Ranking from Crowdsourced Pairwise Comparisons via Smoothed Riemannian Optimization

JIALIN DONG and KAI YANG, ShanghaiTech University and University of Chinese Academy of Sciences
YUANMING SHI, ShanghaiTech University

Social Internet of Things has recently become a promising paradigm for augmenting the capability of humans and devices connected in the networks to provide services. In social Internet of Things network, crowdsourcing that collects the intelligence of the human crowd has served as a powerful tool for data acquisition and distributed computing. To support critical applications (e.g., a recommendation system and assessing the inequality of urban perception), in this article, we shall focus on the collaborative ranking problems for user preference prediction from crowdsourced pairwise comparisons. Based on the Bradley–Terry–Luce (BTL) model, a maximum likelihood estimation (MLE) is proposed via low-rank approach in order to estimate the underlying weight/score matrix, thereby predicting the ranking list for each user. A novel regularized formulation with the smoothed surrogate of elementwise infinity norm is proposed in order to address the unique challenge of the coupled the non-smooth elementwise infinity norm constraint and non-convex low-rank constraint in the MLE problem. We solve the resulting smoothed rank-constrained optimization problem via developing the Riemannian trust-region algorithm on quotient manifolds of fixed-rank matrices, which enjoys the superlinear convergence rate. The admirable performance and algorithmic advantages of the proposed method over the state-of-the-art algorithms are demonstrated via numerical results. Moreover, the proposed method outperforms state-of-the-art algorithms on large collaborative filtering datasets in both success rate of inferring preference and normalized discounted cumulative gain.

CCS Concepts: • **Human-centered computing → Collaborative and social computing**; **Social recommendation**;

Additional Key Words and Phrases: Ranking, pairwise comparison, crowdsourced data, social Internet of Things, low-rank optimization, smoothed matrix manifold optimization

ACM Reference format:
Jialin Dong, Kai Yang, and Yuanming Shi. 2020. Ranking from Crowdsourced Pairwise Comparisons via Smoothed Riemannian Optimization. *ACM Trans. Knowl. Discov. Data* 14, 2, Article 19 (February 2020), 26 pages.
https://doi.org/10.1145/3372407

Part of this work was presented at the IEEE International Conference on Data Mining Workshops (ICDMW), New Orleans, LA, Dec. 2017. This work was supported by the National Nature Science Foundation of China under Grant 61601290.
Authors' addresses: J. Dong, K. Yang, and Y. Shi (corresponding author) are with Shanghai Tech University, Shanghai 201210, China; emails: {dongjl, yangkai, shiym}@shanghaitech.edu.cn.

1556-4681/2020/02-ART19 $15.00
https://doi.org/10.1145/3372407

1 INTRODUCTION

Recent advancements in sensor technologies and communication has facilitated the development of the Internet of Things (IoT) where various devices, e.g., sensors, smartphones, autonomous cars, and the like, are connected to the internet [39]. By leveraging the interaction among the devices in IoT, the social IoT introduces social relationships among devices [44]. In the social IoT, the principles of crowdsourcing have been exploited in solving a series of issues involving data acquisition, processing, and collaborations among participators in a crowdsourcing network [44]. To be specific, crowdsourcing that exploits the collective intelligence of the human crowd holds many advantages such as scalability, mobility, and cost-efficiency and can effectively enhance the intelligence of the services, and improve the interactions between the human and the environment [24]. The advantages of crowdsourcing enable a crowdsourcing solution to support a great number of applications in the social IoT, ranging from social trend understanding and positioning services [22, 47] to smart home, smart city, and smart traffic [25].

In recent years, ranking problems have been considered in order to support a variety of applications such as recommendation system in social IoT networks where the human crowd contribute to data collection and acquisition. Moreover, a growing body of literature has focused on ranking prediction from pairwise comparison [23] with applications including evaluating people's perception of cities from pairwise comparisons of street views of the cities [37], recommendation system, [40] and online sequential survey sampling for quantifying the popularity of proposals among voters [38]. Compared with the conventional numerical measurement considered in the collaborative ranking [4, 28, 43, 45], pairwise comparison measurement has advantages in statistics (e.g., consistency) [46] and manageability (e.g., facilitating ease of data storage and transmission). To be specific, it is complex and costly for users to reflect preference on a single number [46]. Another problem concerning numerical rating is that users are tend to rate high score on already top-ranked item which is known as selection bias. In this article, we are interested in the ranking prediction from crowdsourced pairwise measurements in social IoT. Consider reducing the overhead of data collection, we only collect partial pairwise comparisons.

Collecting data from crowed workers on crowdsourcing platforms such as Amazon Mechanical Turk, Zooinverse, Planet Hunters, and the like. for various applications has recently become prominent. Crowdsourcing has also become a powerful tool to collect pairwise measurements in the social IoT networks in order to serve some applications, i.e., recommendation system and assessing the inequality of urban perception. From the collected pairwise comparisons, the underlying information can be revealed. There are two typical ranking problems: aggregate ranking [18] and collaborative ranking [35] which assume the existence of the underlying preference weight/score vector and matrix respectively. A line of works has focused on recovering matrices or vectors based on various parametric models, e.g., strong stochastic transitivity (SST) model [40] and Bradley–Terry–Luce (BTL) model [3]. Specifically, the class of SST models is defined by assuming the existence of a total ordering of items. In contrast to the SST models, the BTL models assume the existence of "quality" parameter for each item, which facilitates to deal with the heterogeneity of the workers in crowdsourcing system and exploit the diversity of preferences among items [40]. In this article, the pairwise comparison measurements based on the BTL model is exploited in order to support efficient algorithm design. In addition, we assume that the underlying preference weight matrix in the BTL model is a low-rank matrix based on the fact that preferences are only dependent on a few factors [36]. This property increases the possibility of recovering the exact ranking lists from partial pairwise comparisons.

In this article, the maximum likelihood estimation (MLE) approach is exploited to estimate the underlying weight/score matrix based on the BTL model, followed by recovering the individual

rankings from the estimated weight/score matrix. In addition, we introduce the elementwise infinity norm constraint to avoid the excessive "spikiness" of the score matrix. Unfortunately, the resulting MLE problem is a low-rank optimization problem, which is known to be non-convex and NP-hard problem [21]. Fortunately, a flourishing body of works has recently made marvelous progress on the class of provable non-convex optimization problems.

1.1 Related Work

The semidefinite programming (SDP) approach via nuclear norm relaxation for low-rank matrix optimization has been provided rigor theoretical guarantees by a growing body of literature [20, 30, 35]. However, SDP is prohibitive from being adopted to high-dimensional data problem due to the high computational and memory cost. To reduce the computational and storage cost, the Burer and Monteiro heuristic (e.g., [34]) has been proposed that factors $\boldsymbol{X} \in \mathbb{R}^{m\times n}$ as $\boldsymbol{U}\boldsymbol{V}^T$ where $\boldsymbol{U} \in \mathbb{R}^{m\times r}$ and $\boldsymbol{V} \in \mathbb{R}^{n\times r}$. However, the optimization problem is non-convex in $\boldsymbol{U}$ and $\boldsymbol{V}$. Fortunately, recent years significant progress has been made on non-convex paradigms for low-rank optimization problems, which improves the computational efficiency in various important problems, including phase retrieval [17], matrix completion [17], matrix sensing, [34] and community detection [5].

Specifically, for low-rank matrix optimization, a line of literature showed that a series of methods, i.e., gradient descent (GD), stochastic gradient descent (SGD), alternating minimization (AltMin), and block coordinate descent (BCD) methods, yield fast local convergence to global minimum matrix optimization (e.g., [19, 26, 41]) based on a benign initial point. Specifically, the work [41] achieved this goal via showing the analogous strong convexity in the neighborhood of the globally optimal solution. Moreover, the Bi-factor gradient descent (BFGD) algorithm is proposed by the work [33] that converges to the rank-r approximation to the underlying matrix.

Contrast to this line of works, works like [9, 29], [1, Section 7] elude careful initialization and provide theoretical analysis. Specifically, GD converges to the local minimizer with random initialization for strict-saddle function problems where all the saddle points have negative curvature [29]. Similarly, SGD converges to the close neighborhood of the global optimum from random initialization in low-rank matrix recovery problems [9]. All the algorithms above are first-order algorithms which endow with linear convergence rate. Recently, trust-region method [1, Section 7] turns out be a powerful algorithm which is guaranteed to converge to local minimizer at a superlinear convergence rate which enjoys lower iteration complexity than the first-order algorithm does. It has also been applied in problems of synchronization and community detection, where Riemannian trust-region algorithm returns global optima since second-order necessary optimality conditions are sufficient to global optimality in this problem [12]. The remarkable recent results of non-convex paradigms are summarized in Table 1.

Compared with the state-of-the-art algorithms, the Riemannian trust-region method [1, Section 7] enjoys the advantages of initialization robustness, computational efficiency, and fast convergence results. Based on the above consideration, the Riemannian trust-region method is adopted to solve the optimization problem in this article. To achieve this goal, unique challenges for the presented low-rank optimization problems arise due to the additional non-smooth elementwise infinity constraint. the original low-rank MLE problem is thus reformulated as a rank-constrained regularized optimization problem on the manifold. Therein, a smoothed surrogate for the elementwise infinity norm is proposed in the objective function. Compared to the commonly used smoothed surrogate, log-barrier method [8, 16], which calls for solving a sequence of optimization problem with different regularized parameters, the proposed smoothed surrogate yields computational efficiency. Furthermore, by exploiting the quotient manifold geometry of fixed-rank matrices, a scalable Riemannian trust-region algorithm is developed, endowed with superlinear convergence rate.

Table 1. Comparison of Related Works

Initialization	Algorithm	Type of problem	Converge result	Optimality guarantee
good point	GD [19, 41] SGD BCD [41] AltMin [26]	low-rank matrix completion	global optimum	analogous strong convexity in the neighborhood of the globally optimal solution
	BFGD [33]	generic convex objective function	the rank-r approximation to the underlying matrix	N/A
random point	GD [29]	strict-saddle objective function	local minimizer	N/A
	SGD [9]	low-rank matrix completion	the close neighborhood of the global optimum	all local minima are very close to a global optimum
	RTR [1, Section 7]	smooth objective function with compact Riemannian manifold	local minimizer	N/A

1.2 Contributions

The major contributions to the ranking problem from crowdsourced pairwise comparisons is summarized as follows:

(1) The low-rank optimization model with coupled fixed-rank constraint and elementwise infinity norm constraint is presented to recover user individual ranking lists from pairwise measurements.
(2) To address the unique challenge of coupled non-smooth elementwise infinity norm constraint and non-convex fixed-rank constraint, the original problem is reformulated as a rank-constrained smoothed regularized optimization problem with a smoothed surrogate of elementwise infinity norm.
(3) To solve the rank-constrained smoothed optimization problem, we develop the smoothed Riemannian trust-region algorithm via exploiting the geometric structure of fixed-rank matrices, which reduces the computational cost and achieve good performance.

Simulation results on both synthetic data and large collaborative filtering datasets demonstrated that the proposed smoothed regularized approach supported by Riemannian trust-region algorithm has the advantages over the state-of-the-art algorithms in terms of both algorithmic advantages and admirable performance.

1.3 Organization

The remainder of this article is organized as follows. In Section 2, we introduce the system model and problem formulations. A regularized smoothed MLE to estimate the underlying score matrix is developed in Section 3. We further present the matrix optimization over quotient manifold of fixed-rank matrices in Section 4. Numerical results will be demonstrated in Section 5. Finally, we conclude the article and discuss several interesting directions of future work in Section 6. The

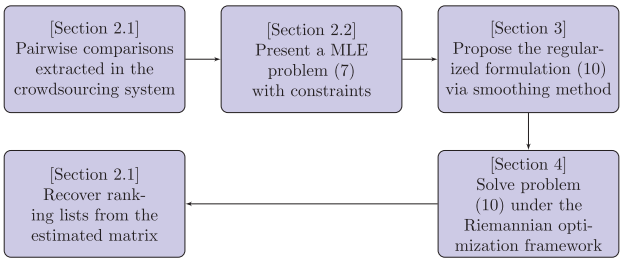


Fig. 1. The proposed framework of approach for ranking from pairwise comparisons in crowdsourcing system.

proof of propositions and details about deriving the related matrix optimization ingredients are illustrated in the appendix. To sum up, the procedure of the proposed approach for ranking from pairwise comparisons in the crowdsourcing system is illustrated in Figure 1.

2 SYSTEM MODEL AND PROBLEM FORMULATION

In this section, we first present the framework of crowdsourcing system. In particular, we consider the recovery of individual rankings from pairwise comparisons which are assumed to be generated according to the BTL model [3, 15]. The individual rankings shall be predicted via estimating the associated weight matrix under the BTL model by using MLE.

2.1 Data Model

Consider a crowdsourcing system that includes m crowd users who provide preferences among n items. We collect pairwise comparisons $\{Y_{ijk} \in \{1,-1\} : (i,j,k) \in \Omega\}$ with $\Omega \subseteq [m] \times [n] \times [n]$ as the observation set, where $[n]$ represents the set $\{1,2,\ldots,n\}$. Here, $Y_{ijk} = 1$ presents that the user i prefers item j to item k, otherwise $Y_{ijk} = -1$. The primary purpose of using pairwise comparisons is to address the inconsistencies among various users [46]. Moreover, the pairwise comparisons can reduce the power-consumption in data management. Note that it is impractical and also not necessary to obtain all pairwise comparisons for the large-scale ranking problem, for which only partial observations are needed [46].

The observation variables are analyzed under the well-known BTL model associated with the logistic distribution [15]. The logistic function is written as

$$f(z) = \frac{1}{1+\exp(\frac{-z}{\sigma})}, \tag{1}$$

where the parameter $\sigma > 0$. Here, the underlying preference score/weight matrix is denoted as $\boldsymbol{X} \in \mathbb{R}^{m\times n}$ and the pairwise comparison measurement between item j and k provided by user i is given by [20]

$$Y_{ijk} = \begin{cases} +1 \text{ with probability } f(\Delta_{ijk}) \\ -1 \text{ with probability } 1 - f(\Delta_{ijk}) \end{cases} \quad \forall\, (i,j,k) \in \Omega, \tag{2}$$

where $\Delta_{ijk} = X_{ij} - X_{ik}$ and the index set of the observed pairwise comparison measurements is denoted as Ω. Here, we assume that the observations are independent with each other.

In this article, we focus on the individual rankings recovery problem. Hence, an associated score $\tau_j^{(i)}$ for each user $i \in [m]$ over the item $j \in [n]$ is introduced as [40]

$$\tau_j^{(i)} := \frac{1}{n}\sum_{k=1}^{n} f(\Delta_{ijk}). \tag{3}$$

The score $\tau_j^{(i)}$ associated with user i denotes the probability that item j is preferred to an item chosen uniformly at random from all n items. We assume that each comparison necessarily results in one winner and the scores $\tau_j^{(i)}$ for $i = 1, \dots m, j = 1, \dots n$ are strictly different almost surely. Furthermore, the ranking list for user i over a set of n items is represented by a mapping $\pi : [n] \to [n]$ given as

$$\tau_{\pi(1)}^{(i)} > \tau_{\pi(2)}^{(i)} > \cdots > \tau_{\pi(n)}^{(i)}, \tag{4}$$

where the k-th ranked item based on the scores derived from (3) is denoted as $\pi(k)$.

In this article, our target is to recover the individual rankings (4) via estimating the weight matrix $\boldsymbol{X}$ from pairwise comparisons [35]. Consider the fact that only a small number of factors affect the preference [2], we assume that the weight matrix is low-rank. Specifically, in a linear factor model, a user's preference vector can be represented by a linear combination of factor vector with corresponding coefficients. Therefore, in a r-factor model, there is a coefficient matrix $\boldsymbol{U} \in \mathbb{R}^{m\times r}$ of which each row represents the proportion of associated factor being considered by each user, and a weight/score matrix $\boldsymbol{V} \in \mathbb{R}^{n\times r}$ of which each row represents weights of each item in terms of various factors. The weight matrix $\boldsymbol{X} \in \mathbb{R}^{m\times n}$ is thus a factorization in the form of $\boldsymbol{U}\boldsymbol{V}^T$ whose rank is at most r [2].

2.2 Maximum-Likelihood Estimation of Weight Matrix

The MLE method is exploited to estimate the weight matrix $\boldsymbol{X}$ from partial pairwise comparisons. In view of the BTL model for the pairwise comparisons (2), the negative log-likelihood function is presented as [20]

$$\begin{aligned}\mathcal{L}_{\Omega,Y}(\boldsymbol{X}) = -\sum_{(i,j,k)\in\Omega} \Big\{ &\mathbb{I}_{(Y_{ijk}=1)} \log(f(\Delta_{ijk})) \\ &+ \mathbb{I}_{(Y_{ijk}=-1)} \log(1 - f(\Delta_{ijk}))\Big\},\end{aligned} \tag{5}$$

where observed pairwise comparisons is denoted as $\boldsymbol{Y} \in \{1, -1\}^{m\times n\times n}$ and $\mathbb{I}_\mu$ is the indicator function, i.e., $\mathbb{I}_\mu = 1$ if the event μ is true, otherwise, $\mathbb{I}_\mu = 0$. To estimate the low-rank score/weight matrix $\boldsymbol{X}$, an optimization problem is proposed that minimizes the negative log-likelihood function with the exact rank constraint:

$$\begin{aligned}\underset{\boldsymbol{X}\in\mathbb{R}^{m\times n}}{\text{minimize}} \quad & \mathcal{L}_{\Omega,Y}(\boldsymbol{X}) \\ \text{subject to} \quad & \text{rank}(\boldsymbol{X}) = r,\end{aligned} \tag{6}$$

where the prior information $r \ll \min\{m, n\}$ denotes the rank of weight matrix. Here, $\mathcal{L}_{\Omega,Y}(\boldsymbol{X})$ can be further written as

$$\mathcal{L}_{\Omega,Y}(\boldsymbol{X}) = -\sum_{(i,j,k)\in\Omega} \log(f(Y_{ijk}(X_{ij} - X_{ik}))). \tag{7}$$

To avoid ill-posedness and the excessive "spikiness" of the matrix of problem (6), the elementwise infinity norm constraint is imposed to bound the magnitude of elements in matrix $\boldsymbol{X}$ [30]. The elementwise infinity norm constraint can be termed as an incoherence requirement to ensure that the estimation matrix is not orthogonal to the observation operator, i.e., the model of generating pairwise comparisons. The optimization problem (6) thus can be reformulate as

$$\mathscr{P}: \underset{\boldsymbol{X}\in\mathbb{R}^{m\times n}}{\text{minimize}} \quad \mathcal{L}_{\Omega,Y}(\boldsymbol{X}) \quad \text{subject to} \quad \operatorname{rank}(\boldsymbol{X}) = r \quad \|\boldsymbol{X}\|_\infty \le \alpha, \tag{8}$$

where an arbitrary reasonable parameter $\alpha > 0$ and $\|\boldsymbol{X}\|_\infty = \max_{i,j}|X_{ij}|$ denotes the elementwise infinity norm. From a sequence of pairwise comparisons, i.e., $Y_{ijk} \in \{1, -1\}$, we estimated the weight matrix $\boldsymbol{X}$ by solving problem $\mathscr{P}$. Then, the ranking list for each user can be recovered based on (3) and (4). However, the fixed-rank constraint makes problem $\mathscr{P}$ (8) non-convex. In this article, we aim at providing efficient algorithms to solve the non-convex optimization problem.

2.3 Problem Analysis

The fixed-rank constraint makes the original problem $\mathscr{P}$ (8) NP-hard. Both non-convex optimization approaches [33] and convex relaxation methods [20] have made remarkable progress to address the NP-hardness. In this subsection, we present the state-of-the-art methods of solving the low-rank optimization problems and present their limitations.

2.3.1 Convex Relaxation Approach. A line of literature [20, 30, 35] exploited the nuclear norm convex relaxation method, yielding the following formulation:

$$\underset{\boldsymbol{X}\in\mathbb{R}^{m\times n}}{\text{minimize}} \quad \mathcal{L}_{\Omega,Y}(\boldsymbol{X}) \quad \text{subject to} \quad \|\boldsymbol{X}\|_* \le \alpha\sqrt{rmn}, \tag{9}$$

where the nuclear norm of $\boldsymbol{X}$, i.e., the summation of singular values of matrix is denoted as $\|\boldsymbol{X}\|_*$. The rank parameter r is sufficiently small, i.e., $r \ll \min(m, n)$. Here, to ensure the elementwise infinity norm constraint in $\mathscr{P}$ (8), we scale the estimated matrix $\boldsymbol{X}$ to $\|\boldsymbol{X}\|_\infty = \alpha$. The computational cost of solving (9) often prohibits the convex relaxation approach to large-dimensional data set. To address this issue, scalable methods, e.g., non-monotone spectral projected-gradient (SPG) method [20], which is based on the projected GD method, have been developed. Specifically, let $\boldsymbol{X} = \boldsymbol{U}\Sigma\boldsymbol{V}^T$ with $\Sigma = \operatorname{diag}(\sigma_1, \ldots, \sigma_2)$. Then the orthogonal projection onto $\{\boldsymbol{X}\colon \|\boldsymbol{X}\|_* \le \alpha\sqrt{rmn}\}$ is provided as [20]

$$\mathcal{P}(\boldsymbol{X}) = \boldsymbol{U}\max\{\Sigma - \lambda\boldsymbol{I}, 0\}\boldsymbol{V}^T, \tag{10}$$

where $\max\{\cdot\}$ is an entry-wise function and $\lambda > 0$ is the smallest value satisfying the constraint $\sum_{i=1}^{d}\max\{\sigma_i - \lambda, 0\} \le \alpha$. The iterate is thus represented as

$$\boldsymbol{X}_{k+1} := \mathcal{P}(\boldsymbol{X}_k - \beta\gamma_k\nabla f(\boldsymbol{X}_k)), \tag{11}$$

where the details for choosing stepsize $\beta \in (0, 1]$ and spectral stepsize γ_k can be found in [20]. However, it is computationally expensive to calculate orthogonal projections via singular value decomposition (SVD) at each iteration. Hence, it is inapplicable to implement such convex paradigms to large-dimensional data.

2.3.2 Non-convex Optimization Paradigms. A line of recent works [5, 33] has developed non-convex optimization algorithms based on the matrix factorization (i.e., factoring $\boldsymbol{X}$ as $\boldsymbol{U}\boldsymbol{V}^T$, where $\boldsymbol{U} \in \mathbb{R}^{m\times r}$ and $\boldsymbol{V} \in \mathbb{R}^{n\times r}$) which enjoys the low computational complexity. Here, log-barrier penalty function [14, 16, Section 11.2] is widely exploited to guarantee the elementwise infinity norm constraint. Hence, problem $\mathscr{P}$ (8) can be transformed to a non-convex optimization problem [8]:

$$\underset{\boldsymbol{U}\in\mathbb{R}^{m\times r},\boldsymbol{V}\in\mathbb{R}^{n\times r}}{\text{minimize}} \mathcal{L}_{\Omega,Y}(\boldsymbol{U}\boldsymbol{V}^T) - \frac{1}{\tau}\sum_{a,b}\log(1 - (\boldsymbol{U}_{a,\cdot}\boldsymbol{V}_{b,\cdot}^T/\alpha)^2), \tag{12}$$

where the a-th row of $\boldsymbol{U}$ is denoted by $\boldsymbol{U}_{a,.}$ and the b-th row of $\boldsymbol{V}$ is denoted by $\boldsymbol{V}_{b,.}$. The tightness of approximation of elementwise infinity norm constraint is determined by the parameter τ. This problem (12) can be solved via the log-barrier method [16] that computes a series of convex problems with the regularized parameter $\tau_0, \mu \cdot \tau_0, \mu^2 \cdot \tau_0, \ldots$ where $\mu > 1$. To be specific, the Bi-factor gradient descent algorithm (BFGD) updates factorizations simultaneously [33]

$$\boldsymbol{U}_{t+1} = \boldsymbol{U}_t - s(\nabla_U F(\boldsymbol{U}_t \boldsymbol{V}_t)), \tag{13}$$

$$\boldsymbol{V}_{t+1} = \boldsymbol{V}_t - s(\nabla_V F(\boldsymbol{U}_t \boldsymbol{V}_t)), \tag{14}$$

where $F(\boldsymbol{X})$ denotes the objective function in (12) and $s > 0$ is the constant stepsize. We scale the estimated matrix $\boldsymbol{X}$ to $\|\boldsymbol{X}\|_\infty = \alpha$. The disadvantages of this methods are that the outer iteration of log-barrier method increases the computational complexity and the first-order method, i.e., BFGD, endows with slow convergence rate.

In this article, the quotient manifold of fixed-rank matrices is exploited to remove the indeterminacy for matrix factorization. The Riemannian trust-region algorithm that enjoys superlinear convergence rate [1, Section 7] is further developed. However, the non-smooth elementwise infinity norm constraint brings extra challenge [1, Section 7]. The procedure of addressing this issue will be introduced in next section.

3 REGULARIZED SMOOTHED MLE FOR SCORE MATRIX ESTIMATION VIA RIEMANNIAN OPTIMIZATION

In this section, to develop more efficient algorithms, we reformulate the low-rank MLE problem (8) to the smoothed regularized optimization problem, thereby developing the matrix manifold optimization by exploiting the geometry of quotient manifold of fixed-rank matrices [32].

3.1 Computational Opportunities via Smoothing Methods

Different from the log-barrier penalty approach [8, 16] to ensure $\|\boldsymbol{X}\|_\infty \le \alpha$, we proposed a smoothed surrogate, of which the advantage will be explicated in the sequel.

Definition 1. [6] Consider a closed and proper convex function g, with $\mathcal{X} \subseteq \operatorname{dom} g$ being a closed convex set and $K \ge 0, \alpha > 0$. The "μ-smooth approximation" of g over $\mathcal{X}$ with parameters (α, β, K), denoted as g_μ, satisfies that

$$\|\nabla g_\mu(\boldsymbol{x}) - \nabla g_\mu(\boldsymbol{y})\|_1 \le \left(K + \frac{\alpha}{\mu}\right) \|\boldsymbol{x} - \boldsymbol{y}\|_\infty, \tag{15}$$

where $\boldsymbol{x}, \boldsymbol{y} \in \mathcal{X}$.

Based on Theorem 4.2 in [6], we propose the smoothed approximation of the non-smooth function $\|\boldsymbol{X}\|_\infty^2$.

Proposition 1. *Given a compact convex set $G_{\boldsymbol{X}} \subseteq \mathbb{R}^{m \times n}$, then the function $p_\mu(\boldsymbol{X}) = \mu \log \sum_{ij} e^{X_{ij}^2/\mu}$ is a μ-smooth approximation of $p(\boldsymbol{X}) = \|\boldsymbol{X}\|_\infty^2$ with parameters $(4M_f^2, \log(mn), 2)$ over $G_{\boldsymbol{X}}$, where $M_f = \max\{\|\boldsymbol{X}\|_\infty : \boldsymbol{X} \in G_{\boldsymbol{X}}\}$.*

Proof. Please refer to Appendix A for details. ■

Based on Proposition 1, $\log \sum_{ij} e^{X_{ij}^2}$ is chosen as the smoothed surrogate of $\|\boldsymbol{X}\|_\infty^2$ in order to guarantee the constraint $\|\boldsymbol{X}\|_\infty^2 \le \alpha^2$ in problem (8). Hence, the regularized smoothed optimization problem can be written as

$$\mathscr{P} : \underset{\boldsymbol{X} \in \mathcal{M}}{\text{minimize}} \ \mathcal{L}_{\Omega, Y}(\boldsymbol{X}) + \lambda \log N(\boldsymbol{X}), \tag{16}$$

where $\lambda = r^2\sqrt{K}\log K$ is a constant regularized parameter to well approximate problem (8) [42] and $N(\boldsymbol{X}) = \sum_{i,j} e^{X_{ij}^2}$. The individual ranking lists can recovered from the estimated matrix based on 4. In addition, we scale the estimated matrix $\boldsymbol{X}$ to $\|\boldsymbol{X}\|_\infty = \alpha$ in order to control the bound on the individual elements. The proposed smoothed approximation is more computationally efficient than the prevalent log-barrier penalty approach [8, 16], since the optimization problem with regularized log-barrier function must be solved by a sequence of problems with different regularized parameters to address the infiniteness near the boundary of feasible set. [14, Section 11.2].

The rank-constrained smoothed optimization formulation with quadratic least-square objective function has been widely investigated in low-rank matrix completion [32], which is endowed with algorithmic advantages by exploiting the geometry of quotient manifold of fixed-rank matrices [1]. However, unique challenges of generalizing geometric concepts in the Euclidean space to the geometric concepts on the quotient manifold of fixed-rank matrices arises due to the complicated structure of the objective function (16). The procedure of addressing this issue will be demonstrated in Section 4.2.

We denote the objective function as $F(\boldsymbol{X})$ in the following discussion. To analyze the geometry of $F(\boldsymbol{X})$, we present the following propositions.

Proposition 2. *The function $F(\boldsymbol{X})$ is smooth (i.e., $F(\boldsymbol{X})$ has Lipschitz gradient and Lipschitz Hessian) and convex over $\{\boldsymbol{X} \in \mathbb{R}^{m\times n} : \|\boldsymbol{X}\|_\infty \le \alpha\}$.*

Proof. Please refer to Appendix B for details. ■

Hence, Proposition 2 demonstrates that the objective function $F(\boldsymbol{X})$ is smooth and convex over the compact convex set in the Euclidean space $\mathbb{R}^{m\times n}$. The smooth convex property of problem (16) paves the way to develop sophisticated algorithms that enjoys a superlinear convergence rate on the manifold $\mathcal{M}$ [1, Section 7.1] which will be explained in the next section.

3.2 Fixed-Rank Matrix Factorization

The primary idea of Riemannian optimization for rank-constrained problem is based on matrix factorization. Three main types of fixed-rank matrix factorization [32] are subspace-projection factorization, polar factorization and balanced factorization. In particular, the balanced factorization is obtained from the SVD, represented as

$$\boldsymbol{X} = (\boldsymbol{U}\Sigma^{\frac{1}{2}})(\Sigma^{\frac{1}{2}}\boldsymbol{V}^T) = \boldsymbol{L}\boldsymbol{R}^T, \tag{17}$$

where $\boldsymbol{L} = \boldsymbol{U}\Sigma^{\frac{1}{2}} \in \mathbb{R}_*^{m\times r}$ and $\boldsymbol{R} = \boldsymbol{V}\Sigma^{\frac{1}{2}} \in \mathbb{R}_*^{n\times r}$ are full-rank matrices [31]. Compared with the other two matrix factorizations, balanced factorization endows with lower-dimensional search space [11]. Moreover, it satisfied the structure of the weight matrix that each row of the coefficient matrix $\boldsymbol{L}$ can represents the extent to which each factor is used and the rows of the factor matrix $\boldsymbol{R}^\top$ are the factors [36]. We thus develop the matrix manifold optimization framework via balanced factorization.

3.3 Quotient Manifold Space

Note that the balanced factorization is not unique based on the fact that $\boldsymbol{X} = \boldsymbol{L}\boldsymbol{M}^{-1}(\boldsymbol{R}\boldsymbol{M}^T)^T = \boldsymbol{L}\boldsymbol{R}^T$ where $\boldsymbol{M} \in \mathrm{GL}(r) = \{\boldsymbol{M} \in \mathbb{R}^{r\times r} : \det(M) \neq 0\}$ [1, 31]. Hence, to address this issue, the *quotient space* $\mathcal{M}/\sim := (\mathbb{R}_*^{m\times r} \times \mathbb{R}_*^{n\times r})/\mathrm{GL}(r)$, where $\mathcal{M} := \mathbb{R}_*^{m\times r} \times \mathbb{R}_*^{n\times r}$ is the *computational space* and $\sim$ represents the equivalence relation, is exploited to represent the search space for problem $\mathscr{P}$. The quotient space describes the set of equivalence classes

$$[(\boldsymbol{L}, \boldsymbol{R})] = \{(\boldsymbol{L}\boldsymbol{M}^{-1}, \boldsymbol{R}\boldsymbol{M}^T) : M \in \mathrm{GL}(r)\}. \tag{18}$$

The quotient manifold plays a vital role in dealing with the unique issue of the matrix factorization. Due to the quotient manifold $\mathcal{M}/\sim$ is an abstract space, the corresponding matrix representations of geometric objects in $\mathcal{M}/\sim$ are required to design algorithms. In addition, the matrix representations can be computed in the computational space $\mathcal{M}$ based on the theory of *Riemannian submersion* [1, Section 3.6.2].

4 MATRIX OPTIMIZATION OVER QUOTIENT MANIFOLDS

In this section, the matrix optimization algorithm is developed over the quotient manifold space endowed with fixed-rank matrices. Specifically, the basic geometric concepts on the quotient manifold and the corresponding matrix representations on the computational space are presented in Section 4.1. Based on the framework of Riemannian optimization, we derive the optimization-related ingredients in Section 4.2, which are utilized to develop Riemannian trust-region method on the quotient manifold space in Section 4.3. Computational complexity will be analyzed in Section 4.4.

4.1 The Framework of Riemannian Optimization

A *Riemannian metric* that characterizes the structure of quotient space on which optimization algorithms is given by [32]

$$g_X(\zeta_X, \xi_X) = \mathrm{Tr}\Big(\big(\boldsymbol{L}^T\boldsymbol{L}\big)^{-1}\zeta_L^T\xi_L\Big) + \mathrm{Tr}\Big(\big(\boldsymbol{R}^T\boldsymbol{R}\big)^{-1}\zeta_R^T\xi_R\Big), \tag{19}$$

where $\zeta_X = (\zeta_L, \zeta_R)$, $\xi_X = (\xi_L, \xi_R) \in T_X\mathcal{M}$ and $\boldsymbol{X} = (\boldsymbol{L}, \boldsymbol{R})$.

In view of the metric (19), the tangent space $T_X\mathcal{M}$ can be decomposed as the sum of two complementary spaces:

$$T_X\mathcal{M} = \mathcal{V}_X\mathcal{M} \oplus \mathcal{H}_X\mathcal{M}, \tag{20}$$

where $\mathcal{V}_X\mathcal{M}$ denotes the *vertical space* and $\mathcal{H}_X\mathcal{M}$ is the *horizontal space*. To be specific, directions of vectors in the horizontal space $\mathcal{H}_X\mathcal{M}$ are orthogonal to the set of equivalence classes $[\boldsymbol{X}]$ (18) and directions of vectors in the vertical space $\mathcal{V}_X\mathcal{M}$ are tangent to the set of equivalence classes (18). Hence, vectors $\xi_X \in \mathcal{H}_X\mathcal{M}$ are invariant along the equivalence class $[\boldsymbol{X}]$ (18). Denote $T_{[X]}(\mathcal{M}/\sim)$ as the tangent space at point $[\boldsymbol{X}]$ on the quotient space $\mathcal{M}/\sim$. There is unique element $\xi_X \in \mathcal{H}_X\mathcal{M}$, called the *horizontal lift* of $\xi_{[X]}$ at $\boldsymbol{X}$, being the matrix representation of $\xi_{[X]} \in T_{[X]}(\mathcal{M}/\sim)$ [1, Section 3.5.8].

Let $\xi_X, \eta_X \in \mathcal{H}_X\mathcal{M}$ be the horizontal lifts of $\xi_{[X]}, \eta_{[X]} \in T_{[X]}(\mathcal{M}/\sim)$ respectively and we define a Riemannian metric on the quotient manifold $\mathcal{M}/\sim$, given by

$$g_{[X]}(\xi_{[X]}, \eta_{[X]}) := g_X(\xi_X, \eta_X). \tag{21}$$

Endowed with this metric, the natural projection $\pi : \mathcal{M} \to \mathcal{M}/\sim$ is *Riemannian submersion* [1, Section 3.6.2] characterizing the relationship between geometrics objects in manifold $\mathcal{M}$ and corresponding ones in quotient manifold $\mathcal{M}/\sim$. Thus, the optimization on abstract $\mathcal{M}/\sim$ can be represented on $\mathcal{M}$. It is known that the cost function $F(\boldsymbol{X})$ is invariable in vertical directions and the search direction is restrict to horizontal directions in the horizontal space $\mathcal{H}_X\mathcal{M}$. Therefore, in the computational space $\mathcal{M}$, the procedure of Riemannian optimization framework can be briefly depicted as detecting the update direction ξ_X on the horizontal space $\mathcal{H}_X\mathcal{M}$. With the calculated descent direction ξ_X, the notion of moving in the direction of ξ_X on the manifold is generalized by mapping $\mathcal{R}_X : \mathcal{H}_X\mathcal{M} \to \mathcal{M}$ called *retraction*. In view of notions above, Algorithm 1 presents the generic matrix manifold optimization algorithm. Moreover, the graphical representation of Algorithm 1 is illustrated in Figure 2.

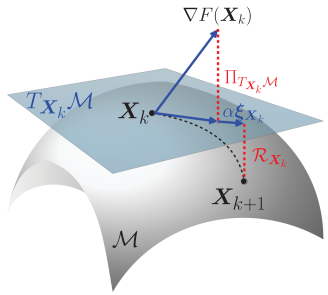


Fig. 2. Graphical representation of the concept of matrix manifold optimization.

ALGORITHM 1: Matrix Manifold Optimization

Given: Riemannian manifold $\mathcal{M}$ with Riemannianmetric g, retraction mapping $\mathcal{R}$, objective function F and the stepsize α.
Output: X_k
1: **Initialize:** initial point $\boldsymbol{X}_0$, $k = 0$
2: **while** not converged **do**
3: Compute a descent direction ξ_k. (e.g., via implementing trust-region method)
4: Update $\boldsymbol{X}_{k+1} = \mathcal{R}_{\boldsymbol{X}_k}(\alpha\xi_k)$
5: $k = k + 1$.
6: **end while**

4.2 Optimization Related Ingredients

In this subsection, the matrix representations of abstract geometrics objects on the quotient manifold is presented in detail to develop Riemannian optimization algorithms. In particular, we mainly contribute to generalize the Euclidean gradient and Euclidean Hessian to the Riemannian gradient and Riemannian Hessian, respectively.

4.2.1 Riemannian Gradient. To implement second-order method, trust region algorithm, on the quotient space $\mathcal{M}/\sim$, the matrix representation (horizontal lift), i.e., $\text{grad}_{\boldsymbol{X}} f \in \mathcal{H}_{\boldsymbol{X}}\mathcal{M}$, of the *Riemannian gradient* $\text{grad}_{[X]} f \in T_{[X]}(\mathcal{M}/\sim)$ is needed, which is derived from the Euclidean gradient of the objective function $F(\boldsymbol{X})$.

We denote the Euclidean partial derivatives as $(\nabla_{\boldsymbol{L}} F(\boldsymbol{X}), \nabla_{\boldsymbol{R}} F(\boldsymbol{X})) \in \mathbb{R}^{m\times r} \times \mathbb{R}^{n\times r}$ and the Riemannian gradient is written as

$$\begin{aligned} \text{grad}_{\boldsymbol{X}} f &= (\text{grad}_{\boldsymbol{L}} f, \text{grad}_{\boldsymbol{R}} f) \\ &= (\nabla_{\boldsymbol{L}} F(\boldsymbol{X})\boldsymbol{L}^T\boldsymbol{L}, \nabla_{\boldsymbol{R}} F(\boldsymbol{X})\boldsymbol{R}^T\boldsymbol{R}), \end{aligned} \tag{22}$$

Please refer to Appendix C for details on the derivation of the Euclidean partial derivatives.

4.2.2 Riemannian Hessian. To employ the trust-region method on the manifold, we further need to define the *Riemannian Hessian* on the tangent space $T_{\boldsymbol{X}}\mathcal{M}$ and project it onto the horizontal space $\mathcal{H}_{\boldsymbol{X}}\mathcal{M}$ in order to compute the matrix representation of the Riemannian Hessian $\text{Hess}_{[X]} f[\xi_X] \in T_{[X]}(\mathcal{M}/\sim)$ on the quotient manifold.

In particular, tangent space projector is required to project the matrix representations of the ambient Euclidean space onto the tangent space $T_{\boldsymbol{X}}\mathcal{M}$. Due to the tangent space of the computational space $\mathcal{M} := \mathbb{R}_*^{m\times r} \times \mathbb{R}_*^{n\times r}$ being $T_{\boldsymbol{X}}\mathcal{M} = (\zeta_L, \xi_R) \in \mathbb{R}^{m\times r} \times \mathbb{R}^{n\times r}$, tangent space projector

Table 2. Optimization-Related Ingredients For Problem $\mathscr{P}$

	$\mathscr{P}$: minimize $\mathcal{L}_{\Omega,Y}(\boldsymbol{L}\boldsymbol{R}^T) + \lambda \log N(\boldsymbol{L}\boldsymbol{R}^T)$
Matrix representation of an element $\boldsymbol{X} \in \mathcal{M}$	$\boldsymbol{X} = (\boldsymbol{L}, \boldsymbol{R})$
Computational space $\mathcal{M}$	$\mathbb{R}_*^{m\times r} \times \mathbb{R}_*^{n\times r}$
Quotient space	$\mathcal{M}/\sim \ := (\mathbb{R}_*^{m\times r} \times \mathbb{R}_*^{n\times r})/\mathrm{GL}(r)$
Metric $g_{\boldsymbol{X}}(\zeta_{\boldsymbol{X}}, \xi_{\boldsymbol{X}})$ for $\zeta_{\boldsymbol{X}}, \xi_{\boldsymbol{X}} \in T_{\boldsymbol{X}}\mathcal{M}$	$g_{\boldsymbol{X}}(\zeta_{\boldsymbol{X}}, \xi_{\boldsymbol{X}}) = \mathrm{Tr}((\boldsymbol{L}^T\boldsymbol{L})^{-1}\zeta_L^T\xi_L) + \mathrm{Tr}((\boldsymbol{R}^T\boldsymbol{R})^{-1}\zeta_R^T\xi_R)$
Riemannian gradient $\mathrm{grad}_{\boldsymbol{X}} f$	$\mathrm{grad}_{\boldsymbol{X}} f = (\mathrm{grad}_{\boldsymbol{L}} f, \mathrm{grad}_{\boldsymbol{R}} f) = (\nabla_{\boldsymbol{L}} F(\boldsymbol{X})\boldsymbol{L}^T\boldsymbol{L}, \nabla_{\boldsymbol{R}} F(\boldsymbol{X})\boldsymbol{R}^T\boldsymbol{R})$
Riemannian Hessian $\mathrm{Hess}_{\boldsymbol{X}} f[\xi_{\boldsymbol{X}}]$	$\mathrm{Hess}_{\boldsymbol{X}} f[\xi_{\boldsymbol{X}}] = \Pi_{\mathcal{H}_{\boldsymbol{X}}\mathcal{M}}(\nabla_{\xi_{\boldsymbol{X}}} \mathrm{grad}_{\boldsymbol{X}} f)$
Retraction $\mathcal{R}_{\boldsymbol{X}} : T_{\boldsymbol{X}}\mathcal{M} \to \mathcal{M}$	$\mathcal{R}_{\boldsymbol{X}}(\xi_{\boldsymbol{X}}) = (\boldsymbol{L} + \xi_L, \boldsymbol{R} + \xi_R)$

$\Pi_{T_{\boldsymbol{X}}\mathcal{M}} : \mathbb{R}^{m\times r} \times \mathbb{R}^{n\times r} \to T_{\boldsymbol{X}}\mathcal{M}$ is given by [31]

$$\Pi_{T_{\boldsymbol{X}}\mathcal{M}}(\boldsymbol{X}) = (\boldsymbol{L}, \boldsymbol{R}). \tag{23}$$

In addition, horizontal space projector is required to project a tangent vector $\boldsymbol{\eta}_{\boldsymbol{X}} \in T_{\boldsymbol{X}}\mathcal{M}$ onto the horizontal space $\mathcal{H}_{\boldsymbol{X}}\mathcal{M}$, which is defined in the sequel.

Proposition 3 (Horizontal Space). *The quotient manifold $\mathcal{M}/\sim$ endowed with the Riemannian metric (21) admits a horizontal space $\mathcal{H}_{\boldsymbol{X}}\mathcal{M} = \{(\zeta_L, \zeta_R) \in \mathbb{R}^{m\times r} \times \mathbb{R}^{n\times r} : \zeta_L^T\boldsymbol{L}\boldsymbol{R}^T\boldsymbol{R} = \boldsymbol{L}^T\boldsymbol{L}\boldsymbol{R}^T\zeta_R\}$, which is the complementary subspace of $\mathcal{V}_{\boldsymbol{X}}\mathcal{M}$ with respect to the Riemannian metric (19), providing the matrix representation of the abstract tangent space $T_{[\boldsymbol{X}]}(\mathcal{M}/\sim)$.*

Proof. Please refer to Appendix E for details. ■

Proposition 4 (Horizontal Space Projection). *The operator $\Pi_{H_{\boldsymbol{X}}\mathcal{M}} : T_{\boldsymbol{X}}\mathcal{M} \to \mathcal{H}_{\boldsymbol{X}}\mathcal{M}$ that projecting vectors on the tangent space onto the horizontal space is called horizontal space projection. It is given as $\Pi_{\mathcal{H}_{\boldsymbol{X}}\mathcal{M}}(\boldsymbol{\eta}_{\boldsymbol{X}}) = (\boldsymbol{\eta}_L + \boldsymbol{L}\boldsymbol{\Lambda}, \boldsymbol{\eta}_R - \boldsymbol{R}\boldsymbol{\Lambda}^T)$, where $\boldsymbol{\Lambda}$ is the solution to the* Lyapunov equation

$$\begin{aligned}\boldsymbol{\Lambda}^T(\boldsymbol{L}^T\boldsymbol{L})(\boldsymbol{R}^T\boldsymbol{R}) + (\boldsymbol{L}^T\boldsymbol{L})(\boldsymbol{R}^T\boldsymbol{R})\boldsymbol{\Lambda}^T \\ = (\boldsymbol{L}^T\boldsymbol{L})\boldsymbol{R}^T\boldsymbol{\eta}_R - \boldsymbol{\eta}_L^T\boldsymbol{L}(\boldsymbol{R}^T\boldsymbol{R}).\end{aligned} \tag{24}$$

Proof. Please refer to Appendix F for details. ■

Propositions 3 characterizes the horizontal space of a Riemannian manifold and demonstrates that there is unique element $\xi_{\boldsymbol{X}} \in \mathcal{H}_{\boldsymbol{X}}\mathcal{M}$, called the *horizontal lift* of $\xi_{[\boldsymbol{X}]}$ at $\boldsymbol{X}$, being the matrix representation of $\xi_{[\boldsymbol{X}]} \in T_{[\boldsymbol{X}]}(\mathcal{M}/\sim)$. It implies that the optimization operations on the abstract tangent space of the quotient manifold space, i.e., $T_{[\boldsymbol{X}]}(\mathcal{M}/\sim)$ can be mapped into the horizontal space of the manifold, i.e., $\mathcal{H}_{\boldsymbol{X}}\mathcal{M}$. Furthermore, Propositions 4 characterizes the horizontal projection that maps the geometric concepts of Euclidean space to the geometric concepts of Riemannian manifold, which facilitates to develop Riemannian optimization algorithms on the manifold. Specifically, the matrix representation of the Riemannian Hessian $\mathrm{Hess}_{[\boldsymbol{X}]} f[\xi_{\boldsymbol{X}}]$ on the Riemannian quotient manifold is given by

$$\mathrm{Hess}_{\boldsymbol{X}} f[\xi_{\boldsymbol{X}}] = \Pi_{\mathcal{H}_{\boldsymbol{X}}\mathcal{M}}(\nabla_{\xi_{\boldsymbol{X}}} \mathrm{grad}_{\boldsymbol{X}} f), \tag{25}$$

where $\mathrm{grad}_{\boldsymbol{X}} f$ (22) is the Riemannian gradient, $\Pi_{\mathcal{H}_{\boldsymbol{X}}\mathcal{M}}$ is the projection operator and $\nabla_{\xi_{\boldsymbol{X}}} \mathrm{grad}_{\boldsymbol{X}} f$ is the *Riemannian connection*. The procedure of deriving the directional derivative of Riemannian gradient is showed in Appendix D.

In summary, the optimization-related ingredients for problem $\mathscr{P}$ are illustrated in Table 2.

4.3 Trust Region Algorithm

In this subsection, based on the matrix manifold optimization framework and the matrix representations mentioned above, we implement the second-order algorithm, i.e., trust-region method, in the computational place $\mathcal{M}$.

Consider a sequence of iterates $\boldsymbol{X}_0, \boldsymbol{X}_1, \ldots$, we assume that the current iterate $\boldsymbol{X}_t \in \mathcal{M}$. The trust-region subproblem is given as

$$\begin{aligned} \underset{\xi_{X_t} \in \mathcal{H}_{X_t}\mathcal{M}}{\text{minimize}} \quad & m(\xi_{X_t}) \\ \text{subject to} \quad & g_{X_t}(\xi_{X_t}, \xi_{X_t}) \leq \delta_t^2, \end{aligned} \tag{26}$$

where δ_t is the trust-region radius in t-th iteration and the cost function is written as

$$\begin{aligned} m(\xi_{X_t}) = F(\boldsymbol{X}_t) &+ g_{X_t}(\xi_{X_t}, \text{grad}_{X_t} f) \\ &+ \frac{1}{2} g_{X_t}(\xi_{X_t}, \text{Hess}_{X_t} f\,[\xi_{X_t}]), \end{aligned} \tag{27}$$

where $\text{Hess}_{X_t} f\,[\xi_{X_t}]$ and $\text{grad}_{X_t} f$ denote the matrix representations of Riemannian Hessian and the Riemannian gradient on the quotient manifold, respectively.

After deriving (approximate) solution ξ_{X_t} of Euclidean trust-region subproblem (26) by using the gradient method [1, Section 7.3], the decisions on whether updating $\boldsymbol{X}_t$ and choosing the new trust-region radius δ_{t+1} are dependent on the quotient [1]

$$\rho_k = \frac{F(\boldsymbol{X}_t) - F(\mathcal{R}_X(\xi_{X_t}))}{m(\boldsymbol{0}_{X_t}) - m(\xi_{X_t})}. \tag{28}$$

If ρ_k is exceedingly small, the trust-region radius should be reduced and keep $\boldsymbol{X}_{k+1}$ unchanged to improve the accuracy of the model. If $\rho_k \gg 1$, even though the model is inaccurate, a significant decrease is being produced during the iteration. Under such condition, the algorithm can expand the trust region radius to examine the exist of a further decrease in the objective value. If ρ_k is proper small, the trust-region radius is maintained and new iterate is updated according to $\boldsymbol{X}_{t+1} = \mathcal{R}_{X_t}(\xi_{X_t})$, where the retraction mapping operator $\mathcal{R}_{X_t} : \mathcal{H}_X\mathcal{M} \to \mathcal{M}$ in each iteration is given by

$$\mathcal{R}_X(\xi_X) = (\boldsymbol{L} + \xi_L, \boldsymbol{R} + \xi_R), \tag{29}$$

where $\xi_X := (\xi_L, \xi_R) \in \mathcal{H}_X\mathcal{M}$ [32].

According to Section 3.1, the objective function in (16) is smooth. With the manifold $\mathcal{M}$ being compact, the Riemannian trust-region algorithm returns the local minimizer with local superlinear convergence rate. More details on the trust region method can refer to [1, Section 7].

4.4 Computational Complexity Analysis

The computational complexity of Algorithm 1 for matrix manifold optimization to minimize the objective function $F(\boldsymbol{X})$ mainly depends on the computational cost of the optimization-related ingredients showed in Table 2, which is demonstrated below.

(1) Computing Riemannian metric (19): $O(mr^2 + nr^2)$.
(2) Computing Riemannian gradient: $O(|\Omega| + mnr + mn + mr^2 + nr^2)$.
(3) Computing the projection operator showed in Proposition (3): $O(mr^2 + nr^2)$.
(4) Computing of Riemannian Hessian (25): $O(|\Omega| + mnr + mn + mr^2 + nr^2)$.

Here, $|\Omega|$ denotes the sampling size. Therefore, the computational cost of per iteration in Algorithm 1 is approximately linear with $|\Omega|$. Compared to the convex relaxation approach solved by

SDP which endows with the computational complexity $O(\max(m,n)n^2)$ [14], the proposed algorithm has much lower computational complexity, i.e., $O(|\Omega| + mnr)$. Note that, compared to the first-order algorithms, the proposed algorithm has remarkable advantage on the superlinear convergence rate. Endowed with such fast convergence rate, the computational complexity of the proposed algorithm is no greater than the one of the first-order algorithms. Thus the proposed algorithm has the notable advantage over the first-order algorithms.

5 NUMERICAL RESULTS

In this section, we present the simulation results of the proposed smoothed matrix manifold optimization algorithm for recovering individual rankings from pairwise comparisons in order to demonstrate the advantages over state-of-the-art algorithms.

5.1 Simulation Settings and Performance Metric

We run the simulations under the following settings. Simulation results are further evaluated with well-defined performance metric. The simulation settings are presented as follows:

(1) Weight matrix $\boldsymbol{X}^*$: The weight/score matrix is generated as $\boldsymbol{X}^* = \boldsymbol{U}\boldsymbol{V}^T$, where $\boldsymbol{U}, \boldsymbol{V} \in \mathbb{R}^{K\times r}$ have i.i.d. entries uniformly chosen from $[-0.5, 0.5]$. Matrix $\boldsymbol{X}^*$ is scaled to $\|\boldsymbol{X}^*\|_\infty = 1$.
(2) Pairwise comparisons Y_{ijk}: We generate the pairwise comparisons from the BTL model with the underlying weight matrix $\boldsymbol{X}^*$ and $\sigma = 0.18$ [20].
(3) Observation/sampling set Ω: Given the sample size of Ω as $|\Omega|$, $|\Omega|$ independent observations is chosen uniformly at random.
(4) Performance metric:
   — Relative mean square error: After scaling the estimated weight matrix $\hat{\boldsymbol{X}}$ such that $\|\hat{\boldsymbol{X}}\|_\infty = 1$, we exploit the relative mean square error (MSE) to assess the performance of weight matrix estimation [35]

$$\text{err}(\hat{\boldsymbol{X}}) = \|\hat{\boldsymbol{X}} - \boldsymbol{X}^*\|_F^2 / \|\boldsymbol{X}^*\|_F^2.$$

   — Success rate: It equals to the number of pairwise comparisons in right orders over the total number of pairwise comparisons [35], which is given by

$$\text{Success rate} = \frac{1}{|\mathcal{S}|} \sum_{(i,j,k)\in\mathcal{S}, Y_{ijk}=1} \mathbb{I}_{\hat{X}_{ij} > \hat{X}_{ik}},$$

   where $\mathbb{I}_\mu$ is the indicator function and $\mathcal{S}$ denotes the test set.
   — Normalized discounted cumulative gain (NDCG)@K: It is the ranking measure for numerical rating, defined as [35]

$$\text{NDCG@}K(i) = \frac{\text{DCG@}K(i, \pi_i)}{\text{DCG@}K(i, \pi_i^*)},$$

   where

$$\text{DCG@}K(i, \pi_i) = \sum_{k=1}^{K} \frac{2^{R^*_{i\pi_i(k)}} - 1}{\log_2(k+1)},$$

   and $\pi_i(k)$ is the index of the k-th ranked item of the test set of i-th user. R^*_{ij} is the true rating of item j given by user i in the datasets and π_i^* maximize the function $\text{DCG@}K(i, \pi_i)$. Note that, the individual ranking lists, i.e., (4) are recovered via computing the associated score, i.e., (3).

We compare three algorithms on synthetic data, described as:

—**Proposed Riemannian trust-region algorithm solving log-sum-exp regularized problem (PRTRS)**: The algorithm is implemented by *Manopt* [13] where the fixed-rank manifold implementation of `fixedrankfactory_2factors` and the solver of `trustregions` are chosen to solve the problem (16) with default option.

—**Bi-factor gradient descent solving log-barrier regularized problem (BFGDB)**: Problem (12) is solved by the algorithm [33]. The regularization term coefficient τ_t is set to $\mu \cdot \tau_{t-1}$ during t-th outer iteration of the algorithm [8, 16]. The Bi-factor gradient descent with the constant stepsize, i.e., $s := \frac{2}{187}\{\frac{1}{\|U_0\|_F^2}, \frac{1}{\|V_0\|_F^2}\}$ [33] is implemented in the inner iteration.

—**Spectral projected-gradient (SPG)**: In this algorithm [20], codes provided in [20] is exploited to solve the problem (9) with setting the elementwise infinity norm constraint coefficient α equal to $\|X^*\|_\infty$ [35].

Set $K = m = n$ for all experiments. The algorithms are initialized with U_0, V_0 whose entries are i.i.d. and drawn form the standard normal distribution, which are scaled to $\|X_0\|_\infty = 0.95$ with $X_0 = U_0 V_0^T$. The PRTRS algorithm terminate either the norm of Riemannian gradient $\|\text{grad}_{X_t} f\| < 10^{-6}$ or the number of iterations exceeding 500. The stopping situation for inner iteration of the BFGDB algorithm is the same as [33] and it is also ended when $\|X\|_\infty \geq 1$. In terms of the outer iteration of BFGDB, the regularization term coefficient τ_1 is chosen as $mn/\mathcal{L}_{\Omega,Y}(X_0)$ and the number of outer iterations is given as

$$\left\lceil \frac{\log mn - \log \eta - \log \tau_1}{\log \mu} \right\rceil \tag{30}$$

where $\eta = 10^{-3}$ and $\mu = 2$ [8, 16]. The setting for SPG algorithm is the same as [20]. The algorithms are carried out under Matlab environment on a desktop computer with an Intel i7-6500U CPU with 64GB memory.

We further compare our algorithm with AltSVM [35] and SGD [10] on large collaborative filtering datasets, i.e., MovieLens100k and MovieLens1m. The settings for AltSVM and SGD algorithm are the same as the code in *https://github.com/dhpark22/collranking* which are carried out under C++ environment on a desktop computer with an Intel i7-6500U CPU with 64 GB memory.

5.2 Convergence Rate of Normalized Objective Function

Consider the setting of $K = 200$ and the sampling size being $(drK\log K)$, where the rescaled sample size is $d = 15$ and the rank of weight matrix is $r = 15$ [30], Figure 3 illustrates the convergence rate of normalized objective functions of different algorithms. The normalized objective function in t-th iteration is represented as $Q(x^t)/Q(x^*)$, where $Q(x)$ denotes the objective function and x^* is the converge result of the algorithm. It shows that PRTRS has faster convergence rate of normalized objective Function than both BFGDB algorithm and SPG algorithm.

5.3 Relative MSE with Different Sample Sizes

Under the setting of $K = 200$ and $r = 10$, we simulate with different sample sizes $(drK \log K)$ [30]. We conduct numerical experiments averaged over 100 realizations to compare these three algorithms with stopping criterion described in Section 5.1. Figure 4 illustrates the relative MSE corresponding to different rescaled sample sizes d. We can see that all the algorithms perform better as the rescaled sample size d increases. Furthermore, PRTRS achieves better performance in terms of MSE than both SPG and BFGDB algorithm.

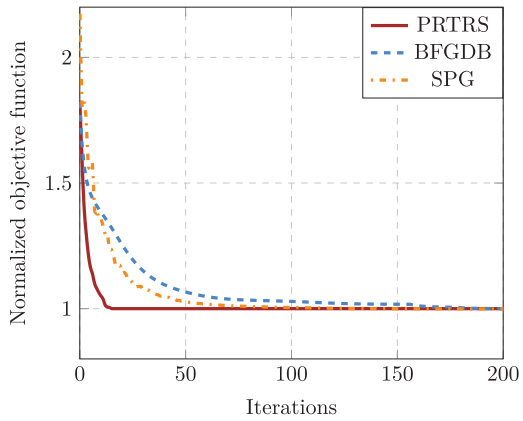


Fig. 3. Convergence rate of the normalized objective functions of different algorithms.

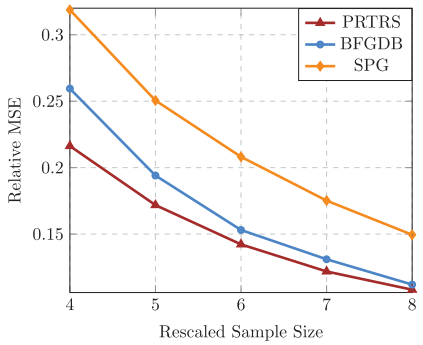


Fig. 4. Relative MSE with different sample sizes d.

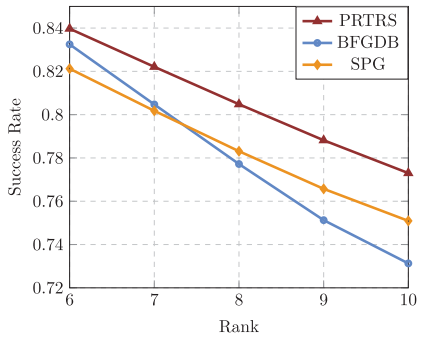
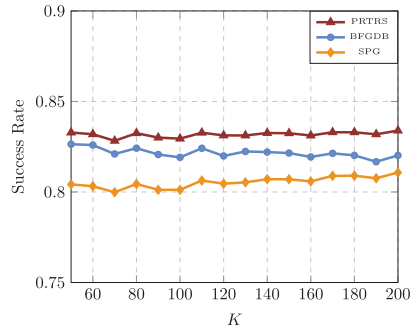


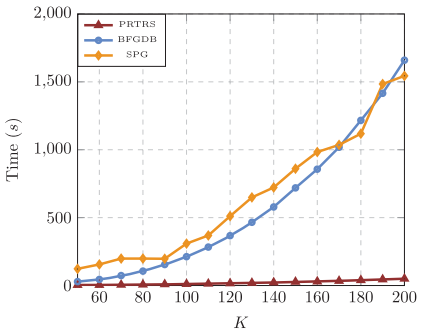
Fig. 5. Success rate with different ranks r.

5.4 Success Rate with Different Ranks

Consider the situation that fixed parameters are $|\Omega| = 2 \times 10^4$ and $K = 200$, we simulate the algorithms with different ranks r of the underlying weight matrix and averaged over 100 realizations. Here, the rank of the manifold in the optimization is set to r. The success rate of ranking recovery is shown in Figure 5. It demonstrates that the proposed algorithm achieves higher success rate than both SPG and BFGDB, especially at higher rank. Moreover, for all algorithms, we can see



(a) success rate with different sizes



(b) computational time with different sizes

Fig. 6. Success rate and corresponding computation time with different sizes.

that the lower rank is, the higher success rate can be achieved, which implies that fewer factors considered in the linear factor model make it easier to recover unobserved data under the same set of observations.

5.5 Computation Time with Different Problem Sizes

With fixed parameters as $r = 10$ and $d = 5$, we conduct the numerical experiments averaged over 100 realizations to simulate three algorithms with different sizes K. The two figures in Figure 6 show the success rate and corresponding computational time with different sizes K respectively. It implies the dramatical advantage in computational time of the proposed PRTRS over other algorithms while achieving better performance. The low computational time of PRTRS is facilitated by the low computational complexity and iteration complexity. To be specific, the proposed algorithm is a second-order algorithm developed on the Riemannian algorithm which enjoys a superlinear convergence rate [1, Section 7.1]. Thus, the proposed algorithm enjoys lower iteration complexity than the first-order algorithms, i,e., SPG [20] and BFGDB [33], which has linear convergence rate. Furthermore, since only second-order directional derivative is required for developing the Riemannian trust-region algorithm, the computational complexity of the proposed algorithm is as lower as the one of first-order algorithm which requires to compute the gradient of the objective function that is $O(|\Omega| + mnr)$.

5.6 Experiments on Practical Datasets

Now we demonstrate that our algorithm has good performance on both pairwise data and rating data in practical datasets. We compared our algorithm with SGD [10] and AltSVM [35] on the

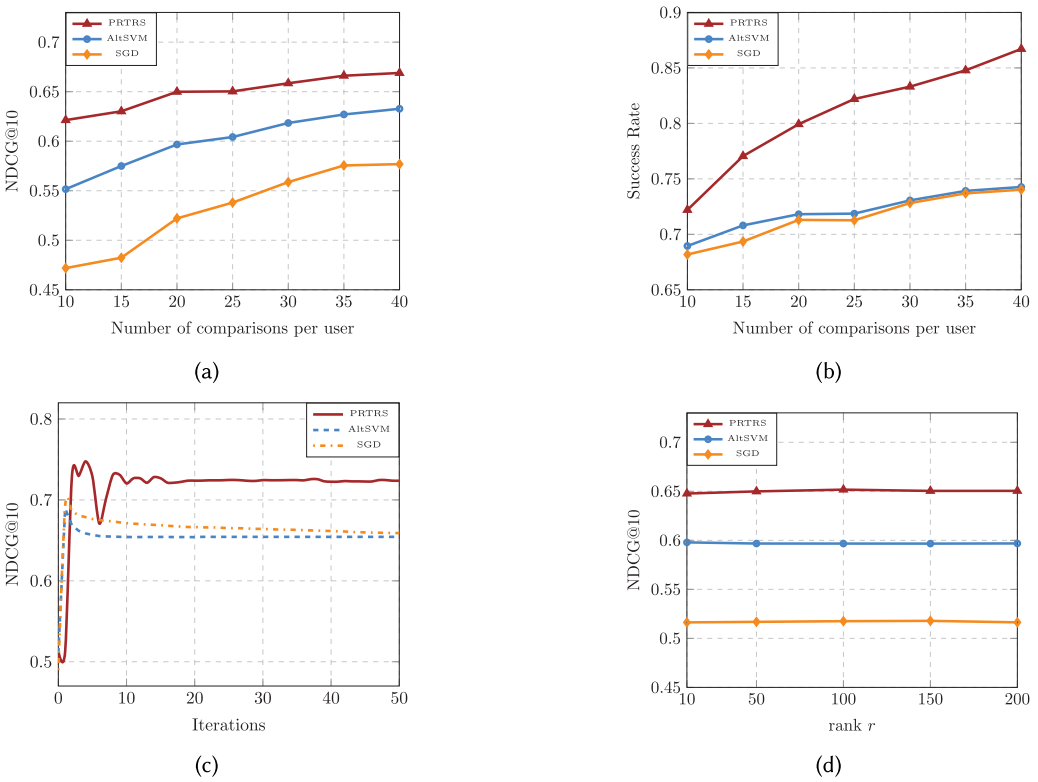


Fig. 7. Numerical experiments on practical datasets.

MovieLens 100k dataset, which contains 100,000 ratings given by 943 users on 1,682 movies. The training/test data Y are generated by converting the integral ratings (i.e., from one to five) to pairwise comparisons. For each user, we subsampled N comparisons as training data while the rest of the comparisons as test data. We dropped out the users with less than $N + 10$ comparisons. The rank r is chosen as 50. In Figure 7(a), we evaluate the performance of different algorithms via NDCG@10. Under the same setting, we further evaluate the performance of our algorithm by computing success rate, illustrated in Figure 7(b). From the above simulations, it turns out that our algorithm outperforms state-of-the-art algorithms in terms of both the ranking measure for numerical ratings and the measure for inferring preferences.

We now show the performance of our algorithm on the larger practical datasets. The simulations are based on the MovieLens 1m dataset, which contains 1,000,209 ratings given by 6,040 users on 3,900 movies. For each user, we subsampled 25 comparisons for training and took the rest of ratings for test. The rank r is chosen as 100. Figure 7(c) shows that our algorithm can converge to better result than state-of-the-art algorithms in a few iterations.

We further demonstrate that the choice of rank parameter r has slight influence on the performance of our algorithm when dealing with practical datasets. We used the MovieLens 100k dataset, therein 20 comparisons are subsampled for each user. We compare the above three algorithms, i.e., PRTRS, AltSVM, and SGD in terms of different rank parameters r. Figure 7(d) shows that the performance of our algorithm is robust to the rank r when dealing with practical datasets.

In summary, simulations show that the proposed Riemannian trust-region algorithm has remarkable advantages over the BFGD and SPG algorithm in terms of algorithmic advantages

(i.e., computational time and rate of convergence) and admirable performance (i.e., success rate and MSE). The superior performance of the proposed algorithm is mainly facilitated by the geometry of quotient manifold space that address the ununique issue of the balanced factorization and the trust-region algorithm that enjoys superlinear convergence rate. Simulations on large collaborative filtering datasets further demonstrate the advantages of our proposed algorithm over AltSVM, and SGD in dealing with both pairwise data and numerical data.

6 CONCLUSIONS

In this article, a low-rank optimization problem was presented to recover the individual ranking lists from crowdsourced pairwise comparisons in social IoT networks. To address the unique challenge of the coupled rank constraint and elementwise infinity norm constraint in the low-rank estimation formulation, a smoothed surrogate of elementwise infinity norm was further proposed to smooth the objective function. In addition, we introduced an optimization framework of Riemannian optimization via generalizing the Euclidean geometric concepts to the geometric concepts on the quotient manifold space of fixed-rank matrices. Based on this framework, we developed a Riemannian trust-region algorithm that enjoys the superlinear rate of convergence. Numerical results showed that the proposed algorithm remarkably surpasses the state-of-the-art algorithms in estimation performance improvements and algorithmic advantages.

APPENDICES

A PROOF OF PROPOSITION 1

To proof that function $p_\mu(\boldsymbol{X})$ is the smoothed surrogate of $p(\boldsymbol{X})$, we first present the smoothed surrogate of function $d(\boldsymbol{X}) = \max_{i,j} X_{ij}$ in the space $\mathbb{R}^{m\times n}$ based on the definitions (e.g., inner product, conjugate function and dual norm) represented in [27]. We then finish the proof based on the results in [6]. According to discussions above, the Lamma is represented in the following:

Lemma 1. *The conjugate function of* $\omega(\boldsymbol{X}) = \log(\sum_{i,j} e^{X_{ij}})$ *is represented as* $\omega^*(\boldsymbol{U}) = \sum_{i,j} U_{ij} \log U_{ij}$ *with domain* $\omega^* = \{\boldsymbol{U} \in \mathbb{R}^{m\times n} : \sum_{ij} U_{ij} = 1, U_{ij} \geq 0\}$, *which is a 1-strongly convex function with respect to the* $\|\cdot\|_1$ *norm. Note that* $\|\boldsymbol{A}\|_1 = \sum_{i,j} |A_{ij}|$.

Proof. It is obvious to generalize the conclusion in $\mathbb{R}^n$ [7] to the space $\mathbb{R}^{m\times n}$. ■

Based on Lemma 1 and the theorem on strong/smooth duality in [27], we conclude that the function $\omega = \omega^{**}$ is 1-strongly smooth with respect to the elementwise infinity norm $\|\cdot\|_\infty$. Moreover, it is easy to check that [6], $\forall\, \mu > 0$ and $\boldsymbol{X} \in \mathbb{R}^{m\times n}$, there is

$$d(\boldsymbol{X}) \leq \mu\omega\left(\frac{\boldsymbol{X}}{\mu}\right), \tag{31}$$

and $\omega(\boldsymbol{0}) = \log(mn)$. Thus, according to Theorem 4.2 in [6], $d_\mu(\boldsymbol{X}) = \mu\omega(\frac{\boldsymbol{X}}{\mu}) = \mu \log(\sum_{i,j} e^{X_{ij}/\mu})$ is a μ-smooth approximation of $d(\boldsymbol{X})$ with parameters $(1, \log(mn), 0)$.

Proof of Proposition 1: Let $c : \mathbb{R}^{m\times n} \to \mathbb{R}^{m\times n}$ $c(\boldsymbol{X}) = \boldsymbol{X} \circ \boldsymbol{X}$, where "$\circ$" denotes the Hadamard product operation, e.g., elementwise product. Since $d_\mu(\boldsymbol{X})$ is a $(1, \log(mn), 0)$-smooth approximation of $d(\boldsymbol{X})$ over $\mathbb{R}^{m\times n}$, according to definition of smoothable function [6], it follows that there is a decomposition $\log(mn) = \beta_1 + \beta_2$, for which [6]

$$d(\boldsymbol{Z}) - \beta_1\mu \leq d_\mu(\boldsymbol{Z}) \leq d(\boldsymbol{Z}) + \beta_2\mu \text{ for every } \boldsymbol{Z} \in \mathbb{R}^{m\times n}. \tag{32}$$

Let $Z = c(X)$ such that $p(X) = d(c(X))$ and $p_\mu(X) = d_\mu(c(X))$, then for every $X \in G_X$, we obtain that

$$p(X) - \beta_1\mu \le p_\mu(X) \le p(X) + \beta_2\mu, \tag{33}$$

which preserves the property of smoothable function [6] with $\beta = \log(mn)$. To derive the parameters for smooth approximation, we first introduce the elementwise gradient matrix of function p_μ by the chain rule, given as

$$\nabla p_\mu(X) = 2X \circ \nabla d_\mu(c(X)). \tag{34}$$

Based on definition of smoothable function [6], for all $X, U \in G_X$, we let

$$D_\mu(X, U) = \nabla d_\mu(c(X)) - \nabla d_\mu(c(U)), \tag{35}$$
$$\Delta_{X,U} = X - U, \tag{36}$$
$$\Delta_c = c(X) - c(U), \tag{37}$$

then there is

$$\begin{aligned}
&\|\nabla p_\mu(X) - \nabla p_\mu(U)\|_1 \\
\le &\|2X \circ D_\mu(X, U)\|_1 + \|2\Delta_{X,U} \circ \nabla d_\mu(c(U))\|_1 \\
\le &\|2X\|_\infty \cdot \|D_\mu(X, U)\|_1 + \|2\Delta_{X,U}\|_\infty \cdot \|\nabla d_\mu(c(U))\|_1 \\
\overset{*}{\le} &\frac{2}{\mu}\|X\|_\infty \cdot \|\Delta_c\|_\infty + 2\|\Delta_{X,U}\|_\infty \cdot \|\nabla d_\mu(c(U))\|_1,
\end{aligned} \tag{38}$$

where the inequality $(*)$ follows that $d_\mu(X)$ has a Lipschitz gradient with constant $\frac{1}{\mu}$. To present the inequality (38) in detail, for $\forall\, Z \in \mathbb{R}^{m\times n}$, we have

$$\|\nabla d_\mu(Z)\|_1 = \sum_{i,j}\left|\frac{e^{Z_{ij}/\mu}}{\sum_{i,j} e^{Z_{ij}/\mu}}\right| = 1, \tag{39}$$

$$\|c(X) - c(U)\|_\infty \le 2M_f\|X - U\|_\infty, \tag{40}$$

where $M_f = \max\{\|X\|_\infty : X \in G_X\}$. Based on the formulation (39), (40), we conclude that

$$\|\nabla p_\mu(X) - \nabla p_\mu(U)\|_1 \le \left(\frac{4}{\mu}M_f^2 + 2\right)\|X - U\|_\infty, \tag{41}$$

which implies that $p_\mu(X) = d_\mu(c(X)) = \mu\log\sum_{ij} e^{X_{ij}^2/\mu}$ is a μ-smooth approximation of $p(X) = d(c(X)) = \|X\|_\infty^2$ over G_X with parameters $(4M_f^2, \log(mn), 2)$. ■

B PROOF OF PROPOSITION 2

We have already known $\log N(X)$ is convex and endows with Lipschitz gradient from Appendix A. Based on the notions mentioned in Appendix A, we further proof it has Lipschitz Hessian as well.

Proof of Proposition 2: The directional derivative of $\nabla p_\mu(X)$ in the direction of $\xi \in \mathbb{R}^{m\times n}$ is derived as

$$\nabla^2 p_\mu(X)[\xi] = 2\xi \circ \nabla d_\mu(c(X)) + 4X^{\circ 2} \circ \xi \circ \nabla^2 d_\mu(c(X)), \tag{42}$$

where $\nabla^2 d_\mu(\boldsymbol{Z}) = \frac{e^{\boldsymbol{Z}/\mu}}{\mu \sum_{i,j} e^{Z_{ij}/\mu}}$. Let

$$D^2_\mu(\boldsymbol{X}, \boldsymbol{U}) = \xi \circ (\nabla^2 d_\mu(c(\boldsymbol{X})) - \nabla^2 d_\mu(c(\boldsymbol{U}))), \tag{43}$$

$$\Delta^2_{\boldsymbol{X},\boldsymbol{U}} = \xi \circ \Delta_c. \tag{44}$$

Thus, based on the equations (35) and (37), there is

$$\begin{aligned}
&\|\nabla^2 p_\mu(\boldsymbol{X})[\xi] - \nabla^2 p_\mu(\boldsymbol{U})[\xi]\|_1 \\
&= \|2\xi \circ D_\mu(\boldsymbol{X}, \boldsymbol{U}) + 4\boldsymbol{X}^{\circ 2} \circ D^2_\mu(\boldsymbol{X}) + 4\Delta^2_{\boldsymbol{X},\boldsymbol{U}} \circ \nabla^2 d_\mu(c(\boldsymbol{U}))\|_1 \\
&\leq \|2\xi\|_\infty \cdot \|D_\mu(\boldsymbol{X}, \boldsymbol{U})\|_1 + \|4\boldsymbol{X}^{\circ 2}\|_\infty \cdot \left\|D^2_\mu(\boldsymbol{X}, \boldsymbol{U})\right\|_1 + \Gamma \\
&\overset{*}{\leq} 2\left(\frac{1}{\mu} + \frac{2\|\boldsymbol{X}\|^2_\infty}{\mu^2}\right)\|\xi\|_\infty \cdot \|\Delta_c\|_\infty + \Gamma,
\end{aligned} \tag{45}$$

where $\Gamma = \|4\Delta^2_{\boldsymbol{X},\boldsymbol{U}}\|_\infty \cdot \|\nabla^2 d_\mu(c(\boldsymbol{U}))\|_1$. The inequality $(*)$ follows that $d_\mu(\boldsymbol{X})$ has a Lipschitz gradient with constant $\frac{1}{\mu}$ and $\nabla d_\mu(\boldsymbol{X})$ has a Lipschitz gradient with constant $\frac{1}{\mu^2}$. Moreover, there is

$$\|\nabla^2 d_\mu(\boldsymbol{Z})\|_1 = \sum_{i,j}\left|\frac{e^{Z_{ij}/\mu}}{\mu \sum_{i,j} e^{Z_{ij}/\mu}}\right| = \frac{1}{\mu}. \tag{46}$$

Thus, based on the formulation (40) and (46), we claim that

$$\begin{aligned}
&\|\nabla^2 p_\mu(\boldsymbol{X})[\xi] - \nabla^2 p_\mu(\boldsymbol{U})[\xi]\|_1 \\
&= \left(\frac{12}{\mu} + \frac{8M^2_f}{\mu^2}\right) M_f \|\xi\|_\infty \cdot \|\boldsymbol{X}\|_\infty,
\end{aligned} \tag{47}$$

which demonstrates the directional derivative of function $\nabla p_\mu(\boldsymbol{X})$ is Lipschitz continuous with respect to elementwise infinity norm. To sum up, the function $\log N(\boldsymbol{X})$ is smooth endowing with Lipschitz gradient and Lipschitz Hessian. As for the smoothness and convexity of the function $\mathcal{L}_{\Omega,Y}(\boldsymbol{X})$, it will be illustrated in the following.

Based on the results in following Appendix C and D, we derived the Euclidean gradient (53) and the Euclidean directional Hessian (57) of function $\mathcal{L}_{\Omega,Y}(\boldsymbol{X})$ (7). Define the inner product as $\langle \boldsymbol{X}, \boldsymbol{Y}\rangle := \mathrm{Tr}(\boldsymbol{X}^T\boldsymbol{Y}) = \mathrm{Tr}(\boldsymbol{Y}^T\boldsymbol{X})$. Therefore, with $\forall\, \xi \in \mathbb{R}^{m\times n}$, it is easy to check that

$$\begin{aligned}
&\langle \xi, \nabla^2 \mathcal{L}_{\Omega,Y}(\boldsymbol{X})[\xi]\rangle \\
&= \sum_{(i,j,k)\in\Omega} \mathrm{Tr}\Big(M_{ijk}(\xi) h(M_{ijk}(\boldsymbol{X})) \boldsymbol{A}^T_{ijk} \cdot \xi\Big) \\
&= \sum_{(i,j,k)\in\Omega} M_{ijk}(\xi) h(M_{ijk}(\boldsymbol{X})) \cdot Y_{ijk}(\xi_{ij} - \xi_{ik}) \\
&= \sum_{(i,j,k)\in\Omega} h(M_{ijk}(\boldsymbol{X}))(\xi_{ij} - \xi_{ik})^2 \geq 0,
\end{aligned} \tag{48}$$

which implies $\mathcal{L}_{\Omega,Y}(\boldsymbol{X})$ is a convex function.

For two matrices $\boldsymbol{X}, \boldsymbol{U} \in \{\boldsymbol{X} \in \mathbb{R}^{m\times n} : \|\boldsymbol{X}\|_\infty \le \alpha\}$, there is

$$\begin{aligned}
&\|\nabla \mathcal{L}_{\Omega,Y}(\boldsymbol{X}) - \nabla \mathcal{L}_{\Omega,Y}(\boldsymbol{U})\|_1 \\
&= \left\| \sum_{(i,j,k)\in\Omega} g(M_{ijk}(\boldsymbol{X}))M'_{ijk}(\boldsymbol{X}) - g(M_{ijk}(\boldsymbol{U}))M'_{ijk}(\boldsymbol{U}) \right\|_1 \\
&\le 02 \sum_{(i,j,k)\in\Omega} e^{Y_{ijk}(U_{ij}-U_{ik})} |e^{Y_{ijk}(X_{ij}-U_{ij}-X_{ik}+U_{ik})} - 1| \\
&\le \frac{2|\Omega|e^{2\alpha}(e^{2\|\boldsymbol{X}-\boldsymbol{U}\|_\infty} - 1)}{\|\boldsymbol{X}-\boldsymbol{U}\|_\infty} \|\boldsymbol{X}-\boldsymbol{U}\|_\infty \\
&\overset{(*)}{\le} \frac{|\Omega|e^{2\alpha}(e^{4\alpha}-1)}{\alpha} \|\boldsymbol{X}-\boldsymbol{U}\|_\infty,
\end{aligned} \tag{49}$$

where $|\Omega|$ denotes the sampling size. The inequality $(*)$ follows that $m(x) = \frac{e^{2x}-1}{x}$ is a monotonically increasing function as $x \in [-\alpha, \alpha], \alpha > 0$. The inequality (49) demonstrates the derivative of function $\mathcal{L}_{\Omega,Y}(\boldsymbol{X})$ is Lipschitz continuous with respect to elementwise infinity norm.

Let $\boldsymbol{\xi} \in \mathbb{R}^{m\times n}$ denote the directional matrix and let $v_x = X_{ij} - X_{ik}, v_u = U_{ij} - U_{ik}$, then we have

$$\begin{aligned}
&\|\nabla^2 \mathcal{L}_{\Omega,Y}(\boldsymbol{X})[\boldsymbol{\xi}] - \nabla^2 \mathcal{L}_{\Omega,Y}(\boldsymbol{U})[\boldsymbol{\xi}]\|_1 \\
&\overset{(*)}{\le} 2 \sum_{(i,j,k)\in\Omega} \left| M_{ijk}(\boldsymbol{\xi}) \left[e^{Y_{ijk}v_u}(\varrho - 1) + \psi(\varrho^{-2} - 1) \right] \right| \\
&\le 2 \sum_{(i,j,k)\in\Omega} |M_{ijk}(\boldsymbol{\xi})| \left[e^{Y_{ijk}v_u}|\varrho - 1| + \psi|\varrho^{-2} - 1| \right] \\
&\le 4|\Omega| \cdot \|\boldsymbol{\xi}\|_\infty \left[e^{2\alpha}|\varrho - 1| + e^{6\alpha}|\varrho^{-2} - 1| \right] \\
&\overset{(**)}{\le} \frac{2|\Omega|(e^{2\alpha}(e^{4\alpha}-1) + e^{6\alpha}(e^{8\alpha}-1))}{\alpha} \|\boldsymbol{X}-\boldsymbol{U}\|_\infty \cdot \|\boldsymbol{\xi}\|_\infty,
\end{aligned} \tag{50}$$

where $\varrho = e^{Y_{ijk}(v_x - v_u)}$ and $\psi = e^{Y_{ijk}(2v_x + v_u)}$. The inequalities (*) and (**) follow the analogous procedure as the third and last step in (49) respectively. The inequality (50) demonstrates the directional derivative of function $\nabla \mathcal{L}_{\Omega,Y}(\boldsymbol{X})$ is Lipschitz continuous with respect to elementwise infinity norm. To sum up, $\mathcal{L}_{\Omega,Y}(\boldsymbol{X})$ is a convex and smooth function endowed with Lipschitz gradient and Lipschitz Hessian over $\{\boldsymbol{X} \in \mathbb{R}^{m\times n} : \|\boldsymbol{X}\|_\infty \le \alpha\}$ in the Euclidean space [27]. ■

C COMPUTING THE EUCLIDEAN PARTIAL DERIVATIVES

Let $M_{ijk}(\boldsymbol{X})$ denote $Y_{ijk}(X_{ij} - X_{ik})$, which is a linear function of $\boldsymbol{X}$. The derivative of M_{ijk} with respect to the matrix $\boldsymbol{X}$ is

$$M'_{ijk}(\boldsymbol{X}) = Y_{ijk}\boldsymbol{\delta}_{ij} - Y_{ijk}\boldsymbol{\delta}_{ik}, \tag{51}$$

where $\boldsymbol{\delta}_{ij}$ is a $m \times n$ matrix with $[\boldsymbol{\delta}_{ij}]_{ij} = 1$ and being zeros otherwise. In addition, the derivative of $N(\boldsymbol{X})$ is given by

$$N'(\boldsymbol{X}) = 2\boldsymbol{X} \circ \exp(\boldsymbol{X}^{\circ 2}), \tag{52}$$

where '$\circ$' denotes the Hadamard product operation, i.e., elementwise product/power. We derive the Euclidean derivative of the function $\mathcal{L}_{\Omega,Y}(\boldsymbol{X})$ (7) as

$$\nabla \mathcal{L}_{\Omega,Y}(\boldsymbol{X}) = \sum_{(i,j,k)\in\Omega} g(M_{ijk}(\boldsymbol{X}))M'_{ijk}(\boldsymbol{X}), \tag{53}$$

where $g(x) = -\frac{1}{1+e^x}$. Hence, the Euclidean derivative of the objective function $F(\boldsymbol{X})$ is written as

$$\nabla F(\boldsymbol{X}) = \nabla \mathcal{L}_{\Omega,Y}(\boldsymbol{X}) + \lambda\, Q(\boldsymbol{X}), \tag{54}$$

where $Q(\boldsymbol{X}) = N'(\boldsymbol{X})/N(\boldsymbol{X})$. Then we have

$$\nabla_L f(\boldsymbol{X}) = \nabla F(\boldsymbol{X}) \cdot \boldsymbol{R},\ \nabla_R f(\boldsymbol{X}) = \nabla F(\boldsymbol{X})^T \cdot \boldsymbol{L}. \tag{55}$$

D COMPUTING THE DIRECTIONAL DERIVATIVE OF RIEMANNIAN GRADIENT

Based on the results presented in Appendix C, we now derive the directional derivative of Riemannian gradient. The directional derivative of $Q(\boldsymbol{X})$ in the direction of $\xi_X := (\xi_L, \xi_R) \in \mathcal{H}_X\mathcal{M}$ is written as

$$\begin{aligned}\nabla Q(\boldsymbol{X})[\xi_X] =& \frac{1}{N^2(\boldsymbol{X})}[N(\boldsymbol{X})(2e^{\boldsymbol{X}^{\circ 2}} + 2\boldsymbol{X} \circ \boldsymbol{N}'(\boldsymbol{X})) \circ \boldsymbol{K} \\ & - \boldsymbol{N}'(\boldsymbol{X}) \sum_{ij} (\boldsymbol{N}'(\boldsymbol{X}) \circ \boldsymbol{K})_{ij}],\end{aligned} \tag{56}$$

where $\boldsymbol{K} = \xi_L \boldsymbol{R}^T + \boldsymbol{L}\xi_R^T$.

Given i, j, k and $M'_{ijk}(\boldsymbol{X})$ that is a constant matrix replaced by $\boldsymbol{A}_{ijk}$, then the directional derivative of Euclidean gradient $\nabla \mathcal{L}_{\Omega,Y}(\boldsymbol{X})$ in the direction of $\xi \in \mathbb{R}^{m\times n}$ is derived as

$$\nabla^2 \mathcal{L}_{\Omega,Y}(\boldsymbol{X})[\xi] = \sum_{(i,j,k)\in\Omega} M_{ijk}(\xi) h(M_{ijk}(\boldsymbol{X}))\boldsymbol{A}_{ijk}, \tag{57}$$

where $h(x) = \frac{e^x}{(1+e^x)^2}$. Thus the directional derivative of Euclidean gradient $\nabla F(\boldsymbol{X})$ in the direction of $\xi_X = (\xi_L, \xi_R)$ is denoted as

$$\begin{aligned}&\nabla^2 F(\boldsymbol{X})[\xi_X] \\ &= \sum_{(i,j,k)\in\Omega} \left[M_{ijk}(\xi_L \boldsymbol{R}^T) + M_{ijk}\left(\boldsymbol{L}\xi_R^T\right)\right] \cdot h(M_{ijk}(\boldsymbol{X}))\boldsymbol{A}_{ijk}, \\ &\quad + \lambda \nabla Q(\boldsymbol{X})[\xi_X].\end{aligned} \tag{58}$$

From (55) and (58), we can derive the directional second-order partial derivatives of $f(\boldsymbol{X})$ in the direction of ξ_X as

$$\nabla_L^2 f(\boldsymbol{X})[\xi_X] = \nabla^2 F(\boldsymbol{X})[\xi_X] \cdot \boldsymbol{R} + \nabla F(\boldsymbol{X}) \cdot \xi_R, \tag{59}$$

$$\nabla_R^2 f(\boldsymbol{X})[\xi_X] = \nabla^2 F(\boldsymbol{X})[\xi_X]^T \cdot \boldsymbol{L} + \nabla F(\boldsymbol{X})^T \cdot \xi_L. \tag{60}$$

Thus, the directional derivative of Riemannian Gradient is written as [32]

$$\nabla_{\xi_X} \mathrm{grad}_{\boldsymbol{X}} f[\xi_X] = (\nabla_{\xi_L} \mathrm{grad}_L f[\xi_L], \nabla_{\xi_R} \mathrm{grad}_R f[\xi_R]) \tag{61}$$

where

$$\begin{aligned}\nabla_{\xi_L} \mathrm{grad}_{\boldsymbol{L}} f[\xi_L] =& \nabla_L^2 f(\boldsymbol{X})[\xi_X] \cdot \boldsymbol{L}^T\boldsymbol{L} \\ & + 2\nabla_L f(\boldsymbol{X}) \cdot \mathrm{Sym}\left(\xi_L^T \boldsymbol{L}\right)\end{aligned} \tag{62}$$

$$\begin{aligned}\nabla_{\xi_R} \mathrm{grad}_{\boldsymbol{R}} f[\xi_R] =& \nabla_R^2 f(\boldsymbol{X})[\xi_X] \cdot \boldsymbol{R}^T\boldsymbol{R} \\ & + 2\nabla_R f(\boldsymbol{X}) \cdot \mathrm{Sym}\left(\xi_R^T \boldsymbol{R}\right)\end{aligned} \tag{63}$$

E PROOF OF PROPOSITION 3: HORIZONTAL SPACE

The vertical space $\mathcal{V}_X\mathcal{M}$ is defined as the tangent space to the equivalence class. Based on the vertical space for the Grassmann manifold [1, Example 3.6.4], we derive the matrix representation for the vertical space as

$$\mathcal{V}_X\mathcal{M} = \{(-\boldsymbol{L}\boldsymbol{\Lambda}, \boldsymbol{R}\boldsymbol{\Lambda}^T) : \boldsymbol{\Lambda} \in \mathbb{R}^{r\times r}\}. \tag{64}$$

We define $\boldsymbol{\zeta}_X = (\boldsymbol{\zeta}_L, \boldsymbol{\zeta}_R) \in \mathcal{H}_X\mathcal{M}$ and $\boldsymbol{\xi}_X = (\boldsymbol{\xi}_L, \boldsymbol{\xi}_R) \in \mathcal{V}_X\mathcal{M}$. As horizontal space $\mathcal{H}_X\mathcal{M}$ is the complementary subspace of $\mathcal{V}_X\mathcal{M}$ with respect to the Riemannian metric (19), $\boldsymbol{\zeta}_X$ is orthogonal to $\boldsymbol{\xi}_X$ in the sense of Riemannian metric g_X (19), i.e.,

$$\mathrm{Tr}\Big((\boldsymbol{L}^T\boldsymbol{L})^{-1}\boldsymbol{\zeta}_L^T\boldsymbol{\xi}_L\Big) + \mathrm{Tr}\Big((\boldsymbol{R}^T\boldsymbol{R})^{-1}\boldsymbol{\zeta}_R^T\boldsymbol{\xi}_R\Big) = 0. \tag{65}$$

After simplifying the equation (65) by exploiting the properties of trace, we derive the horizontal space as

$$\mathcal{H}_X\mathcal{M} = \Big\{(\boldsymbol{\zeta}_L, \boldsymbol{\zeta}_R) \in \mathbb{R}^{m\times r} \times \mathbb{R}^{n\times r} : \boldsymbol{\zeta}_L^T\boldsymbol{L}\boldsymbol{R}^T\boldsymbol{R} = \boldsymbol{L}^T\boldsymbol{L}\boldsymbol{R}^T\boldsymbol{\zeta}_R\Big\}. \tag{66}$$

F PROOF OF PROPOSITION 4: HORIZONTAL SPACE PROJECTION

The horizontal space projector $\Pi_{\mathcal{H}_X\mathcal{M}}$ is the operator of extracting the horizontal component of the tangent vector. Based on the relationship between tangent space and two complementary spaces (20), the expression for the projector $\Pi_{\mathcal{H}_X\mathcal{M}}$ is

$$\Pi_{\mathcal{H}_X\mathcal{M}}(\boldsymbol{\eta}_X) = (\boldsymbol{\eta}_L + \boldsymbol{L}\boldsymbol{\Lambda}, \boldsymbol{\eta}_R - \boldsymbol{R}\boldsymbol{\Lambda}^T) = (\boldsymbol{\zeta}_L, \boldsymbol{\zeta}_R) \tag{67}$$

where $\boldsymbol{\eta}_X = (\boldsymbol{\eta}_L, \boldsymbol{\eta}_R) \in T_X\mathcal{M}$ and $\boldsymbol{\zeta}_X = (\boldsymbol{\zeta}_L, \boldsymbol{\zeta}_R) \in \mathcal{H}_X\mathcal{M}$. As $\boldsymbol{\zeta}_X$ belongs to the horizontal space (66), we have

$$\boldsymbol{\zeta}_L^T\boldsymbol{L}\boldsymbol{R}^T\boldsymbol{R} = \boldsymbol{L}^T\boldsymbol{L}\boldsymbol{R}^T\boldsymbol{\zeta}_R, \tag{68}$$

$$(\boldsymbol{\eta}_L + \boldsymbol{L}\boldsymbol{\Lambda})^T\boldsymbol{L}\boldsymbol{R}^T\boldsymbol{R} = \boldsymbol{L}^T\boldsymbol{L}\boldsymbol{R}^T(\boldsymbol{\eta}_R - \boldsymbol{R}\boldsymbol{\Lambda}^T), \tag{69}$$

which can be rewritten as the equation (24).

REFERENCES

[1] P. A. Absil, R. Mahony, and R. Sepulchre. 2009. *Optimization Algorithms on Matrix Manifolds*. Princeton University Press.

[2] Alekh Agarwal, Sahand Negahban, and Martin J. Wainwright. 2012. Fast global convergence of gradient methods for high-dimensional statistical recovery. *Ann. Stat.* 40, 5 (Oct. 2012), 2452–2482.

[3] Mine Alsan, Ranjitha Prasad, and Vincent Y. F. Tan. 2018. Lower bounds on the bayes risk of the Bayesian BTL model with applications to comparison graphs. *IEEE J. Sel. Topics Signal Process.* 12, 5 (Apr. 2018), 975–988.

[4] Suhrid Balakrishnan and Sumit Chopra. 2012. Collaborative ranking. In *Proc. ACM Int. Conference on Web Search and Data Mining (WSDM'12)*. ACM, 143–152.

[5] Afonso S. Bandeira, Nicolas Boumal, and Vladislav Voroninski. 2016. On the low-rank approach for semidefinite programs arising in synchronization and community detection. In *Proc. Conf. Learn. Theory (COLT'16)*. 23–26.

[6] Amir Beck and Marc Teboulle. 2012. Smoothing and first order methods: A unified framework. *SIAM J. Optim.* 22, 2 (Jun. 2012), 557–580.

[7] Amir Beck and Marc Teboulle. 2003. Mirror descent and nonlinear projected subgradient methods for convex optimization. *Oper. Res. Lett.* 31, 3 (May. 2003), 167–175.

[8] Sonia A. Bhaskar and Adel Javanmard. 2015. 1-bit matrix completion under exact low-rank constraint. In *In Proc. IEEE Conf. Inform. Science and Systems (CISS'15)*. IEEE, 1–6.

[9] Srinadh Bhojanapalli, Behnam Neyshabur, and Nati Srebro. 2016. Global optimality of local search for low rank matrix recovery. In *Adv. Neural. Inf. Process. Syst. (NIPS'16)*. 3873–3881.

[10] Léon Bottou. 2010. Large-scale machine learning with stochastic gradient descent. In *Proc. Int. Conf. Comput. Stat.*Springer, 177–186.

[11] Nicolas Boumal and Pierre-Antoine Absil. 2011. RTRMC: A Riemannian trust-region method for low-rank matrix completion. In *Adv. Neural. Inf. Process. Syst. (NIPS'11)*. 406–414.
[12] Nicolas Boumal, P.-A. Absil, and Coralia Cartis. 2018. Global rates of convergence for nonconvex optimization on manifolds. *IMA J Numer. Anal.* 39, 1 (Feb 2018), 1–33.
[13] Nicolas Boumal, Bamdev Mishra, P.-A. Absil, and Rodolphe Sepulchre. 2014. Manopt, a Matlab toolbox for optimization on manifolds. *J. Mach. Learn. Res.* 15 (Jan. 2014), 1455–1459. Retrieved from http://www.manopt.org.
[14] Stephen Boyd and Lieven Vandenberghe. 2004. *Convex Optimization*. Cambridge University Press.
[15] Ralph Allan Bradley and Milton E. Terry. 1952. Rank analysis of incomplete block designs: I. the method of paired comparisons. *Biometrika* 39, 3-4 (Dec. 1952), 324–345.
[16] Emmanuel Candes and Justin Romberg. 2005. ℓ_1-magic: Recovery of sparse signals via convex programming. *California Inst. Technol., Pasadena, CA.*
[17] Yudong Chen and Yuejie Chi. 2018. Harnessing structures in big data via guaranteed low-rank matrix estimation: Recent theory and fast algorithms via convex and nonconvex optimization. *IEEE Signal Process. Mag.* 35, 4 (Jul. 2018), 14–31.
[18] Yuxin Chen and Changho Suh. 2015. Spectral MLE: Top-k rank aggregation from pairwise comparisons. In *Proc. Int. Conf. Mach. Learn. (ICML'15)*. 371–380.
[19] Yudong Chen and Martin J. Wainwright. 2015. Fast low-rank estimation by projected gradient descent: General statistical and algorithmic guarantees. *arXiv preprint arXiv:1509.03025* (2015).
[20] Mark A. Davenport, Yaniv Plan, Ewout van den Berg, and Mary Wootters. 2014. 1-bit matrix completion. *Inf. Inference* 3, 3 (Sep. 2014), 189–223.
[21] Mark A. Davenport and Justin Romberg. 2016. An overview of low-rank matrix recovery from incomplete observations. *IEEE J. Sel. Topics Signal Process.* 10, 4 (Jan. 2016), 608–622.
[22] Wei Feng, Zheng Yan, Hengrun Zhang, Kai Zeng, Yu Xiao, and Y. Thomas Hou. 2018. A survey on security, privacy, and trust in mobile crowdsourcing. *IEEE Internet Things J.* 5, 4 (Aug. 2018), 2971–2992.
[23] Reinhard Heckel, Max Simchowitz, Kannan Ramchandran, and Martin J. Wainwright. 2018. Approximate ranking from pairwise comparisons. In *International Conference on Artificial Intelligence and Statistics*. 1057–1066.
[24] Jia Hu, Hui Lin, Xuancheng Guo, and Ji Yang. 2018. DTCS: An integrated strategy for enhancing data trustworthiness in mobile crowdsourcing. *IEEE Internet Things J.* 5, 6 (Feb. 2018), 4663–4671.
[25] Luis G. Jaimes, Idalides J. Vergara-Laurens, and Andrew Raij. 2015. A survey of incentive techniques for mobile crowd sensing. *IEEE Internet Things J.* 2, 5 (Oct. 2015), 370–380.
[26] Prateek Jain, Praneeth Netrapalli, and Sujay Sanghavi. 2013. Low-rank matrix completion using alternating minimization. In *ACM Symp. Theory Comput.* ACM, 665–674.
[27] Sham Kakade, Shai Shalev-Shwartz, and Ambuj Tewari. 2009. On the duality of strong convexity and strong smoothness: Learning applications and matrix regularization. *Unpublished Manuscript* (2009).
[28] Joonseok Lee, Samy Bengio, Seungyeon Kim, Guy Lebanon, and Yoram Singer. 2014. Local collaborative ranking. In *Proc. Int. Conf. World Wide Web*. ACM, 85–96.
[29] Jason D. Lee, Max Simchowitz, Michael I. Jordan, and Benjamin Recht. 2016. Gradient descent only converges to minimizers. In *29th Annu. Proc. Conf. Learn. Theory (COLT'16)*. 1246–1257.
[30] Yu Lu and Sahand N. Negahban. 2015. Individualized rank aggregation using nuclear norm regularization. In *Proc. 53rd Annu. Allerton Conf. Commun., Control, Comput.* IEEE, 1473–1479.
[31] Gilles Meyer, Silvere Bonnabel, and Rodolphe Sepulchre. 2011. Linear regression under fixed-rank constraints: A Riemannian approach. In *Proc. Int. Conf. Mach. Learn. (ICML'11)*.
[32] Bamdev Mishra, Gilles Meyer, Silv Bonnabel, and Rodolphe Sepulchre. 2014. Fixed-rank matrix factorizations and Riemannian low-rank optimization. *Comput. Statist.* 29, 3–4 (Jun. 2014), 591–621.
[33] Dohyung Park, Anastasios Kyrillidis, Constantine Caramanis, and Sujay Sanghavi. 2018. Finding low-rank solutions via non-convex matrix factorization, efficiently and provably. *SIAM J. Imaging Sci.* 11, 4 (Oct. 2018), 2165–2204.
[34] Dohyung Park, Anastasios Kyrillidis, Constantine Caramanis, and Sujay Sanghavi. 2017. Non-square matrix sensing without spurious local minima via the Burer-Monteiro approach. In *Proc. Int. Conf. Artificial Intelligence and Statistics (AISTATS'17)*.
[35] Dohyung Park, Joe Neeman, Jin Zhang, Sujay Sanghavi, and Inderjit S. Dhillon. 2015. Preference completion: Large-scale collaborative ranking from pairwise comparisons. In *Proc. Int. Conf. Mach. Learn. (ICML'15)*. 1907–1916.
[36] Jasson D. M. Rennie and Nathan Srebro. 2005. Fast maximum margin matrix factorization for collaborative prediction. In *Proc. Int. Conf. Mach. Learn. (ICML'05)*. 713–719.
[37] Philip Salesses, Katja Schechtner, and César A. Hidalgo. 2013. The collaborative image of the city: Mapping the inequality of urban perception. *PloS One* 8, 7 (Jun. 2013), e68400.
[38] Matthew J. Salganik and Karen E. C. Levy. 2015. Wiki surveys: Open and quantifiable social data collection. *PloS One* 10, 5 (May. 2015), e0123483.

[39] Omer Berat Sezer, Erdogan Dogdu, and Ahmet Murat Ozbayoglu. 2018. Context-aware computing, learning, and big data in Internet of Things: A survey. *IEEE Internet Things J.* 5, 1 (Feb. 2018), 1–27.
[40] Nihar B. Shah and Martin J. Wainwright. 2017. Simple, robust and optimal ranking from pairwise comparisons. *J. Mach. Learn. Res.* 18, 199 (Jan. 2017), 1–199.
[41] Ruoyu Sun and Zhi-Quan Luo. 2016. Guaranteed matrix completion via non-convex factorization. *IEEE Trans. Inf. Theory* 62, 11 (Nov. 2016), 6535–6579.
[42] Alan M. Thompson, John C. Brown, Jim W. Kay, and D. Michael Titterington. 1991. A study of methods of choosing the smoothing parameter in image restoration by regularization. *IEEE Trans. Pattern Anal. Mach. Intell.* 13, 4 (Apr. 1991), 326–339.
[43] Maksims Volkovs and Richard S. Zemel. 2012. Collaborative ranking with 17 parameters. In *Adv. Neural. Inf. Process. Syst. (NIPS'12)*. 2294–2302.
[44] Kun Wang, Xin Qi, Lei Shu, Der-jiunn Deng, and Joel J. P. C. Rodrigues. 2016. Toward trustworthy crowdsourcing in the social Internet of Things. *IEEE Wireless Commun.* 23, 5 (Oct. 2016), 30–36.
[45] Markus Weimer, Alexandros Karatzoglou, Quoc V. Le, and Alex J. Smola. 2008. Cofi rank-maximum margin matrix factorization for collaborative ranking. In *Adv. Neural. Inf. Process. Syst. (NIPS'08)*. 1593–1600.
[46] Jun Xu, Wei Zeng, Yanyan Lan, Jiafeng Guo, and Xueqi Cheng. 2018. Modeling the parameter interactions in ranking SVM with low-rank approximation. *IEEE Trans. Knowl. Data Eng.* 31, 6 (Jun. 2018), 1181–1193.
[47] Peng Yang, Ning Zhang, Shan Zhang, Kan Yang, Li Yu, and Xuemin Shen. 2017. Identifying the most valuable workers in fog-assisted spatial crowdsourcing. *IEEE Internet Things J.* 4, 5 (Oct. 2017), 1193–1203.

Received January 2019; revised September 2019; accepted November 2019