), 209–215. DOI:10.1016/j.automatica.2016.01.055

[19] S. S. Ram, A. Nedić, and V. V. Veeravalli: Distributed stochastic subgradient projection algorithms for convex optimization. J. Optim. Theory Appl. *147* (2010), 516–545. DOI:10.1007/s10957-010-9737-7

[20] G. Shi, B. D. O. Anderson, and U. Helmke: Network flows that solve linear equations. IEEE Trans. Automat. Control *62* (2017), 2659–2674. DOI:10.1109/TAC.2016.2612819

[21] Z. Wen, W. Yin, and Y. Zhang: Solving a low-rank factorization model for matrix completion by a nonlinear successive over-relaxation algorithm. Math. Programm. Comput. *4* (2012), 333–361. DOI:10.1007/s12532-012-0044-1

[22] F. Xu and G. He: New algorithms for nonnegative matrix completion. Pacific J. Optim. *11* (2015), 459–469.

[23] S. Yang, Q. Liu, and J. Wang: A multi-agent system with a proportional-integral protocol for distributed constrained optimization. IEEE Trans. Automat. Control *62* (2016), 3461–3467. DOI:10.1109/TAC.2016.2610945

[24] P. Yi, Y. Hong, and F. Liu: Distributed gradient algorithm for constrained optimization with application to load sharing in power systems. Systems Control Lett. *83* (2015), 45–52. DOI:10.1016/j.sysconle.2015.06.006

[25] J. Yin, L. Gao, and Z. M. Zhang: Scalable nonnegative matrix factorization with block-wise updates. In: Joint European Conference on Machine Learning and Knowledge Discovery in Databases, Springer–Heidelberg, Berlin 2014, pp. 337–352. DOI:10.1007/978-3-662-44845-8_22

[26] D. Yuan, D. W. C. Ho, and S. Xu: Regularized primal–dual subgradient method for distributed constrained optimization. IEEE Trans. Cybernet. *46* (2015), 2109–2118. DOI:10.1109/TCYB.2015.2464255

[27] X. Zeng and K. Cao: Computation of linear algebraic equations with solvability verification over multi-agent networks. Kybernetika *53* (2017), 803–819. DOI:10.14736/kyb-2017-5-0803

[28] X. Zeng, P. Yi, and Y. Hong: Distributed continuous-time algorithm for constrained convex optimizations via nonsmooth analysis approach. IEEE Trans. Automat. Control *62* (2016), 5227–5233. DOI:10.1109/TAC.2016.2628807

[29] X. Zeng, S. Liang, Y. Hong, and J. Chen: Distributed computation of linear matrix equations: An optimization perspective. IEEE Trans. Automat. Control *64* (2019), 1858–1873. DOI:10.1109/TAC.2018.2847603

Zhipeng Tu, Key Lab of Systems and Control, Academy of Mathematics and Systems Science, University of Chinese Academy of Sciences, Beijing 100190. P. R. China.
e-mail: tuzhipeng@amss.ac.cn

Weijian Li, Department of Automation, University of Science and Technology of China, Hefei 230027. P. R. China.
e-mail: ustcwjli@mail.ustc.edu.cn