Decentralized and Privacy-Preserving Low-Rank Matrix Completion

An-Ya Lin & Qing Ling

Journal of the Operations Research Society of China

ISSN 2194-668X
Volume 3
Number 2

J. Oper. Res. Soc. China (2015) 3:189-205
DOI 10.1007/s40305-015-0080-4

Springer

J. Oper. Res. Soc. China (2015) 3:189–205
DOI 10.1007/s40305-015-0080-4

Decentralized and Privacy-Preserving Low-Rank Matrix Completion

An-Ya Lin[1] · **Qing Ling**[1]

Received: 11 November 2014 / Revised: 6 February 2015 / Accepted: 6 February 2015 /
Published online: 5 May 2015

Abstract In this paper, we propose a decentralized algorithm to solve the low-rank matrix completion problem and analyze its privacy-preserving property. Suppose that we want to recover a low-rank matrix $D = [D_1, D_2, \cdots, D_L]$ from a subset of its entries. In a network composed of L agents, each agent i observes some entries of D_i. We factorize the unknown matrix D as the product of a public matrix X which is common to all agents and a private matrix $Y = [Y_1, Y_2, \cdots, Y_L]$ of which Y_i is held by agent i only. Each agent i updates Y_i and its local estimate of X, denoted by $X_{(i)}$, in an alternating manner. Through exchanging information with neighbors, all the agents move toward a consensus on the estimates $X_{(i)}$. Once the consensus is (nearly) reached throughout the network, each agent i recovers $D_i = X_{(i)}Y_i$, thus D is recovered. In this progress, communication through the network may disclose sensitive information about the data matrices D_i to a malicious agent. We prove that in the proposed algorithm, D-LMaFit, if the network topology is well designed, the malicious agent is unable to reconstruct the sensitive information from others.

Keywords Decentralized algorithm · Matrix completion · Privacy-preserving

Mathematics Subject Classification 49R99 · 90C90

1 Introduction

Completing a low-rank (or approximately low-rank) matrix from an incomplete set of its entries has been a hot research topic in recent years [5,6,25]. This problemarises

✉ Qing Ling
qingling@mail.ustc.edu.cn

An-Ya Lin
linanya@mail.ustc.edu.cn

[1] Department of Automation, University of Science and Technology of China, Hefei, China

in various applications, such as collaborative filtering [1], system identification [19], internet traffic analysis [36], sensor localization [22], video processing [17], and phase retrieval [7], to name a few.

This paper considers *decentralized matrix completion*, in which a network of agents collaborate to complete a low-rank matrix that is the collection of multiple local data matrices. To be specific, each agent observes some entries of its own local data matrix and exchanges information with its neighbors to complete the unobserved ones [18,21]. Particularly, we focus on designing a *privacy-preserving* decentralized matrix completion algorithm such that a malicious agent in the network is unable to recover local data matrices of other agents through neighboring information exchange. This privacy-preserving property is critical to protecting sensitive data that come from multiple sources (for example, medical data of hospitals, selling data of merchants), yet utilizing the data to jointly accomplish a matrix completion task.

1.1 Our Contributions

This paper develops a decentralized matrix completion algorithm D-LMaFit and analyzes its privacy-preserving property.

D-LMaFit is motivated by LMaFit, a centralized nonconvex matrix factorization approach [32]. In D-LMaFit, each agent factorizes its local data matrix, which is to be completed, to the product of a *public matrix* and a *private matrix*, and updates them in an alternating fashion. The private matrix is kept locally, while the public matrix is shared with its neighbors. The public matrices, which are computed from the private matrices, are expected to reach a consensus eventually. We propose a dynamic average consensus scheme to achieve such a consensus, which is, according to numerical experiments, close to the value of the public matrix in the centralized case.

We establish the privacy-preserving property of D-LMaFit, in the sense that a malicious agent is unable to recover the local data matrices of some interested agents through observing the exchanged public matrices from its neighbors. With particular note, strict privacy preservation is hard to prove, as the malicious agent may be able to recover the local data matrices by utilizing their special structures. We prove weak privacy preservation, in the sense that the malicious agent is unable to invert a dynamic linear system whose inputs are the private matrices of the interested agents and outputs are the public matrices of the malicious agent and its neighbors. The analysis shows that privacy of an agent cannot be preserved, if and only if the agent and its neighbors belong to a set that contains the malicious agent and its neighbors. This theoretical result provides a guideline to adjust network topology for the sake of privacy preservation.

1.2 Related Work

Some existing algorithms solve the matrix completion problem based on nuclear-norm minimization, which is a convex program and tends to yield a low-rank matrix [4,12,20]. In light of the fact that the matrix to complete can be (approximately) decomposed as the product of two lower dimensional matrices, [15,32] consider a matrix factorization formulation that is nonconvex. The convex approach guarantees

global convergence to the optimal solution but needs singular value decompositions, which are expensive on large-scale matrices. The nonconvex approach is subject to the existence of local minima, though [15] also guarantees global convergence to the optimal solution given that the initial iterate is good enough. The nonconvex approach often has satisfactory recovery performance, and the computation cost is lower than that of the convex approach as it avoids singular value decompositions [32].

Most of the existing matrix completion algorithms are *centralized*, i.e., a fusion center collects the observed entries of the data matrix and recovers the rest. However, network applications prefer *decentralized computing*, in which a network of geographically distributed agents separately collect data and collaboratively accomplish an optimization task [13,29]. The decentralized computing scheme is advantageous over the centralized one as the agents do not need to transmit raw data to the fusion center, which often relies on costly multi-hop communication. More importantly, decentralized computing helps preserve data privacy since an agent no longer shares its raw data with either the fusion center or other agents. For matrix completion, each agent completes its own data matrix locally, only with the aid of necessary information exchange with its neighbors. This property is particularly of interest in collaborative filtering of sensitive data, such as medical or economic data.

Traditional network computing relies on cryptographical tools to preserve data privacy, including multi-party computation [14,28], randomization [9], and data aggregation [27]. The cryptographical techniques are well studied but expensive and fail to utilize the problem structures in network optimization. The privacy-preserving property of D-LMaFit is similar to that of the decentralized autonomous online learning (DAOL) algorithm [35]. In DAOL, a network of decentralized agents collaboratively performs online learning through exchanging neighboring iterates and descending on instantaneous local cost functions. The agents expect to prevent leaking local gradients, which may be sensitive, to others, through exchanging neighboring iterates. The privacy-preserving abilities of D-LMaFit and DAOL are not cryptographical but topology-dependent.

Notations Throughout the paper, for a matrix D, denote $\text{rank}(D)$ as its rank, $||D||_*$ as its nuclear norm, and $||D||_{\text{F}}$ as its Frobenius norm.

2 Problem Statement

Consider a bidirectionally connected network composed of L agents. There is an undirected edge between two agents if they can communicate with each other through one hop. Represent the network by $(\mathcal{V}, \mathcal{E})$ where $\mathcal{V}$ is the set of agents and $\mathcal{E}$ is the set of undirected edges. For agent i, denote the set of its neighbors by $\mathcal{N}_i \triangleq \{j|(i, j) \in \mathcal{E}\}$. For a group of agents $\mathcal{I}$, denote the set of their neighbors by $\mathcal{N}_{\mathcal{I}} \triangleq \{j|i \in \mathcal{I}, j \notin \mathcal{I}, (i, j) \in \mathcal{E}\}$.

The goal of the network is to collaboratively complete a low-rank matrix in a decentralized fashion. To be specific, agent i observes some entries from a local data matrix $D_i \in \mathbb{R}^{N\times P_i}$, and the set of observed coordinates is Ω_i. The whole data matrix is $D = [D_1, D_2, \cdots, D_L] \in \mathbb{R}^{N\times P}$, where $P = \sum_{i=1}^{L} P_i$. Denoting d_{np} as the entry at

the nth row and the pth column of D and $\Omega \triangleq \bigcup_{i=1}^{L} \Omega_i$ as the collection of observed coordinates, d_{np} is known for $(n, p) \in \Omega \subset \{(n, p) : 1 \leqslant n \leqslant N, 1 \leqslant p \leqslant P\}$; otherwise d_{np} needs completion for $(n, p) \notin \Omega$. The data matrix D is with rank no more than K or can be well approximated by a matrix with rank no more than K, where $K \ll \min(N, P)$.

The matrix completion task is decentralized in the sense that there is no central controller to collect all local observations $\{d_{np}|(n, p) \in \Omega\}$ and perform computation in a centralized manner. Instead, we develop a decentralized matrix completion algorithm, in which agents communicate and collaborate with their neighbors to accomplish the task. This decentralized computation scheme helps preserve data privacy. We expect that each agent i is able to protect its local data matrix D_i, whose entries are either known (those in Ω_i) or to be estimated (those not in Ω_i), from a malicious agent in the network.

3 Centralized Matrix Completion

3.1 Matrix Factorization Formulation

We begin from a matrix factorization formulation of the matrix completion problem. Observe that a matrix $Z \in \mathbb{R}^{N\times P}$ with rank no more than K can be factorized as $Z = XY$, where $X \in \mathbb{R}^{N\times K}$ and $Y \in \mathbb{R}^{K\times P}$. This fact motivates the following matrix factorization formulation

$$\begin{aligned} &\min_{X,Y,Z} \ \tfrac{1}{2}||XY - Z||_{\mathrm{F}}^2, \\ &\text{s.t.} \ \ z_{np} = d_{np}, \quad \forall (n, p) \in \Omega, \end{aligned} \tag{3.1}$$

where $Z \in \mathbb{R}^{N\times P}$ is an auxiliary matrix [32]. After solving (3.1), either XY or Z can be adopted as an estimate of the incomplete data matrix D.

There is an obvious difference between finding the lowest rank matrix D, often done by minimizing $||D||_*$ instead of $\mathrm{rank}(D)$ and finding a factorization with rank up to K like (3.1). The latter needs a rank K a priori or to dynamically update a rank estimate. This is a disadvantage but in applications such as internet traffic analysis [36] and sensor localization [22], K is known either theoretically or empirically. More importantly, as we will see in Sect. 4, the factorization approach enables efficient matrix completion in a decentralized network. It is advantageous over the nuclear-norm approach since the latter needs decentralized singular value decompositions, which are computationally expensive and even intractable in decentralized computing.

3.2 Centralized Algorithm: LMaFit

We review a centralized algorithm, developed in [32] and named as LMaFit, which is based on nonlinear Gauss–Seidel iterations applied to (3.1). At time k, LMaFit generates iterates

$$X^{k+1} = Z^k \left(Y^k\right)^{\mathrm{T}} \left(Y^k \left(Y^k\right)^{\mathrm{T}}\right)^{\dagger}, \tag{3.2}$$

$$Y^{k+1} = \left(\left(X^{k+1}\right)^{\mathrm{T}} X^{k+1}\right)^{\dagger} \left(X^{k+1}\right)^{\mathrm{T}} Z^k, \tag{3.3}$$

$$Z^{k+1} = X^{k+1} Y^{k+1} + P_{\Omega}\left(D - X^{k+1} Y^{k+1}\right), \tag{3.4}$$

where † stands for Moore-Penrose pseudo-inverse and P_{Ω} denotes projection onto the set Ω. If $(n, m) \in \Omega$, (3.4) yields $z_{nm}^{k+1} = d_{nm}$; otherwise, if $(n, m) \notin \Omega$ z_{nm}^{k+1} equals to the entry at the nth row and the mth column of $X^{k+1} Y^{k+1}$.

Because only the product XY, rather than individual X and Y, is needed for the sake of matrix completion, one can simplify the algorithm as shown in Lemma 3.1.

Lemma 3.1 *Replacing the updating rule* (3.2) *by*

$$X^{k+1} = cZ^k \left(Y^k\right)^{\mathrm{T}}, \quad c > 0, \tag{3.5}$$

does not change the sequences $\{X^k Y^k\}$ *and* $\{Z^k\}$.

LMaFit starts from initial estimates Y^0 and Z^0 and iteratively updates (3.5), (3.3), and (3.4). Convergence of the iterations (3.5), (3.3), and (3.4) is established in [32] and given in Lemma 3.2.

Lemma 3.2 *Let* $\{X^k, Y^k, Z^k\}$ *be a sequence generated by the iterations* (3.5), (3.3), *and* (3.4). *Assume that* $\{P_{\Omega^c}(X^k Y^k)\}$ *is bounded where* Ω^c *denotes the set of unobserved coordinates, i.e., the complementary set of* Ω. *Then any accumulation point of* $\{X^k, Y^k, Z^k\}$ *is a stationary point of* (3.1).

3.3 Challenge in Decentralized Implementation

Next we discuss the challenge of implementing LMaFit in a decentralized network. Recall that the whole data matrix is $D = [D_1, D_2, \cdots, D_L]$ and agent i only has access to the coordinate set Ω_i of the data matrix D_i. Therefore, (3.4) suggests that Z must be segmented to $[Z_1, Z_2, \cdots, Z_L]$ where $Z_i \in \mathbb{R}^{N \times P_i}$ is held by agent i. Similarly, to implement (3.3), Y is also split to different groups of columns such that $Y = [Y_1, Y_2, \cdots, Y_L]$ and $Y_i \in \mathbb{R}^{K \times P_i}$ is held by agent i. However, X cannot be segmented and distributed to the agents since (3.5) contains the summation of $Z_i (Y_i)^{\mathrm{T}}$ over all agents i.

To conclude, a naive implementation of LMaFit in a decentralized network generates iterates

$$X^{k+1} = c \sum_{i=1}^{L} Z_i^k \left(Y_i^k\right)^{\mathrm{T}}, \tag{3.6}$$

$$Y_i^{k+1} = \left(\left(X^{k+1}\right)^{\mathrm{T}} X^{k+1}\right)^{\dagger} \left(X^{k+1}\right)^{\mathrm{T}} Z_i^k, \quad \forall i, \tag{3.7}$$

$$Z_i^{k+1} = X^{k+1} Y_i^{k+1} + P_{\Omega_i}\left(D_i - X^{k+1} Y_i^{k+1}\right), \quad \forall i. \tag{3.8}$$

Observe that agent i is in charge of the updates of Y_i^{k+1} and Z_i^{k+1}, as shown in (3.7) and (3.8). However, computing (3.7) and (3.8) relies on a common X^{k+1}, whose value is determined by the summation of locally available terms $Z_i^k (Y_i^k)^{\mathrm{T}}$, as shown in (3.6). Therefore, such an implementation requires information aggregation of the whole network and is hence not decentralized. In Sect. 4, we shall propose a novel decentralized algorithm to address this issue.

4 Decentralized Matrix Completion

4.1 Dynamic Average Consensus

According to the discussion in Sect. 3.3, a decentralized implementation of LMaFit depends on an efficient approach to solve (3.6). Observe that if choosing $c = (1/L)$, (3.6) turns to

$$X^{k+1} = \frac{1}{L} \sum_{i=1}^{L} Z_i^k \left(Y_i^k\right)^{\mathrm{T}},$$

which is the well-known average consensus problem [8,24]. Given that each agent holds a local term $Z_i^k (Y_i^k)^{\mathrm{T}}$ at time k, the average consensus problem requires to average all local terms over the network. This can be done by introducing an iterative subroutine that mixes the local terms. This approach is costly, however, because the network needs a large amount of iterations to reach an *exact* consensus on the averaged value, and such an iterative subroutine must be run at every time.

A key observation in this paper is that *exact* average consensus at every time is not necessary. Instead, a properly designed *dynamic average consensus* step, which *inexactly* calculates the average of the local terms, enables an efficient decentralized implementation of LMaFit. Specifically, suppose that agent i has calculated the values of Y_i^k and Z_i^k at time k. We let the agents run a single mixing step, which is not necessarily able to reach the exact average $(1/L)\sum_{i=1}^{L} Z_i^k \left(Y_i^k\right)^{\mathrm{T}}$. This process is termed as *dynamic average consensus* since the local variables Y_i^k and Z_i^k are changing, and the network is trying to dynamically track the true average of the local terms $Z_i^k \left(Y_i^k\right)^{\mathrm{T}}$. If Y_i^k and Z_i^k gradually reach their steady-state values, successful dynamic tracking is possible, as we can observe from numerical experiments in Sect. 6.

We perform dynamic average consensus as follows. Define $X_{(i)}$ as the *local copy* of X at agent i and initialize it as $X_{(i)}^0$. At time $k = 0$, agent i updates its local copy $X_{(i)}^1$ through

$$X_{(i)}^1 = \sum_{j=1}^{L} w_{ij} X_{(j)}^0 - \alpha \left(X_{(i)}^0 - Z_i^0 \left(Y_i^0 \right)^{\mathrm{T}} \right), \tag{4.1}$$

and at time $k > 0$, the iteration becomes

$$\begin{aligned} X_{(i)}^{k+1} = X_{(i)}^k + \sum_{j=1}^{L} w_{ij} X_{(j)}^k - \sum_{j=1}^{L} \widetilde{w}_{ij} X_{(j)}^{k-1} \\ - \alpha \left(X_{(i)}^k - X_{(i)}^{k-1} - Z_i^k \left(Y_i^k \right)^{\mathrm{T}} + Z_i^{k-1} \left(Y_i^{k-1} \right)^{\mathrm{T}} \right). \end{aligned} \tag{4.2}$$

In the updates (4.1) and (4.2), $\alpha > 0$ is a constant stepsize, while w_{ij} and $\widetilde{w}_{ij}$ are entries at the ith row and the jth column of two mixing matrices $W \in \mathbb{R}^{L \times L}$ and $\widetilde{W} \in \mathbb{R}^{L \times L}$, respectively. We impose the following assumptions on W and $\widetilde{W}$ for decentralized implementation.

Assumption 4.1 *(Mixing matrices)* Consider a *connected network* $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ consisting of a set of agents $\mathcal{V} = \{1, 2, \cdots, L\}$ and a set of undirected edges $\mathcal{E}$. The mixing matrices $W, \widetilde{W} \in \mathbb{R}^{L \times L}$ satisfy

1. If $i \neq j$ and $(i, j) \notin \mathcal{E}$, then $w_{ij} = \widetilde{w}_{ij} = 0$; otherwise $w_{ij} > 0$, $\widetilde{w}_{ij} > 0$.
2. $W = W^{\mathrm{T}}$, $\widetilde{W} = \widetilde{W}^{\mathrm{T}}$.
3. $\mathrm{null}\{W - \widetilde{W}\} = \mathrm{span}\{1\}$, $\mathrm{null}\{I - \widetilde{W}\} \supseteq \mathrm{span}\{1\}$.
4. $\widetilde{W} \succ 0$ and $\frac{I+W}{2} \succcurlyeq \widetilde{W} \succcurlyeq W$.

The dynamic average consensus updates (4.1) and (4.2) are motivated by EXTRA, a first-order algorithm that solves decentralized static optimization problems [30]. If for all agents i and for all times $k \geqslant 0$, $Z_i^k (Y_i^k)^{\mathrm{T}} = Z_i^0 (Y_i^0)^{\mathrm{T}}$ are constant, then (4.1) and (4.2) guarantee that for every agent i, $X_{(i)}^k$ converges to the exact average $(1/L) \sum_{i=1}^{L} Z_i^0 (Y_i^0)^{\mathrm{T}}$. Here, the updates are adapted to reach a dynamic average consensus.

Assumptions on the mixing matrices W and $\widetilde{W}$ are also the same as those in EXTRA [30]. Observe that the static average consensus algorithms [8,24] and the decentralized subgradient method [23] choose the mixing matrix W in the same way; that is, W satisfies Part 1 and Part 2 of Assumption 4.1, which has an eigenvalue 1 and all others within $(-1, 1)$. Typical choices of W include the Laplacian-based constant edge weight matrix [33], the Metropolis constant edge weight matrix [2,34], and the symmetric fastest distributed linear averaging (FDLA) matrix [33]. After choosing W, a simple choice of $\widetilde{W}$ is $\widetilde{W} = (1/2)(I_{L \times L} + W)$ that satisfies Assumption 4.1.

We briefly summarize existing works on dynamic average consensus. When communication among the agents is continuous, proportional and proportional-integral algorithms are able to track the dynamic average of decentralized inputs [11,31]. The requirement of continuous communication is relaxed in [16] through an event-triggered mechanism, and [37] considers a discrete-time communication model. In the decentralized matrix completion algorithm in [18], the alternating direction method of multipliers is adopted for dynamic average consensus. However, the existence of both

primal and dual variables brings difficulties to its analysis. Considering the page limit and the focus of this paper, we leave detailed discussion on this issue as our future work.

Algorithm 1: D-LMaFit

Step 1: Initialization. Each agent i initializes its local variables Z_i^0 as $P_{\Omega_i}(D_i)$ and Y_i^0 as a random matrix, respectively.

Step 2: Update of $X_{(i)}$. At time k, agent i updates $X_{(i)}^{k+1}$. If $k = 0$, the update is [cf. (4.1)]

$$X_{(i)}^1 = \sum_{j=1}^L w_{ij} X_{(j)}^0 - \alpha \left(X_{(i)}^0 - Z_i^0 (Y_i^0)^{\mathrm{T}} \right),$$

and if $k > 0$, the update is [cf. (4.2)]

$$X_{(i)}^{k+1} = X_{(i)}^k + \sum_{j=1}^L w_{ij} X_{(j)}^k - \sum_{j=1}^L \widetilde{w}_{ij} X_{(j)}^{k-1} - \alpha \left(X_{(i)}^k - X_{(i)}^{k-1} - Z_i^k (Y_i^k)^{\mathrm{T}} + Z_i^{k-1} \left(Y_i^{k-1} \right)^{\mathrm{T}} \right).$$

Step 3: Update of Y_i and Z_i. At time k, agent i updates Y_i^{k+1} and Z_i^{k+1} from [cf. (3.7) and (3.8)]

$$Y_i^{k+1} = \left((X_{(i)}^{k+1})^{\mathrm{T}} X_{(i)}^{k+1} \right)^{\dagger} \left(X_{(i)}^{k+1} \right)^{\mathrm{T}} Z_i^k,$$

$$Z_i^{k+1} = X_{(i)}^{k+1} Y_i^{k+1} + P_{\Omega_i} \left(D_i - X_{(i)}^{k+1} Y_i^{k+1} \right).$$

4.2 Decentralized Algorithm

Summarizing the discussion in Sects. 3.3 and 4.1, we propose a decentralized matrix completion algorithm that replaces the exact average consensus step (3.6) by the dynamic average consensus step (4.1) or (4.2), which keeps the naturally decentralized updates (3.7) and (3.8). We outline the algorithm, named as D-LMaFit, in Algorithm 1.

We interpret D-LMaFit as a combination of dynamic average consensus on *public information* and autonomous local update on *private information*. The low-rank matrix factorization $XY = X[Y_1, Y_2, \cdots, Y_L]$ can be viewed as the product of a *public matrix* X and a collection of *private matrices* Y_i, $i = 1, 2, \cdots, L$. The approximation of XY, denoted by $Z = [Z_1, Z_2, \cdots, Z_L]$, is also a collection of *private matrices* Z_i. In D-LMaFit, agent i updates its local copy of the public matrix X, denoted by $X_{(i)}$, from private matrices Y_i and Z_i. After calculating $X_{(i)}$, Y_i and Z_i can be obtained locally.

With particular note, the proposed *decentralized algorithm* is different than a *distributed or parallel algorithm* [26], as the latter needs a fusion center to coordinate the agents, while the former relies on autonomous collaboration.

5 Topology-Dependent Privacy Preservation

5.1 Malicious Agent

Suppose that a malicious agent M attempts to recover the local data matrices of some other agents through limited information exchange during the optimization process.

We focus on the case that there is only one malicious agent for simplicity; indeed, multiple malicious agents can be treated as a single *super* agent and our analysis still holds. We define the behavior and the goal of the malicious agent M as follows.

Assumption 5.1 *(The malicious agent M)*

1. The malicious agent M is passive, in the sense that it follows the decentralized matrix completion algorithm and does not provide false values of the public matrix $X_{(M)}$ to any other agents.
2. The malicious agent M is interested in recovering the local data matrices D_i of a set of agents $i \in \mathcal{I}$. This is equivalent to recovering either the private matrices Y_i or the auxiliary matrices Z_i, $\forall i \in \mathcal{I}$.

Part 1 of Assumption 5.1 indicates that the malicious agent M does not perturb the optimization process; otherwise, it may be able to obtain more information through sending out properly designed values of $X_{(M)}$. This is reasonable since a malicious agent or a person hijacking the agent often wants to conceal its identity. Part 2 of Assumption 5.1 means that the malicious agent M attempts to obtain D_i via recovering Z_i or Y_i, $\forall i \in \mathcal{I}$. Observe that Z_i is a completion of D_i. Similarly, if the private matrix Y_i is known and all the agents have reached consensus on the public matrix X, then $X_{(M)}Y_i$ is also a completion of D_i. With particular note, we omit the time index here. Indeed, the malicious agent M is interested in the values of Z_i^k and $X_{(M)}^k Y_i^k$ when k is large enough; at that time, these values are hopefully close to D_i.

Remark 5.2 In the analysis below, we discuss protecting the values of $Z_i Y_i^{\mathrm{T}}$, other than those of Z_i or Y_i^{T}, as the optimization process naturally encodes $Z_i Y_i^{\mathrm{T}}$ in a series of linear equations. We argue that even M knows $Z_i Y_i^{\mathrm{T}}$, it does not necessarily know either Z_i or Y_i^{T}. Suppose that in the idealized case, all local copies of the public matrix, including $X_{(M)}$, converge to the true value. By $Z_i = X_{(M)}Y_i$, knowing $Z_i Y_i^{\mathrm{T}}$ is equivalent to knowing $X_{(M)}Y_i Y_i^{\mathrm{T}}$. In this case, M may be able to calculate $Y_i Y_i^{\mathrm{T}}$; however, finding Y_i and Z_i, similarly, is still ambiguous.

Remark 5.3 We also assume that the malicious agent M is aware of the values of the mixing matrices W and $\widetilde{W}$. Note that in the existing decentralized algorithms that generate the mixing matrices [2,8,23,24,30,33,34], the malicious agent M may be able to know some values of their entries, but not necessarily all of them. Hence, the privacy-preserving property of D-LMaFit is stronger than that analyzed in this section.

5.2 Invertibility of Linear Time-Invariant System

Observe (4.1) and (4.2). During the optimization process, the malicious agent M is able to obtain a series of public matrices $X_{(j)}$ from itself and its neighbors, $j \in M \cup \mathcal{N}_M$. These public matrices encode the series of values of $Z_i Y_i^{\mathrm{T}}$, in which $i \in \mathcal{I}$ and $\mathcal{I}$ is the set of interested agents. Hence, preserving privacy of the agents in $\mathcal{I}$ boils down to the following question: *Is the malicious agent M able to solve such a series of linear equations to find $Z_i Y_i^{\mathrm{T}}$, $i \in \mathcal{I}$?* This question is equivalent to analyzing the invertibility of a linear time-invariant system, as we shall discuss below.

We define the linear time-invariant system as follows. Define

$$P^k = \begin{bmatrix} X_{(1)}^k \\ X_{(2)}^k \\ \cdots \\ X_{(L)}^k \end{bmatrix} \in \mathbb{R}^{LN\times K}, \quad Q^k = \begin{bmatrix} \alpha Z_1^k \left(Y_1^k\right)^{\mathrm{T}} \\ \alpha Z_2^k \left(Y_2^k\right)^{\mathrm{T}} \\ \cdots \\ \alpha Z_L^k \left(Y_L^k\right)^{\mathrm{T}} \end{bmatrix} \in \mathbb{R}^{LN\times K},$$

and

$$\widetilde{P}^k = \begin{bmatrix} P^{k+1} \\ P^k \end{bmatrix} \in \mathbb{R}^{2LN\times K}, \quad \widetilde{Q}^k = \begin{bmatrix} Q^{k+1} - Q^k \end{bmatrix} \in \mathbb{R}^{LN\times K}.$$

Further, define

$$A = \begin{bmatrix} (1-\alpha)I_{LN\times LN} + W \otimes I_{N\times N} & \alpha I_{LN\times LN} - \widetilde{W} \otimes I_{N\times N} \\ I_{LN\times LN} & 0_{LN\times LN} \end{bmatrix} \in \mathbb{R}^{2LN\times 2LN},$$

and

$$B = \begin{bmatrix} I_{LN\times LN} \\ 0_{LN\times LN} \end{bmatrix} \in \mathbb{R}^{2LN\times 2LN}.$$

Using these definitions, the recursion (4.2) can be represented by

$$\widetilde{P}^{k+1} = A\widetilde{P}^k + B\widetilde{Q}^k. \tag{5.1}$$

The malicious agent M observes some blocks in $\widetilde{P}^{k+1}$ (namely $X_{(i)}^{k+2}$ and $X_{(i)}^{k+1}$, $\forall i \in \mathcal{N}_M \cup M$) and attempts to recover some blocks in $\widetilde{Q}^k$ (namely $\alpha Z_i^{k+1}\left(Y_i^{k+1}\right)^{\mathrm{T}} - \alpha Z_i^k \left(Y_i^k\right)^{\mathrm{T}}, \forall i \in \mathcal{I}$) in (5.1). Rewrite this as a linear time-invariant system

$$\mathcal{S}: \begin{cases} \widetilde{P}^{k+1} = A\widetilde{P}^k + B_{\mathcal{I}}\widetilde{Q}_{\mathcal{I}}^k + B_{\mathcal{I}^c}\widetilde{Q}_{\mathcal{I}^c}^k, \\ \widetilde{V}^k = C\widetilde{P}^k. \end{cases} \tag{5.2}$$

In (5.2), $\widetilde{Q}_{\mathcal{I}}^k \in \mathbb{R}^{|\mathcal{I}|N\times K}$ selects those row blocks in $\widetilde{Q}^k$ belonging to the interested agents in $\mathcal{I}$, and $B_{\mathcal{I}} \in \mathbb{R}^{2LN\times|\mathcal{I}|N}$ selects the corresponding column blocks in B. Similarly, $\widetilde{Q}_{\mathcal{I}^c}^k \in \mathbb{R}^{(L-|\mathcal{I}|)N\times K}$ selects those row blocks in $\widetilde{Q}^k$ not belonging to the interested agents in $\mathcal{I}$, and $B_{\mathcal{I}^c} \in \mathbb{R}^{2LN\times(L-|\mathcal{I}|)N}$ selects the corresponding column blocks in B. In the observation equation $\widetilde{V}^k = C\widetilde{P}^k$, $C \in \mathbb{R}^{2(|\mathcal{N}_M|+1)N\times 2LN}$ stands for a selection matrix that selects those row blocks in $\widetilde{P}^k$, which belong to agent M and its neighbors, to construct the observation matrix $\widetilde{V}^k \in \mathbb{R}^{2(|\mathcal{N}_M|+1)N\times K}$.

In the linear time-invariant system (5.2), the input sequences are $\{\widetilde{Q}_{\mathcal{I}}^k\}$ and $\{\widetilde{Q}_{\mathcal{I}^c}^k\}$, the state sequence is $\{\widetilde{P}^k\}$, and the output sequence is $\{\widetilde{V}^k\}$. We are interested in the subsystem from $\{\widetilde{Q}_{\mathcal{I}}^k\}$ to $\{\widetilde{V}^k\}$, which is invertible if one can solve the values of $\widetilde{Q}_{\mathcal{I}}^k$ from the observations $\widetilde{V}^k$ and a series of linear equations [3]. Hence, analyzing the

privacy preservation property of D-LMaFit boils down to investigating the invertibility of the system (5.2). With particular note, in a practical network, the malicious agent M may utilize the special structures of $\widetilde{Q}_{\mathcal{I}}^k$ to enable reconstruction, other than simply solve the linear equations. For example, each block of $\widetilde{Q}_{\mathcal{I}}^k$ is known as the difference between two matrices, which are possibly low-rank. Hence, the privacy preservation property discussed here is in a weak sense.

5.3 Preliminaries

Our analysis uses the concept of z-transfer matrix of the system $\mathcal{S}$ defined in (5.2). Without loss of generality, let $N = K = 1$ such that $\widetilde{P}^k$, $\widetilde{Q}_{\mathcal{I}}^k$, $\widetilde{Q}_{\mathcal{I}^c}^k$, and $\widetilde{V}^k$ are all vectors other than matrices. As the system $\mathcal{S}$ is linear, our analysis still holds for the matrix case.

Denote the z-transfer matrix from $\widetilde{Q}_{\mathcal{I}}^k$ to $\widetilde{V}^k$ as $T_{\mathcal{I}}(z) = C(zI_{2LN\times 2LN} - A)^{-1}B_{\mathcal{I}}$, that from $\widetilde{Q}_{\mathcal{I}^c}^k$ to $\widetilde{V}^k$ as $T_{\mathcal{I}^c}(z) = C(zI_{2LN\times 2LN} - A)^{-1}B_{\mathcal{I}^c}$, and that from $\widetilde{Q}^k$ to $\widetilde{V}^k$ as $T(z) = C(zI_{2LN\times 2LN} - A)^{-1}B$. We have that

$$\operatorname{rank}\big(T(z)\big) = \operatorname{rank}\left(\left[\, T_{\mathcal{I}}(z), T_{\mathcal{I}^c}(z) \,\right]\right).$$

Observe that the elements of $T(z)$, $T_{\mathcal{I}}(z)$, and $T_{\mathcal{I}^c}(z)$ are rational functions, and the matrix rank is defined over the rational expression field [3].

The rank of a transfer matrix is related to the topology of a *virtual network*, as shown in [10]. To be specific, let each input, state, and output of a linear system be a node in the virtual network. There exists a directed edge from one node to another, if the source node determines the value of the destination node in the state-space model. Consider the transfer matrix from a group of input nodes to a group of output nodes. Its rank equals to the maximum number of disjoint paths from the input nodes to the output nodes [10].

Applying this result to the system (5.2), we can see that the topology of the virtual network is determined by that of the underlying communication graph $(\mathcal{V}, \mathcal{E})$ since the state matrix A is constructed from $(\mathcal{V}, \mathcal{E})$. This way, we connect the ranks of $T(z)$ and $T_{\mathcal{I}}(z)$ with the network topology. Here are two conclusions that are useful in the analysis below: (i) $\operatorname{rank}(T(z))$ equals to the maximum number of vertex disjoint paths from all the agents in the network to $M \cup \mathcal{N}_M$; (ii) $\operatorname{rank}(T_{\mathcal{I}^c}(z))$ equals to the maximum number of vertex disjoint paths from the agents in $\mathcal{I}^c$ to $M \cup \mathcal{N}_M$.

5.4 Main Results

We give our main theorem below.

Theorem 5.4 *Consider a subsystem of $\mathcal{S}$ in (5.2), where the input is $\widetilde{Q}_{\mathcal{I}}^k$ and the output is $\widetilde{V}^k$. The subsystem is invertible if and only if $\mathcal{I} \cup \mathcal{N}_{\mathcal{I}} \subseteq M \cup \mathcal{N}_M$.*

Proof Part I: Proof of sufficiency. Proof of sufficiency is straightforward. Observe that if $\mathcal{I} \cup \mathcal{N}_{\mathcal{I}} \subseteq M \cup \mathcal{N}_M$, the malicious agent M has full knowledge of those row blocks,

which correspond to $\mathcal{I}$, of the linear equation $\widetilde{P}^{k+1} = A\widetilde{P}^k + B\widetilde{Q}^k$. To see so, recall that M has access to the corresponding blocks of $\widetilde{P}^{k+1}$ as $\mathcal{I}$ belongs to its neighbor set. On the other hand, because of the special structure of A, the corresponding blocks of $A\widetilde{P}^k$ are only determined by those blocks of $\widetilde{P}^k$ belonging to the agents in $\mathcal{I} \cup \mathcal{N}_{\mathcal{I}}$, which are known by M as $\mathcal{I} \cup \mathcal{N}_{\mathcal{I}} \subseteq M \cup \mathcal{N}_M$. Therefore, the malicious agent M is able to calculate of $\widetilde{Q}^k_{\mathcal{I}}$ directly. Below we prove necessity.

Part II: Proof of necessity. The proof of necessity is similar to that of Theorem 7 in [35]. However, our system is essentially second order while the one in [35] is first order, which makes the proof different. To be concrete, relations between the underlying communication graph and the virtual graphs constructed from the systems are different.

Step 1*: A necessary rank condition for invertibility.* First, we show that to determine a unique sequence of inputs $\widetilde{Q}^k_{\mathcal{I}}$ from the observations $\widetilde{V}^k$, we must have

$$\text{rank}\,(T(z)) - \text{rank}\,(T_{\mathcal{I}^c}(z)) = |\mathcal{I}|. \tag{5.3}$$

Apparently, the difference of the two ranks is at most $|\mathcal{I}|$. Now we prove it must be $|\mathcal{I}|$ by contradiction. If the difference of the two ranks is less than $|\mathcal{I}|$, there exists at least one column of $T_{\mathcal{I}}(z)$, denoted by $T^i_{\mathcal{I}}(z)$, which is linearly dependent on the other columns of $T(z)$. Then we can find a vector $\widetilde{Q}(z)$ whose ith element is nonzero and satisfies $T(z)\widetilde{Q}(z) = 0$, where $i \in \mathcal{I}$. In other words, no matter how the input sequence corresponding to the ith element of $\widetilde{Q}(z)$ varies, the output is zero. Therefore, we are unable to determine a unique sequence of inputs $\widetilde{Q}^k_{\mathcal{I}}$ from the observations $\widetilde{V}^k$ if (5.3) does not hold.

Step 2*: $\mathcal{I}$ being a subset of $M \cup \mathcal{N}_M$.* Second, according to [10], $\text{rank}(T(z))$ equals to the maximum number of vertex disjoint paths from all the agents in the network to $M \cup \mathcal{N}_M$. Because $M \cup \mathcal{N}_M$ is a subset of all the agents, $\text{rank}(T(z))$ must be $|\mathcal{N}_M| + 1$. Therefore, (5.3) can be rewritten as

$$|\mathcal{N}_M| + 1 = \text{rank}\,(T_{\mathcal{I}^c}(z)) + |\mathcal{I}|. \tag{5.4}$$

Splitting the set $M \cup \mathcal{N}_M$ into two sets, $M \cup \mathcal{N}_M - \mathcal{I}$ and $\{M \cup \mathcal{N}_M\} \cap \mathcal{I}$, we have that

$$|\mathcal{N}_M| + 1 = |M \cup \mathcal{N}_M - \mathcal{I}| + |\{M \cup \mathcal{N}_M\} \cap \mathcal{I}|. \tag{5.5}$$

If $\mathcal{I}$ is not contained in $M \cup \mathcal{N}_M$, then it holds $|\{M \cup \mathcal{N}_M\} \cap \mathcal{I}| < |\mathcal{I}|$. Furthermore, from [10], rank $(T_{\mathcal{I}^c}(z))$ equals to the maximum number of vertex disjoint paths from the agents in $\mathcal{I}^c$ to $M \cup \mathcal{N}_M$. Since $M \cup \mathcal{N}_M - \mathcal{I}$ is a subset of $M \cup \mathcal{N}_M$, we have $|M \cup \mathcal{N}_M - \mathcal{I}| \leqslant \text{rank}\,(T_{\mathcal{I}^c}(z))$. Thus, (5.5) implies $|\mathcal{N}_M| + 1 < \text{rank}\,(T_{\mathcal{I}^c}(z)) + |\mathcal{I}|$ that contradicts with (5.4). Therefore, $\mathcal{I}$ must be a subset of $M \cup \mathcal{N}_M$.

Step 3*: $\mathcal{N}_{\mathcal{I}}$ being a subset of $M \cup \mathcal{N}_M$.* Third, if $\mathcal{N}_{\mathcal{I}}$ is not contained in $M \cup \mathcal{N}_M$, then we have $|M \cup \mathcal{N}_M - \mathcal{I}| < \text{rank}\,(T_{\mathcal{I}^c}(z))$. As we have already known that $\mathcal{I}$ is contained in $M \cup \mathcal{N}_M$, we have $|\{M \cup \mathcal{N}_M\} \cap \mathcal{I}| = |\mathcal{I}|$. Again, (5.5) implies $|\mathcal{N}_M| + 1 < \text{rank}\,(T_{\mathcal{I}^c}(z)) + |\mathcal{I}|$ that contradicts with (5.4). Therefore, $\mathcal{N}_{\mathcal{I}}$ must also be a subset of $M \cup \mathcal{N}_M$. This completes the proof.

We illustrate Theorem 5.4 with two examples. In Fig. 1, the set of interested agents is $\mathcal{I} = \{1, 2, 3, 4\}$, and its neighbor set is $\mathcal{N}_{\mathcal{I}} = \{5, 7, M\}$; the neighbor set of M is $\mathcal{N}_M = \{1, 3, 4, 7\}$. According to Theorem 5.4, M is unable to recover the sensitive information of all agents in $\mathcal{I}$. Indeed, the privacy of 1, 2, and 3 is guaranteed, but it is not the case for 4 because 4 is only connected to M. If M adds neighbors, 2 and 5, as shown in Fig. 2, then the neighbor set of M is $\mathcal{N}_M = \{1, 2, 3, 4, 5, 7\}$. In this case, the malicious agent M is able to recover the sensitive information of all the agents in $\mathcal{I}$. Therefore, Theorem 5.4 provides us a guideline of designing network topology to protect data privacy.

Remark 5.5 Recall that invertibility means solving $\widetilde{Q}^k_{\mathcal{I}}$ from a set of linear equations without using any prior knowledge. However, $\widetilde{Q}^k_{\mathcal{I}}$ has its own structure. Consider $\widetilde{Q}^k$ that contains $\widetilde{Q}^k_{\mathcal{I}}$; $\widetilde{Q}^k = \left[Q^{k+1} - Q^k\right]$ and the ith row block of Q^k is $\alpha Z_i^k (Y_i^k)^{\mathrm{T}}$, a factorization of two private matrices Z_i^k and Y_i^k. If $\alpha Z_i^k (Y_i^k)^{\mathrm{T}}$ is low-rank, which means that $K < \min(N, P_i)$, the malicious agent may use this prior knowledge to obtain guesses of them. We leave privacy preservation under this circumstance as an open question to future research.

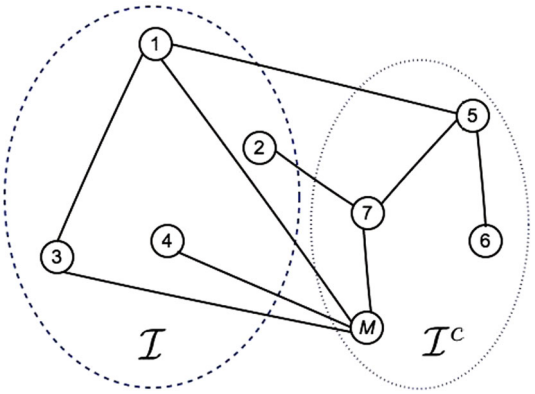

Fig. 1 An illustrative example: the malicious agent M is unable to recover the sensitive information of agents 1, 2, and 3 in $\mathcal{I}$, while it is able to recover that of agent 4

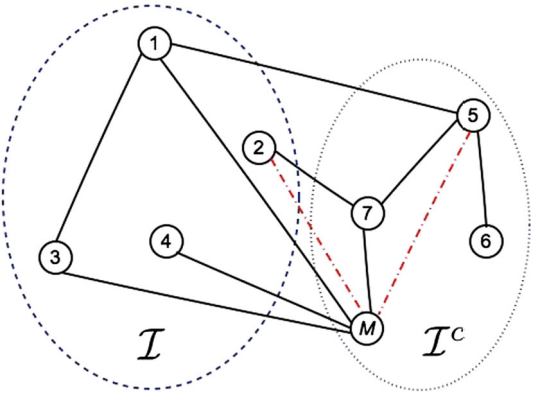

Fig. 2 An illustrative example: the malicious agent M is able to recover the sensitive information of all agents in $\mathcal{I}$

6 Numerical Experiments

In the numerical experiments, we assume that in the graph only 20% of the edges are bi-directional connected. A noise-free data matrix D of rank K is generated by $D = U\mathrm{Diag}(d)V^{\mathrm{T}}$, where the entries of $U \in \mathbb{R}^{N\times K}$, $d \in \mathbb{R}^{K}$, and $V \in \mathbb{R}^{P\times K}$ are i.i.d sampled from the standard normal distribution. Each of the L agents holds P/L columns of D. The subset Ω contains randomly chosen $100 \times p$ remaining entries need completion. We set the simulation parameters $L = 50$, $N = 200$, $P = 2000$, and $K = 4$.

We compare two algorithms in the numerical experiments: the centralized LMaFit and the decentralized D-LMaFit. In D-LMaFit, the weight matrix W is generated from the FDLA rule and $\widetilde{W} = (1+W)/2$. Performance of the algorithms is measured by the relative error: $||D - XY||_{\mathrm{F}}/||D||_{\mathrm{F}}$. In D-LMaFit, we get XY by collecting the products $X_{(i)}Y_i$, i.e., $XY = [X_{(1)}Y_1, \cdots, X_{(L)}Y_L]$.

We compare LMaFit and D-LMaFit with different stepsizes α in Fig. 3. The decentralized algorithm is able to converge to the same solution as that of the centralized one, even though the inexact average consensus step introduces errors. The decentralized algorithm is slower than its centralized counterpart; this is reasonable since information diffusion throughout the whole network requires more time. The stepsize α is a critical parameter of the decentralized algorithms. Figure 3 shows that, as long as α is chosen within a proper range, D-LMaFit still reaches the same accuracy of recovery, though the convergence speed is different. From Fig. 3, $\alpha = 0.4$ gives the fastest convergence, while $\alpha = 0.1$ gives the slowest. If α is too large, say $\alpha = 0.6$, D-LMaFit converges to a wrong solution.

In the second set of experiments, we show the impact of the different sampling ratios $p = 0.2, 0.3, 0.4$. For each p, we hand-tune the stepsize α to its best value in D-LMaFit. In Fig. 4, when the sampling ratio p is high, both centralized and decentralized algorithms are able to provide accurate recover. When p is small, their performance

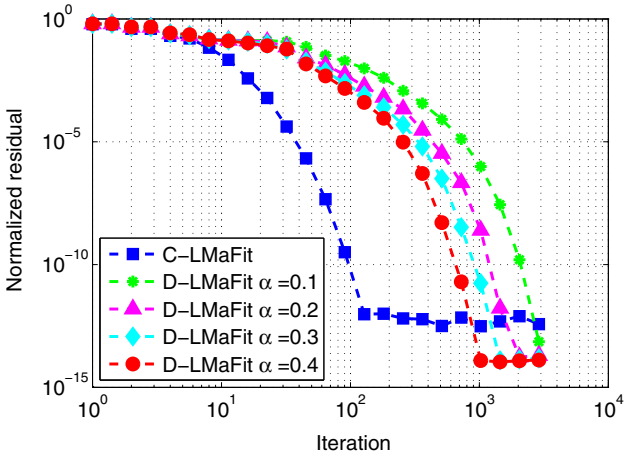

Fig. 3 Performance of D-LMaFit for different stepsizes α; here $p = 0.4$

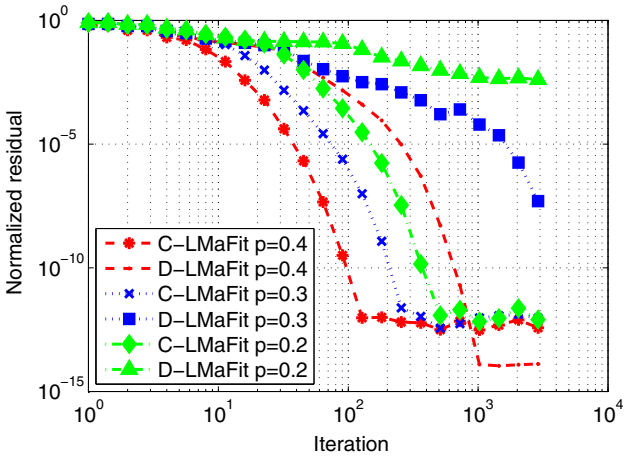

Fig. 4 Performance of D-LMaFit and RD-LMaFit for different sampling ratios p

degrades and convergence takes longer time. D-LMaFit suffers more from less samples than its centralized counterpart, because each agent has less information at its hand and relies more on the cooperation of the whole network.

7 Conclusion

In this paper, we propose a decentralized privacy-preserving algorithm, D-LMaFit, to solve the matrix completion problem. The main idea is to factorize the data matrices such that the updates boil down to alternating minimization over a public matrix and multiple private matrices. To enable efficient decentralized implementation, we let each agent have a local copy of the public matrix and force the local copies to reach a consensus through a dynamic average consensus scheme. We prove the sufficient and necessary condition under which a malicious agent is able to recover the local data matrices of some interested ones. The condition is determined only by network topology; thus, it provides a guideline of designing a privacy-preserving network.

References

[1] Bennett, J., Lanning, S.: The Netflix prize. In: Proceedings of Knowledge Discovery and Data Mining Cup (2007)
[2] Boyd, S., Diaconis, P., Xiao, L.: Fastest mixing Markov chain on a graph. SIAM Rev. **46**, 667–689 (2004)
[3] Brogan, W.: Modern Control Theory, 3rd edn. Prentice-Hall, Upper Saddle River (1991)
[4] Cai, J., Candes, E., Shen, Z.: A singular value thresholding algorithm for matrix completion. SIAM J. Optim. **20**, 1956–1982 (2010)
[5] Candes, E., Recht, B.: Exact matrix completion via convex optimization. Found. Comput. Math. **9**, 717–772 (2009)
[6] Candes, E., Plan, Y.: Matrix completion with noise. Proc. IEEE **98**(6), 925–936 (2010)
[7] Candes, E., Eldar, Y., Strohmer, T., Voroninski, V.: Phase retrieval via matrix completion. SIAM J. Imaging Sci. **6**, 199–225 (2013)

[8] Cao, Y., Yu, W., Ren, W., Chen, G.: An overview of recent progress in the study of distributed multi-agent coordination. IEEE Trans. Ind. Inform. **9**, 427–438 (2013)
[9] Chaudhuri, K., Monteleoni, C.: Privacy-preserving logistic regression. In: Proceedings of Neural Information Processing Systems (2009)
[10] Dion, J., Commault, C., van der Woude, J.: Generic properties and control of linear structured systems: a survey. Automatica **39**, 1125–1144 (2003)
[11] Freeman, R., Yang, P., Lynch, K.: Stability and convergence properties of dynamic average consensus estimators. In: Proceedings of IEEE Conference on Decision and Control (2006)
[12] Hu, Y., Zhang, D., Ye, J., Li, X., He, X.: Fast and accurate matrix completion via truncated nuclear norm regularization. IEEE Trans. Pattern Anal. Mach. Intell. **35**, 2117–2130 (2013)
[13] Johansson, B.: On Distributed optimization in networked systems. Ph.D. Thesis, Royal Institute of Technology (2008)
[14] Kearns, M., Tan, J., Wortman, J.: Privacy-preserving belief propagation and sampling. In: Proceedings of Neural Information Processing Systems (2007)
[15] Keshavan, R., Montanari, A., Oh, S.: Matrix completion from noisy entries. J. Mach. Learn. Res. **11**, 2057–2078 (2010)
[16] Kia, S., Cortes, J., Martinez, S.: Dynamic average consensus with distributed event-triggered communication. In: Proceedings of IEEE Conference on Decision and Control (2014)
[17] Li, H., Lu, Z., Wang, Z., Ling, Q., Li, W.: Detection of blotch and scratch in video based on video decomposition. IEEE Trans. Circuits Syst. Video Technol. **23**, 1887–1900 (2013)
[18] Ling, Q., Xu, Y., Yin, W., Wen, Z.: Decentralized low-rank matrix completion. In: Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (2012)
[19] Liu, Z., Vandenberghe, L.: Interior-point method for nuclear norm approximation with application to system identification. SIAM J. Matrix Anal. Appl. **31**, 1235–1256 (2009)
[20] Ma, S., Goldfarb, D., Chen, L.: Fixed point and Bregman iterative methods for matrix rank minimization. Math. Program. Ser. A **128**, 321–353 (2011)
[21] Mardani, M., Mateos, G., Giannakis, G.: Decentralized sparsity-regularized rank minimization: algorithms and applications. IEEE Trans. Signal Process. **61**, 5374–5388 (2013)
[22] Montanari, A., Oh, S.: On positioning via distributed matrix completion. In: Proceedings of IEEE Sensor Array and Multichannel Signal Processing (2010)
[23] Nedic, A., Ozdaglar, A.: Distributed subgradient methods for multiagent optimization. IEEE Trans. Autom. Control **54**, 48–61 (2009)
[24] Olfati-Saber, R., Fax, J., Murray, R.: Consensus and cooperation in networked multi-agent systems. Proc. IEEE **95**, 215–233 (2007)
[25] Recht, B., Fazel, M., Parrilo, P.: Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization. SIAM Rev. **52**, 471–501 (2010)
[26] Recht, B., Re, C.: Parallel stochastic gradient algorithms for large-scale matrix completion. Math. Program. Comput. **5**, 201–226 (2013)
[27] Ruping, S.: SVM classifier estimation from group probabilities. In: Proceedings of International Conference on Machine Learning (2010)
[28] Sakuma, J., Arai, H.: Online prediction with privacy. In: Proceedings of International Conference on Machine Learning (2010)
[29] Sayed, A.: Adaptation, learning, and optimization over networks. Found. Trends Mach. Learn. **7**, 311–801 (2014)
[30] Shi, W., Ling, Q., Wu, G., Yin, W.: EXTRA: An exact first-order algorithm for decentralized consensus optimization. arXiv:1404.6264
[31] Spanos, D., Olfati-Saber, R., Murray, R.: Dynamic consensus for mobile networks. In: Proceedings of International Federation of Automatic Control World Congres (2005)
[32] Wen, Z., Yin, W., Zhang, Y.: Solving a low-rank factorization model for matrix completion by a nonlinear successive over-relaxation algorithm. Math. Program. Comput. **4**, 333–361 (2012)
[33] Xiao, L., Boyd, S.: Fast linear iterations for distributed averaging. Syst. Control Lett. **53**, 65–78 (2004)
[34] Xiao, L., Boyd, S., Kim, S.: Distributed average consensus with least-mean-square deviation. J. Parallel Distrib. Comput. **67**, 33–46 (2007)
[35] Yan, F., Sundaram, S., Vishwanathan, S., Qi, Y.: Distributed autonomous online learning: Regrets and intrinsic privacy-preserving properties. IEEE Trans. Knowl. Data Eng. **25**, 2483–2493 (2013)

[36] Zhang, Y., Roughan, M., Willinger, W., Qiu, L.: Spartio-temporal compressive sensing and Internet traffic matrices. In: Proceedings of Association for Computing Machinery's Special Interest Group on Data Communications (2009)
[37] Zhu, M., Martinez, S.: Discrete-time dynamic average consensus. Automatica **46**, 322–329 (2010)