Ranking from Crowdsourced Pairwise Comparisons via Matrix Manifold Optimization

Jialin Dong

ShanghaiTech University
Outline

- **Introduction**
- **Four Vignettes:**
  - System Model and Problem Formulation
  - Problem Analysis
    - Convex Methods
      - Disadvantage: Why Not Convex Optimization?
    - Scalable Nonconvex Optimization
      - Motivation: Why Nonconvex Optimization?
  - Matrix Manifold Optimization
    - Regularized Smoothed Maximum-likelihood Estimation
  - Simulation Results
- **Summary**
Part I: Introduction
Ranking

- Given \( n \) items, infer an ordering on the items based on partial sampled data

- Applications
Crowdsourcing

- **Crowdsourcing**: Harness the power of human computation to solve tasks
- **Applications**: Machine learning models, clustering data, scene recognition
Pairwise Measurements

- **Pairwise relations** over a few object pairs: difficult to directly measure each individual object

- **Applications** localization, Alignment, registration and synchronization, community detection... [Chen & Goldsmith, 14]
Part II : Four Vignettes
Vignettes A: *System Model and Problem Formulation*
System Model

- **m** crowd users, **n** items to be ranked
- **Underlying weight matrix** $X$: low-rank, unknown
- **Pairwise comparisons** $Y_{ijk} \in \{1, -1\}$, $(i, j, k) \in \Omega \subseteq [m] \times [n] \times [n]
- **Sampling set** $\Omega$: partial data
System Model

- **BTL model** [Bradley & Terry, 1952]:

  \[ Y_{ijk} = \begin{cases} +1 \text{ w.p. } f(\Delta_{ijk}) \\ -1 \text{ w.p. } 1 - f(\Delta_{ijk}) \end{cases} \quad \forall (i, j, k) \in \Omega, \]

  where \(\Delta_{ijk} = X_{ij} - X_{ik}\) with the logistic function is

  \[ f(z) = \frac{1}{1 + \exp\left(-\frac{z}{\sigma}\right)} \]

- **Associated score** \(\tau_j^{(i)} := \frac{1}{n} \sum_{k=1}^{n} f(\Delta_{ijk})\)

- **Individual ranking list for user i:**

  \[ \tau_{\pi(1)}^{(i)} > \tau_{\pi(2)}^{(i)} > \cdots > \tau_{\pi(n)}^{(i)}, \]
Problem Formulation

- **Maximum-likelihood estimation** (MLE): the negative log-likelihood function is given by

\[ \mathcal{L}_{\Omega,\mathcal{Y}}(X) = \sum_{(i,j,k) \in \Omega} -\log(f(Y_{ijk}(X_{ij} - X_{ik}))) \]

- **Low-rank matrix optimization problem**

\[
\begin{align*}
\text{minimize} & \quad \mathcal{L}_{\Omega,\mathcal{Y}}(X) \\
\text{subject to} & \quad \text{rank}(X) = r \\
& \quad \|X\|_\infty \leq \alpha,
\end{align*}
\]

where \( r \ll \min\{m, n\}. \)

How to solve this low-rank matrix optimization problem?
Vignettes B: *Problem Analysis*
Vignettes B. I: Convex Methods
Generic low-rank matrix optimization

- Rank-constrained matrix optimization problem

\[
\begin{align*}
\text{minimize} \quad & f(\mathbf{A}(\mathbf{M})) \\
\text{subject to} \quad & \text{rank}(\mathbf{M}) = r
\end{align*}
\]

- \( f : \mathbb{R}^d \rightarrow \mathbb{R} \) is a real-linear map on \( n \times n \) matrices
- \( \mathbf{A} : \mathbb{R}^{n \times n} \rightarrow \mathbb{R}^d \) is convex and differentiable

- **Challenge I:** Reliably solve the low-rank matrix problem at scale
- **Challenge II:** Develop optimization algorithms with optimal storage
A brief biased history of convex methods

- **1990s:** Interior-point methods *(computationally expensive)*
  - Storage cost $\Theta(n^4)$ for Hessian

- **2000s:** Convex first-order methods
  - (Accelerated) proximal gradient, spectral bundle methods, and others
  - Store matrix variable $\Theta(n^2)$

- **2008-Present:** Storage-efficient convex first-order methods
  - Conditional gradient method (CGM) and extensions
  - Store matrix in low-rank form $O(tn)$ after $t$ iterations: *no storage guarantees*

*Interior-point:* Nemirovski & Nesterov 1994; ... *First-order:* Rockafellar 1976; Helmberg & Rendl 1997; Auslender & Teboulle 2006; ... *CGM:* Frank & Wolfe 1956; Levitin & Poljak 1967; Jaggi 2013; ...
Convex Relaxation Approach

- Nuclear norm relaxation to solve the original problem:

\[
\begin{align*}
\minimize_{X \in \mathbb{R}^{m \times n}} & \quad \mathcal{L}_{\Omega, \gamma}(X) \\
\text{subject to} & \quad \|X\|_* \leq \alpha \sqrt{rmn},
\end{align*}
\]

- **Theoretical foundations**: Beautiful, nearly complete theory

- **Effective algorithms**: Spectral projected-gradient (SPG) [Davenport et al., 14], Newton-ADMM method [Ali et al., 17]…

  - Use generic methods for not huge problems: high level language support (CVX/CVXPY/Convex.jl) makes prototyping easy
Why not

- Convex methods have slow memory hogs, high computational complexity
  - Computationally expensive:
  - Storage issue:
Vignettes B.2: *Scalable Nonconvex Optimization*
Recent advances in nonconvex optimization

- **2009–Present: Nonconvex heuristics**
  - Burer–Monteiro factorization idea + various nonlinear programming methods
  - Store low-rank matrix factors $\Theta(rn)$

- **Guaranteed solutions:** Global optimality with statistical assumptions
  - Matrix completion/recovery: [Sun-Luo’14], [Chen-Wainwright’15], [Ge-Lee-Ma’16],…
  - Phase retrieval: [Candes et al., 15], [Chen-Candes’ 15], [Sun-Qu-Wright’16]
  - Community detection/phase synchronization [Bandeira-Boumal-Voroninski’16], [Montanari et al., 17],…
Why Nonconvex Optimization

- **Convex methods:**
  - Slow memory hogs: $\Theta(n^2)$
  - High computational complexity, e.g., singular value decomposition

- **Nonconvex methods:** fast, lightweight
  - Under certain statistical models with benign global geometry
  - Store low-rank matrix factors $\Theta(rn)$
Vignettes C: Matrix Manifold Optimization
What is matrix manifold optimization?

- Matrix manifold (or Riemannian) optimization problem

\[
\begin{align*}
\text{minimize} & \quad f(M) \\
\text{subject to} & \quad M \in \mathcal{M}
\end{align*}
\]

- \( f : \mathbb{R}^{m \times n} \rightarrow \mathbb{R} \) is a smooth function

- \( \mathcal{M} \) is a Riemannian manifold: spheres, orthonormal bases (Stiefel), rotations, positive definite matrices, fixed-rank matrices, Euclidean distance matrices, semidefinite fixed-rank matrices, linear subspaces (Grassmann), phases, essential matrices, fixed-rank tensors, Euclidean spaces...

How to reformulate the original problem to Riemannian optimization problem?
Reformulate to the Riemannian optimization problem

- The original problem

\[
\begin{align*}
\text{minimize} & \quad \mathcal{L}_{\Omega,Y}(X) \\
\text{subject to} & \quad \text{rank}(X) = r \\
& \quad \|X\|_{\infty} \leq \alpha,
\end{align*}
\]

where \( r \leq \min\{m, n\} \).

- The reformulated matrix manifold optimization problem
  - Problem with respect to **Fixed-rank matrices manifold**
    \[ X \in \mathcal{M}, \quad \mathcal{M} := \mathbb{R}^{m \times r}_* \times \mathbb{R}^{n \times r}_*, \quad \mathbb{R}^{m \times r}_* : \mathbb{R}^{n \times r} \text{ removing the origin} \]
  - **Challenge**: nonsmooth elementwise infinity norm constraint
Smoothing

- **Motivation:** derive smooth objective to implement Riemannian optimization

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

non-smooth $\|X\|_\infty^2$  

smoothed $\mu \log \sum_{i,j} e^{X_{i,j}^2/\mu}$
Regularization

- **Motivation:** Address the constraint of smoothed surrogate of the element-wise infinity norm

- Convex regularized function:

\[
\mathcal{P} \text{ minimize } \quad F(\mathbf{X}) := \mathcal{L}_{\mathbf{X,Y}}(\mathbf{X}) + \lambda \log N(\mathbf{X}),
\]

where \( \mathcal{M} = \mathbb{R}_{*}^{m \times r} \times \mathbb{R}_{*}^{n \times r} \) \( N(\mathbf{X}) = \sum_{i,j} e^{X_{i,j}^2} \) and \( \lambda = r^2 \sqrt{K} \log K \)

How to develop the algorithm on the manifold?
Manifold optimization paradigms

- Generalize Euclidean gradient (Hessian) to **Riemannian gradient (Hessian)**

\[
\nabla_{\mathcal{M}} f(X^{(k)}) = P_{X^{(k)}} \left( \nabla f(X^{(k)}) \right)
\]

Riemannian Gradient  \hspace{2cm}  Euclidean Gradient

\[
X^{(k+1)} = \mathcal{R}_{X^{(k)}} ( -\alpha^{(k)} \nabla_{\mathcal{M}} f(X^{(k)}))
\]

Retraction Operator

- We need Riemannian geometry: 1) linearize search space \( \mathcal{M} \) into a tangent space \( T_{X} \mathcal{M} \); 2) pick a metric, i.e., Riemannian metric, on \( T_{X} \mathcal{M} \) to give intrinsic notions of gradient and Hessian
Optimization on the manifold: main idea
Optimization on the manifold: main idea
Optimization on the manifold: main idea
Optimization on the manifold: main idea
Example: Rayleigh quotient

- Optimization over (sphere) manifold $\mathbb{S}^{n-1} = \{x \in \mathbb{R}^n : x^T x = 1\}$

  \[
  \text{minimize } f(x) = -x^T Ax \quad \text{subject to } x^T x = 1
  \]

  ➢ The cost function is smooth on $\mathbb{S}^{n-1}$, symmetric matrix $A \in \mathbb{R}^{n \times n}$

- Step 1: Compute the **Euclidean gradient** in $\mathbb{R}^n$

  \[
  \nabla f(x) = -2Ax
  \]

- Step 2: Compute the **Riemannian gradient** on $\mathbb{S}^{n-1}$ via projecting $\nabla f(x)$ to the tangent space using the orthogonal projector $\text{Proj}_x u = (I - xx^T)u$

  \[
  \nabla f(x) = \text{Proj}_x \nabla f(x) = -2(I - xx^T)Ax
  \]
Ranking problem in this paper

- Low-rank optimization for ranking via Riemannian optimization

<table>
<thead>
<tr>
<th>Matrix representation of an element $X \in \mathcal{M}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Computational space $\mathcal{M}$</td>
</tr>
<tr>
<td>Quotient space</td>
</tr>
<tr>
<td>Metric $g_X(\xi_X, \xi_X)$ for $\xi_X, \xi_X \in T_X \mathcal{M}$</td>
</tr>
<tr>
<td>Riemannian gradient $\text{grad}_X f$</td>
</tr>
<tr>
<td>Riemannian Hessian $\text{Hess}_X f[\xi_X]$</td>
</tr>
<tr>
<td>Retraction $\mathcal{R}_X : T_X \mathcal{M} \to \mathcal{M}$</td>
</tr>
</tbody>
</table>

| $\mathcal{P}$ : minimize $\mathcal{L}_{\Omega,Y}(LR^T) + \lambda \log N(LR^T)$ |
| $X = (L, R)$ |
| $\mathbb{R}^{m \times r} \times \mathbb{R}^{n \times r}$ |
| $\mathcal{M}/\sim := (\mathbb{R}^{m \times r} \times \mathbb{R}^{n \times r})/\text{GL}(r)$ |
| $g_X(\xi_X, \xi_X) = \text{Tr}((L^TL)^{-1}\xi_L^T\xi_L) + \text{Tr}((R^TR)^{-1}\xi_R^T\xi_R)$ |
| $\text{grad}_X f = (\text{grad}_L f, \text{grad}_R f) = (\nabla_L F(X)L^TL, \nabla_R F(X)R^TR)$ |
| $\text{Hess}_X f[\xi_X] = \Pi_{\mathcal{H}_X \mathcal{M}}(\nabla \xi_X \text{grad}_X f)$ |
| $\mathcal{R}_X(\xi_X) = (L + \xi_L, R + \xi_R)$ |

How to efficiently compute the descent direction on the manifold?
Riemannian trust-region method

- **Sub-optimal problem**

\[
\begin{align*}
\text{minimize} & \quad F(X_t) + g_x(t)(\xi x_t, \text{grad}_x f) + \frac{1}{2} g_x(t)(\xi x_t, \text{Hess}_x f [\xi x_t]) \\
\text{subject to} & \quad g_x(t)(\xi x_t, \xi x_t) \leq \delta_t^2,
\end{align*}
\]

- **Update the iterate**

\[
\rho_k = \frac{F(X_t) - F(\mathcal{R}_X(\xi x_t))}{m(0_{X_t}) - m(\xi x_t)}.
\]

\[
\begin{cases}
X^{(t+1)} = X^{(t)}, & \delta^{(t+1)} = \frac{1}{4} \delta_t, & \rho_k < \frac{1}{4} \\
X^{(t+1)} = X^{(t)}, & \delta^{(t+1)} = \min(2\delta_t, \delta_{\text{max}}), & \rho_k > \frac{3}{4} \\
X^{(t+1)} = \mathcal{R}_{X(t)}(X^{(t)}), & \delta^{(t+1)} = \delta(t), & \text{otherwise}
\end{cases}
\]
Vignettes D: *Simulation Result*
Algorithms & Metric

- **Algorithms**
  - Proposed Riemannian trust-region algorithm solving log-sum-exp regularized problem (PRTRS)
  - Bi-factor gradient descent solving log-barrier regularized problem (BFGDB) [Park et al., 16]
  - Spectral projected-gradient (SPG) [Davenport et al., 14]

- **Metric**
  - Sampling size: $|\Omega| = drK\log K$, rescaled sample size: $d$
  - Relative mean square error (RMSE): $\text{err}(X) = \|X - X^*\|_F^2 / \|X^*\|_F^2$. 
Convergence rate

- **Objective**
  - **PRTRS**
    
    $$F(\mathbf{X}) := \mathcal{L}_{\Omega, \mathbf{Y}}(\mathbf{X}) + \lambda \log N(\mathbf{X})$$
  
    - **BFGDB** [Park et al., 16]
      
      $$\mathcal{L}_{\Omega, \mathbf{Y}}(U \mathbf{V}^T) - \frac{1}{\tau} \sum_{a,b} \log(1 - (U_{a,.} V_{b,.}/\alpha)^2)$$
  
    - **SPG** [Davenport et al., 14]
      
      $$\mathcal{L}_{\Omega, \mathbf{Y}}(\mathbf{X})$$

- **PRTRS:**
  - Faster rate of convergence than BFGDB algorithm
  - Comparable with SPG algorithm
Performance

- Conclusion
  - Rescaled sample size increases, relative MSE reduces
  - **PRTRS**: better performance in terms of MSE than both SPG and BFGDB
Computational Cost

- Conclusion
  - Computational time with different sizes $K$ respectively
  - **PRTRS**: dramatical advantage in computational time of both SPG and BFGDB
Part III: Summary
Concluding remarks

- Ranking from pairwise comparisons in the crowdsourcing system
- Scalable nonconvex optimization algorithms
  - Store low-rank matrix factors $\Theta(n)$
  - Global optimality with statistical assumptions
- Matrix manifold optimization
  - Smoothed regularized MLE
  - Riemannian trust-region algorithm: Outperform the state-of-the-art algorithms
    - performance (i.e., relative MSE)
    - computational cost
    - convergence rate
Thanks