DECENTRALIZED LOW-RANK MATRIX COMPLETION

Qing Ling[*] *Yangyang Xu*[†] *Wotao Yin*[†] *Zaiwen Wen*[‡]

[*] Department of Automation, University of Science and Technology of China, Hefei, Anhui, China
[†] Department of Computational and Applied Mathematics, Rice University, Houston, Texas, USA
[‡] Department of Mathematics, Shanghai Jiaotong University, Shanghai, China

ABSTRACT

This paper introduces algorithms for the *decentralized low-rank matrix completion* problem. Assume a low-rank matrix $\mathbf{W} = [\mathbf{W}_1, \mathbf{W}_2, ..., \mathbf{W}_L]$. In a network, each agent ℓ observes some entries of $\mathbf{W}_\ell$. In order to recover the unobserved entries of $\mathbf{W}$ via decentralized computation, we factorize the unknown matrix $\mathbf{W}$ as the product of a *public matrix* $\mathbf{X}$, common to all agents, and a *private matrix* $\mathbf{Y} = [\mathbf{Y}_1, \mathbf{Y}_2, ..., \mathbf{Y}_L]$, where $\mathbf{Y}_\ell$ is held by agent ℓ. Each agent ℓ alternatively updates $\mathbf{Y}_\ell$ and its *local estimate* of $\mathbf{X}$ while communicating with its neighbors toward a consensus on the estimate. Once this consensus is (nearly) reached throughout the network, each agent ℓ recovers $\mathbf{W}_\ell = \mathbf{X}\mathbf{Y}_\ell$, and thus $\mathbf{W}$ is recovered. The communication cost is scalable to the number of agents, and $\mathbf{W}_\ell$ and $\mathbf{Y}_\ell$ are kept private to agent ℓ to a certain extent. The algorithm is accelerated by extrapolation and compares favorably to the centralized code in terms of recovery quality and robustness to rank over-estimate.

Index Terms— decentralized algorithm, low-rank matrix completion, matrix factorization, privacy protection

1. INTRODUCTION

Recovering an (approximately) low-rank matrix from an incomplete set of its entries is of great research interest (e.g., [1, 2]). This problem arises in collaborative filtering [3], internet traffic analysis [4], and sensor localization [5], etc.

Some existing algorithms [6, 7] solve convex problems based on nuclear-norm minimization, which tends to produce a low-rank matrix; other algorithms [8, 9] decompose the unknown matrix as the product of two lower-dimensional matrices and solve nonconvex problems. The convex approach yields global optimal solutions but needs singular value decompositions (SVDs), which are expensive on large-scale matrices. The nonconvex approach is subject to local minima (guarantees also exist [8] due to good initial solutions), yet its performance is often satisfactory, and the computational cost is lower due to avoiding SVDs [9].

The work of Q. Ling is supported in part by the NSFC grant 61004137. The work of Y. Xu and W. Yin is supported in part by NSF ECCS-1028790, NSF DMS-07-48839, ONR Grant N00014-08-1-1101, ARL and ARO grant W911NF-09-1-0383, and DOD/AFOSR FA9550-10-C-0108 Phase II.

Most these existing algorithms are *centralized* except [5] for symmetric matrices only. In network applications, however, *decentralized computing* is preferred for communication, network topology, and privacy reasons. In *decentralized matrix completion*, a set of geographically distributed but connected agents collect the matrix entries and recover the matrix via communications within neighborhoods; there is *no fusion center* to collect the sampled entries or to recover the matrix.

This paper develops algorithms for decentralized low-rank matrix completion. It assumes a set of L agents and a matrix

$$\mathbf{W} = [\mathbf{W}_1, \mathbf{W}_2, ..., \mathbf{W}_L] \in \mathcal{R}^{N\times M} \tag{1}$$

of rank $r \ll \min\{M, N\}$. Each agent ℓ observes some entries of $\mathbf{W}_\ell \in \mathcal{R}^{N\times M_\ell}$, $\sum_{\ell=1}^{L} M_\ell = M$. The set of observations is $\Omega = \cup_{\ell=1}^{L}\Omega_\ell$. We introduce a *public matrix* $\mathbf{X} \in \mathcal{R}^{N\times r}$, which is common to all agents, and a *private matrix* $\mathbf{Y} = [\mathbf{Y}_1, \mathbf{Y}_2, ..., \mathbf{Y}_L] \in \mathcal{R}^{r\times M}$, where each $\mathbf{Y}_\ell$ corresponds to $\mathbf{W}_\ell$ and is held by agent ℓ. We reconstruct $\mathbf{W} = \mathbf{X}\mathbf{Y}$, which is at most rank r, by recovering $\mathbf{X}$ and $\mathbf{Y}$ in a decentralized fashion. The communication cost is scalable to L, and $\mathbf{W}_\ell$ and $\mathbf{Y}_\ell$ are not shared between agents for privacy protection.

2. FORMULATION AND A CENTRALIZED ALGORITHM

Recent work [9] recovers $\mathbf{W} = \mathbf{X}\mathbf{Y}$ by solving

$$\begin{array}{ll}\min\limits_{\mathbf{X},\mathbf{Y},\mathbf{Z}} & \frac{1}{2}||\mathbf{X}\mathbf{Y} - \mathbf{Z}||_F^2, \\ s.t. & Z_{n,m} = W_{n,m}, \quad \forall (n,m) \in \Omega,\end{array} \tag{2}$$

where $\mathbf{Z} \in \mathcal{R}^{N\times M}$ is an auxiliary matrix and $||\cdot||_F$ denotes the matrix Frobenius norm.

There is an obvious difference between finding the lowest-rank matrix $\mathbf{W}$, often done by minimizing $||\mathbf{W}||_*$ instead of $\mathrm{rank}(\mathbf{W})$, and finding a factorization with rank up to r like (2). The latter needs a rank r *a priori*, or dynamically update a rank estimate. This is a disadvantage but in applications such as internet traffic analysis [4] and sensor localization [5], r is known theoretically or empirically. The disadvantage is also reduced as our decentralized algorithm is robust to *imperfect*

estimation of r to some extent, as we shall demonstrate in numerical experiments. For decentralized computation, we prefer the factorization approach over the nuclear-norm approach since the latter needs decentralized SVD, which is expensive.

Prior to introducing the decentralized algorithm, we review algorithm [9] based on nonlinear Gauss-Seidel (GS) iterations applied to (2). At iteration t, the algorithm generates:

$$\mathbf{X}(t+1) = \mathbf{Z}(t)\mathbf{Y}^T(t)(\mathbf{Y}(t)\mathbf{Y}^T(t))^\dagger, \tag{3a}$$

$$\mathbf{Y}(t+1) = (\mathbf{X}^T(t+1)\mathbf{X}(t+1))^\dagger\mathbf{X}^T(t+1)\mathbf{Z}(t), \tag{3b}$$

$$\mathbf{Z}(t+1) = \mathbf{X}(t+1)\mathbf{Y}(t+1) + P_\Omega(\mathbf{W} - \mathbf{X}(t+1)\mathbf{Y}(t+1)), \tag{3c}$$

where $\dagger$ stands for Moore-Penrose pseudo-inverse, and step (3c) effectively equals $Z_{n,m}(t+1) = W_{n,m}$ for $(n,m) \in \Omega$ and $Z_{n,m}(t+1) = (\mathbf{X}(t+1)\mathbf{Y}(t+1))_{n,m}$ for $(n,m) \notin \Omega$.

Since only the product $\mathbf{XY}$, rather than individual $\mathbf{X}$ and $\mathbf{Y}$, is needed, one can simplify the algorithm.

Lemma 1. Replacing the updating rule (3a) by

$$\mathbf{X}(t+1) = c\mathbf{Z}(t)\mathbf{Y}^T(t), \quad c > 0, \tag{4}$$

does *not* change the sequences $\{\mathbf{X}(t)\mathbf{Y}(t)\}$ and $\{\mathbf{Z}(t)\}$.

Convergence of the algorithm is established in [9].

Theorem 1. Let $\{(X(t), Y(t), Z(t))\}$ be a sequence generated by the updating rule (3) (or steps (4), (3b), and (3c)), and $\{P_{\Omega^c}(X(t)Y(t))\}$ be bounded. Then any accumulation point of $\{(X(t), Y(t), Z(t))\}$ is a stationary point of (2).

3. A DECENTRALIZED GS ALGORITHM

3.1. Consensus Optimization Problem

We shall implement the updates (4), (3b), and (3c) in a decentralized way. Our motivations are :

1) Decompose $\mathbf{Z} = [\mathbf{Z}_1, \mathbf{Z}_2, \ldots, \mathbf{Z}_L]$ so that each $\mathbf{Z}_\ell$ has columns corresponding to $\mathbf{W}_\ell$ and $\mathbf{Y}_\ell$. Each agent ℓ holds $\mathbf{Z}_\ell$ and $\mathbf{Y}_\ell$, as well as those entries of $\mathbf{W}_\ell$ in Ω. $\mathbf{Y}$ is the *private matrix*.

2) Each agent keeps a *local copy* $\mathbf{X}^{(\ell)}$ of the *public matrix* $\mathbf{X}$. Presumably, all local copies shall be identical to $\mathbf{X}$ throughout the network, but we relax this and let $\mathbf{X}^{(\ell)}$ be a local estimate.

Keeping these ideas in mind, we check how the centralized algorithm can be transformed to a decentralized one:

1) The updates (3b) and (3c) are naturally decomposable to local updates at the agents. Provided that agent ℓ's local copy $\mathbf{X}^{(\ell)}$ equals $\mathbf{X}$, it only needs to update its private $\mathbf{Z}_\ell$ and $\mathbf{Y}_\ell$ according to (3b) and (3c), independent of other agents.

2) The update (4) mixes the data of all the agents. Set $c = \frac{1}{L}$; (4) becomes $\mathbf{X}(t+1) = \frac{1}{L}\sum_{\ell=1}^L \mathbf{Z}_\ell(t)\mathbf{Y}_\ell^T(t)$, i.e., $\mathbf{X}(t+1)$ is an average of $\mathbf{Z}_\ell(t)\mathbf{Y}_\ell^T(t)$, $\ell = 1, \ldots, L$, where agent ℓ provides $\mathbf{Z}_\ell(t)\mathbf{Y}_\ell^T(t)$.

If we compute identical local copies $\mathbf{X}^{(\ell)}$, then we need

$$\mathbf{X}^{(\ell)}(t+1) = \frac{1}{L}\sum_{\ell=1}^L \mathbf{Z}_\ell(t)\mathbf{Y}_\ell^T(t), \quad \forall \ell = 1, \ldots, L, \tag{5}$$

which is the so-called *average consensus problem*. At iteration t, we formulate the following problem:

$$\begin{aligned} \min_{\{\mathbf{X}^{(\ell)}(t+1)\}} \quad & \tfrac{1}{2}\sum_{\ell=1}^L ||\mathbf{X}^{(\ell)}(t+1) - \mathbf{Z}_\ell(t)\mathbf{Y}_\ell^T(t)||_F^2, \\ \text{s.t.} \quad & \mathbf{X}^{(\ell)}(t+1) = \mathbf{X}^{(j)}(t+1), \quad \forall j \in \mathcal{N}_\ell, \ \forall \ell, \end{aligned} \tag{6}$$

where $\mathcal{N}_\ell$ denotes the set of neighbors of agent ℓ. Apparently, in a connected network, the solution of (6) satisfies (5) and thus (4) for $\mathbf{X}(t+1) \equiv \mathbf{X}^{(\ell)}(t+1)$, $\forall \ell$, and $c = 1/L$. There are various decentralized algorithms for this problem, e.g., the randomized gossip algorithm [10] and the alternating direction method (ADM) based algorithm [11]. Their connection is analyzed in [12]. This paper adopts the ADM approach.

3.2. Exact and Inexact Consensus Optimization

Since it is expensive to solve an instance of (6) at every iteration, we solve it *inexactly* by truncating the ADM iterations.

For each t, to run a total of S sub-iterations for (6), we divide $[t, t+1]$ to S slots. At sub-iteration s, the steps are:

1) Each agent ℓ updates

$$\begin{aligned} \mathbf{X}^{(\ell)}(t + \tfrac{s+1}{S}) \ = \ & \tfrac{\mathbf{Z}_\ell(t)\mathbf{Y}_\ell^T(t) - \boldsymbol{\alpha}^{(\ell)}(t+\frac{s}{S})}{1+2\beta|\mathcal{N}_\ell|} \\ & + \tfrac{\beta|\mathcal{N}_\ell|\mathbf{X}^{(\ell)}(t+\frac{s}{S}) + \beta\sum_{j\in\mathcal{N}_\ell}\mathbf{X}^{(j)}(t+\frac{s}{S})}{1+2\beta|\mathcal{N}_\ell|}, \end{aligned} \tag{7}$$

where $\boldsymbol{\alpha}^{(\ell)} \in \mathcal{R}^{N\times M}$ is the Lagrange multiplier matrix with the initial value $\boldsymbol{\alpha}^{(\ell)}(0) = \mathbf{0}$, $|\mathcal{N}_\ell|$ is the cardinality of $\mathcal{N}_\ell$, and $\beta > 0$ is a constant.

2) Each agent ℓ obtains $\mathbf{X}^{(j)}(t + \frac{s+1}{S})$, $\forall j \in \mathcal{N}_\ell$, and updates

$$\begin{aligned} & \boldsymbol{\alpha}^{(\ell)}(t + \tfrac{s+1}{S}) = \boldsymbol{\alpha}^{(\ell)}(t + \tfrac{s}{S}) \\ & + \beta\left(|\mathcal{N}_\ell|\mathbf{X}^{(\ell)}(t + \tfrac{s+1}{S}) - \textstyle\sum_{j\in\mathcal{N}_\ell}\mathbf{X}^{(j)}(t + \tfrac{s+1}{S})\right). \end{aligned} \tag{8}$$

$\mathbf{X}^{(j)}(t + \frac{s+1}{S})$, $\forall j \in \mathcal{N}_\ell$, obtained in the second step is used in the first step of next sub-iteration.

Convergence for $S \to \infty$ can be found in [11, 12]. However, we found unnecessary to use a large S. A small S leads to inexact $\{\mathbf{X}^{(\ell)}\}$, yet this inexactness does not significantly slow down the overall progress on $\{\mathbf{X}^{(\ell)}\}$, $\mathbf{Y}$, and $\mathbf{Z}$. During the early iterations, exact $\{\mathbf{X}^{(\ell)}\}$ in terms of (5) is better but not much better than an inexact one. Through experiments, we found that running (7) and (8) just once for each t (i.e., setting $S = 1$) gives the lowest total cost.

3.3. Decentralized Matrix Completion

Summarizing the discussions and applying one sub-iteration for (6) at each t, we readily give the proposed decentralized matrix completion algorithm:

Step 1: Initialization. Agent ℓ initializes $\mathbf{X}^{(\ell)}(0)$ and $\mathbf{Y}_\ell(0)$ as random matrices, $\boldsymbol{\alpha}^{(\ell)}(0) = \mathbf{0}$, $\mathbf{Z}_\ell(0) = P_{\Omega_\ell}(\mathbf{W}_\ell)$.

Step 2: Update of $\mathbf{X}$: Each agent ℓ updates

$$\mathbf{X}^{(\ell)}(t+1) = \frac{\mathbf{Z}_\ell(t)\mathbf{Y}_\ell^T(t) - \boldsymbol{\alpha}^{(\ell)}(t)}{1+2\beta|\mathcal{N}_\ell|} + \frac{\beta|\mathcal{N}_\ell|\mathbf{X}^{(\ell)}(t) + \beta\sum_{j\in\mathcal{N}_\ell}\mathbf{X}^{(j)}(t)}{1+2\beta|\mathcal{N}_\ell|}. \quad (9)$$

Step 3: Update of $\boldsymbol{\alpha}$. Each agent ℓ updates

$$\boldsymbol{\alpha}^{(\ell)}(t+1) = \boldsymbol{\alpha}^{(\ell)}(t) + \beta\left(|\mathcal{N}_\ell|\mathbf{X}^{(\ell)}(t+1) - \sum_{j\in\mathcal{N}_\ell}\mathbf{X}^{(j)}(t+1)\right). \quad (10)$$

Step 4: Update of $\mathbf{Y}$. Each agent ℓ updates

$$\mathbf{Y}_\ell(t+1) = ((\mathbf{X}^{(\ell)}(t+1))^T\mathbf{X}^{(\ell)}(t+1))^{-1}(\mathbf{X}^{(\ell)}(t+1))^T\mathbf{Z}_\ell(t). \quad (11)$$

Step 5: Update of $\mathbf{Z}$. Each agent ℓ updates

$$\mathbf{Z}_\ell(t+1) = \mathbf{X}^{(\ell)}(t+1)\mathbf{Y}_\ell(t+1) + P_{\Omega_\ell}(\mathbf{W}_\ell - \mathbf{X}^{(\ell)}(t+1)\mathbf{Y}_\ell(t+1)). \quad (12)$$

Step 6: Iteration. Go to Step 2 until a stopping rule is met.

Remark 1. As in the centralized algorithm in [9], one can apply successive over-relaxation (SOR) extrapolation to accelerate the decentralized algorithm. For each agent ℓ, introduce an extrapolation weight ω_ℓ with initial value 1, compute $\mathbf{Q}_\ell(t) = \omega_\ell(t)\mathbf{Z}_\ell(t) + (1-\omega_\ell(t))\mathbf{X}^{(\ell)}(t)\mathbf{Y}_\ell(t)$ before $\mathbf{X}^{(\ell)}$ is updated at iteration t, and replace $\mathbf{Z}_\ell$ in (9) and (11) by $\mathbf{Q}_\ell$. To update parameter ω_ℓ, we initially choose $\delta > 0$ and $\gamma_{\min} \in (0,1)$ and, at iteration t, let

$$\gamma_\ell(t) = \frac{\|\mathcal{P}_{\Omega_\ell}(X^{(\ell)}(t)Y_\ell(t)) - \mathcal{P}_{\Omega_\ell}(W_\ell)\|_F}{\|\mathcal{P}_{\Omega_\ell}(X^{(\ell)}(t-1)Y_\ell(t-1)) - \mathcal{P}_{\Omega_\ell}(W_\ell)\|_F}.$$

A small $\gamma_\ell(t)$ indicates the current ω_ℓ works well. Hence, we keep the same ω_ℓ if $\gamma_\ell(t) < \gamma_{\min}$; if $\gamma_\ell(t) \in [\gamma_{\min}, 1)$, ω_ℓ is slightly increased to $\omega_\ell + \delta$; otherwise, we reset $\omega_\ell = 1$ if $\gamma_\ell(t) \geq 1$. In our tests, we fix $\delta = 0.1$ and $\gamma_{\min} = 0.7$.

3.4. Privacy Protection and Communication Cost

In some applications of decentralized multi-agent systems, agents may want to avoid sharing original data with others. For our problem, it is possible for each agent ℓ to *keep* $\mathbf{W}_\ell$, $\mathbf{Z}_\ell$, *and* $\mathbf{Y}_\ell$ *private* to certain extent.

Since there is no information exchange at (11) and (12) between different agents, we shall focus on the information exchange in (9) and (10), where each agent ℓ broadcasts $\mathbf{X}^{(\ell)}$ to its neighbors. In light of (9), it is nontrivial for the neighbors to decrypt $\{\mathbf{Z}_\ell(t)\}$ and $\{\mathbf{Y}_\ell^T(t)\}$ from $\{\mathbf{X}^{(\ell)}(t)\}$; this would require to know the $\mathbf{X}^{(j)}$-sum in (9), as well as tracking $\{\boldsymbol{\alpha}^{(\ell)}(t)\}$. The extrapolation in Remark 1 will make the decryption more difficult since the extrapolation weight ω_ℓ is all-time private to agent ℓ and subject to frequent changes. Though we cannot assert full privacy, decryption is certainly a nontrivial task especially when the network topology is complicated.

Communication cost is one of the critical design considerations for networked multi-agent systems. In wired systems, communications occupy network bandwidth; in wireless systems, especially battery-supplied wireless sensor networks, communications consume both bandwidth and energy. For our problem, the only communication at each iteration is that each agent ℓ broadcasts $\mathbf{X}^{(\ell)}$ of size $N \times r$ to its neighbors. Assuming that the iterative algorithm terminates after T iterations, the average communication cost per agent is $T \times N \times r$. A smaller r would lead to less communication. Furthermore, the communication cost is evenly distributed, and no agent communicates much more than others; hence, the network is robust to congestion and incidental failures of some agents.

4. NUMERICAL EXPERIMENTS

We assume that L agents are uniformly randomly deployed in a 100×100 two-dimensional area. Any two agents within a distance of 30 are *directly connected* with bi-directional wireless channels; the resulting communication network is connected. A noise-free data matrix $\mathbf{W}$ of rank K is generated by $\mathbf{W} = \mathbf{U}\mathrm{Diag}(\mathbf{d})\mathbf{V}^T$, where the entries of $\mathbf{U} \in \mathcal{R}^{N\times K}$, $\mathbf{d} \in \mathcal{R}^K$, and $\mathbf{V} \in \mathcal{R}^{M\times K}$ are i.i.d. sampled from the standard normal distribution. We let rank K vary in the experiments. Each of the L agents holds M/L columns of $\mathbf{W}$. The sample set Ω contains randomly chosen $100 \times p$ percentage of the entries of $\mathbf{W}$, so the rest entries need recovery. Regarding unknown matrices $\mathbf{X} \in \mathcal{R}^{N\times r}$ and $\mathbf{Y} \in \mathcal{R}^{r\times M}$, size r may or may not equal K. In the ADM step, $\beta = 1$ is fixed.

The performance is measured in the Frobenius-norm relative error: $||\mathbf{W} - \mathbf{XY}||_F/||\mathbf{W}||_F$. For decentralized algorithms, we get $\mathbf{XY}$ by collecting each product $\mathbf{X}^{(\ell)}\mathbf{Y}_\ell$, i.e., $\mathbf{XY} = [\mathbf{X}^{(1)}\mathbf{Y}_1, \ldots, \mathbf{X}^{(L)}\mathbf{Y}_L]$. In the first set of tests (Fig. 1 and Fig. 2), we set $L = 50$, $N = 40$ and $M = 500$. In the second set of tests (Fig. 3 and Fig. 4), we set $L = 25$, $N = 300$ and $M = 500$.

Test with exact rank estimate $r = K$**.** Consider rank K is known or accurately estimated. For $p = 0.8$ and $K = 4$, the performance of centralized GS (Cen-GS) and decentralized GS (Dec-GS) algorithms is depicted in Fig. 1. Both algorithms converge to the exact recovery at a linear rate. While the centralized one reaches the machine precision within 100 iterations, the decentralized one needs roughly 10 times the iterations for reaching a similar accuracy. This reflects the trade-off in cost between centralized and decentralized computing. The former is faster yet comes with the hidden costs on data collection, for which the knowledge of network topology is also required; the latter is slower, yet skips data collection and protects data privacy.

Test with rank over-estimate $r > K$**.** Over-estimating the rank will reduce the performance of both the centralized GS and decentralized GS algorithms. For the same p but

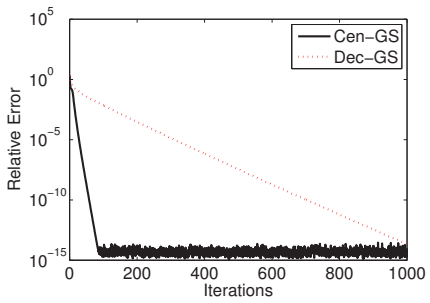

Fig. 1. Cen-GS and Dec-GS with $p = 0.8$ and $r = 4$.

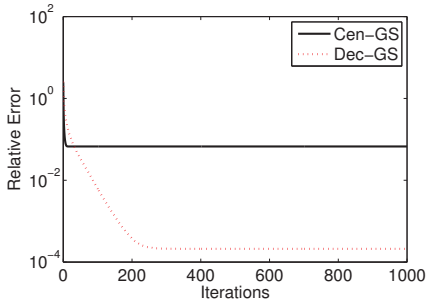

Fig. 2. Cen-GS and Dec-GS with $p = 0.8$ and $r = 6$.

larger $r = 6$, the relative error of both algorithms increase; see Fig. 2. Surprisingly, while the centralized algorithm becomes stagnated, the decentralized algorithm continues to improve and gives more accurate recovery.

SOR acceleration and test of varying r. We compare Cen-SOR (the centralized algorithm with extrapolation) to our decentralized algorithms Dec-GS (without extrapolation) and Dec-SOR (with extrapolation). For fixed $K = 10$, we vary $r = 10, 15, 20$. The performance up to 300 iterations are given in Fig. 3. Dec-SOR is always better that Dec-GS. Cen-SOR is much better with the exact rank and, otherwise, much worse. However, we can introduce a dynamic r-update mechanism to Cen-SOR; the resulting algorithm, known as LMafit in [9], significantly improves the performance. It is our future research to incorporate this in the decentralized algorithm.

Test of varying sample ratios. The larger the rank or the fewer the samples, the more difficult the recovery. For original matrix ranks $K = 10, 20, 30$ each, we test a series of

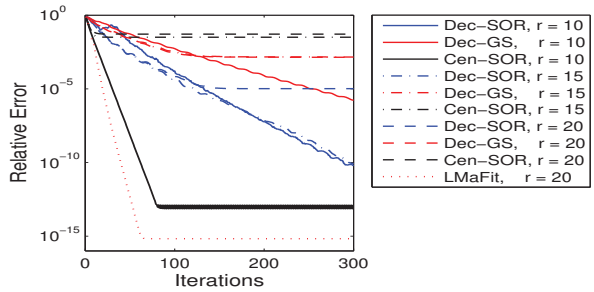

Fig. 3. Dec-GS, Dec-SOR and Cen-SOR results from 50% samples and with $r = 10, 15, 20$ (true rank $K = 10$).

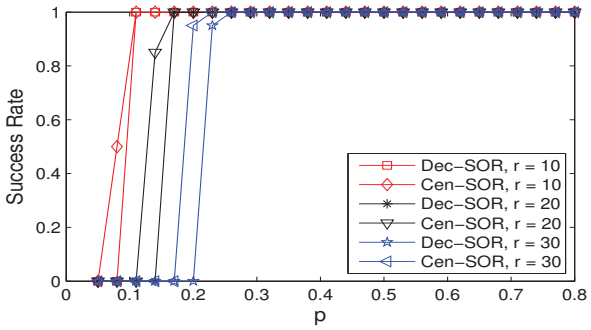

Fig. 4. Recovery success rates of Cen-SOR and Dec-SOR vs sample ratios p at different true ranks $r = K = 10, 20, 30$.

sample ratios $p = 0.05, 0.08, \ldots, 0.8$ and check the chance of acceptable recovery (relative error $\|\mathbf{W} - \mathbf{XY}\|_F / \|\mathbf{W}\|_F < 10^{-3}$) over 20 independent runs. Exact ranks are given to all algorithms. Fig. 4 shows that Dec-SOR is just slightly worse than Cen-SOR. Dec-GS, which is not shown in Fig. 4, performs similar to Dec-SOR.

5. REFERENCES

[1] B. Recht, M. Fazel, and P. A. Parrilo, "Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization," SIAM Review, vol. 52, pp. 471–501, 2010

[2] E. Candes and B. Recht, "Exact matrix completion via convex optimization," Foundations of Computational Mathematics, vol. 9, pp. 717–772, 2009

[3] J. Bennett and S. Lanning, "The Netflix prize," In Proc. of KDDCup, 2007

[4] Y. Zhang, M. Roughan, W. Willinger, and L. Qiu, "Spartio-temporal compressive sensing and Internet traffic matrices," In Proc. of SIGCOMM, 2009

[5] A. Montanari and S. Oh, "On positioning via distributed matrix completion," In Proc. of SAM, 2010

[6] J. Cai, E. Candes, and Z. Shen, "A singular value thresholding algorithm for matrix completion," SIAM J. on Optimization, vol. 20, pp. 1956–1982, 2010

[7] S. Ma, D. Goldfarb, and L. Chen. "Fixed point and Bregman iterative methods for matrix rank minimization," Mathematical Programming Series A, vol. 128, pp. 321–353, 2011

[8] R. Keshavan, A. Montanari, and S. Oh, "Matrix completion from noisy entries," J. of Machine Learning Research, To Appear

[9] Z. Wen, W. Yin, and Y. Zhang, "Solving a low-rank factorization model for matrix completion by a nonlinear successive over-relaxation algorithm," Rice CAAM Tech Report 10-07

[10] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah, "Randomized gossip algorithms," IEEE Trans. on Information Theory, vol. 52, pp. 2508–2530, 2006

[11] J. Bazerque and G. Giannakis, "Distributed spectrum sensing for cognitive radio networks by exploiting sparsity," IEEE Trans. on Signal Processing, vol. 58, pp. 1847–1862, 2010

[12] T. Erseghe, D. Zennaro, E. Dall'Anese, and L. Vangelista, "Fast consensus by the alternating direction multipliers method," IEEE Trans. on Signal Processing, To Appear