Divide-and-Conquer Matrix Factorization

Lester Mackey

Collaborators: Ameet Talwalkar and Michael I. Jordan

January 23, 2012
Motivation: Low-rank Matrix Factorization

\[
\begin{bmatrix}
1 & 3 & 3 & 1 & \ldots & 3 \\
5 & 4 & 4 & 5 & \ldots & 4 \\
3 & 5 & 5 & 3 & \ldots & 5 \\
\end{bmatrix}
\]
Motivation: Low-rank Matrix Factorization

\[
\begin{bmatrix}
1 & 3 & 3 & 1 & \ldots & 3 \\
5 & 4 & 4 & 5 & \ldots & 4 \\
3 & 5 & 5 & 3 & \ldots & 5 \\
\end{bmatrix}
= 
\begin{bmatrix}
1 & 3 \\
5 & 4 \\
3 & 5 \\
\end{bmatrix}
\begin{bmatrix}
1 & 0 & 0 & 1 & \ldots & 0 \\
0 & 1 & 1 & 0 & \ldots & 1 \\
\end{bmatrix}
\]
Motivation: Low-rank Matrix Factorization

\[
\begin{bmatrix}
1 & 3 & 3 & 1 & \ldots & 3 \\
5 & 4 & 4 & 5 & \ldots & 4 \\
3 & 5 & 5 & 3 & \ldots & 5 \\
\end{bmatrix}
\]
Motivation: Low-rank Matrix Factorization

\[
\begin{bmatrix}
1 & ? & 3 & 1 & \ldots & 3 \\
5 & ? & 4 & ? & \ldots & ? \\
? & 5 & ? & 3 & \ldots & 5
\end{bmatrix}
\]
Motivation: Low-rank Matrix Factorization

\[
\begin{bmatrix}
5 & ? & 3 & 1 & \ldots & 4 \\
5 & ? & 4 & ? & \ldots & ? \\
? & 5 & ? & 3 & \ldots & 5
\end{bmatrix}
\]
Motivation: Low-rank Matrix Factorization

\[
\begin{bmatrix}
5 & ? & 3 & 1 & \ldots & 4 \\
5 & ? & 4 & ? & \ldots & ? \\
? & 5 & ? & 3 & \ldots & 5 \\
\end{bmatrix} \rightarrow \begin{bmatrix}
1 & 3 & 3 & 1 & \ldots & 3 \\
5 & 4 & 4 & 5 & \ldots & 4 \\
3 & 5 & 5 & 3 & \ldots & 5 \\
\end{bmatrix}
\]
Introduction

Motivation: Low-rank Matrix Factorization

**Goal:** Given a matrix $M \in \mathbb{R}^{m \times n}$ formed by deleting and corrupting the entries of $L_0$, recover the underlying low-rank matrix $L_0$.

$$
M = \begin{bmatrix}
5 & ? & 3 & 1 & \ldots & 4 \\
5 & ? & 4 & ? & \ldots & ? \\
? & 5 & ? & 3 & \ldots & 5
\end{bmatrix} \rightarrow \begin{bmatrix}
1 & 3 & 3 & 1 & \ldots & 3 \\
5 & 4 & 4 & 5 & \ldots & 4 \\
3 & 5 & 5 & 3 & \ldots & 5
\end{bmatrix} = L_0
$$

1. **Matrix completion (MC):** Small fraction of entries revealed
2. **Robust matrix factorization (RMF):** Fraction of entries grossly corrupted
Goal: Given a matrix $M \in \mathbb{R}^{m \times n}$ formed by deleting and corrupting the entries of $L_0$, recover the underlying low-rank matrix $L_0$.

$$M = \begin{bmatrix} 5 & ? & 3 & 1 & \ldots & 4 \\ 5 & ? & 4 & ? & \ldots & ? \\ ? & 5 & ? & 3 & \ldots & 5 \end{bmatrix} \rightarrow \begin{bmatrix} 1 & 3 & 3 & 1 & \ldots & 3 \\ 5 & 4 & 4 & 5 & \ldots & 4 \\ 3 & 5 & 5 & 3 & \ldots & 5 \end{bmatrix} = L_0$$

Examples: Matrix completion
- Collaborative filtering: How will user $i$ rate movie $j$?
  - Netflix: 10 million users, 100K DVD titles
- Ranking on the web: Is URL $j$ relevant to user $i$?
  - Google News: millions of articles, millions of users
Motivation: Low-rank Matrix Factorization

**Goal:** Given a matrix $\mathbf{M} \in \mathbb{R}^{m \times n}$ formed by deleting and corrupting the entries of $\mathbf{L}_0$, recover the low-rank matrix $\mathbf{L}_0$.

\[
\mathbf{M} = \begin{bmatrix}
5 & ? & 3 & 1 & \ldots & 4 \\
5 & ? & 4 & ? & \ldots & ? \\
? & 5 & ? & 3 & \ldots & 5 \\
\end{bmatrix} \rightarrow \begin{bmatrix}
1 & 3 & 3 & 1 & \ldots & 3 \\
5 & 4 & 4 & 5 & \ldots & 4 \\
3 & 5 & 5 & 3 & \ldots & 5 \\
\end{bmatrix} = \mathbf{L}_0
\]

**Examples:** Robust matrix factorization

- Background modeling/foreground activity detection

(Candès, Li, Ma, and Wright, 2011)
Motivation: Low-rank Matrix Factorization

Goal: Given a matrix $M \in \mathbb{R}^{m \times n}$ formed by deleting and corrupting the entries of $L_0$, recover the low-rank matrix $L_0$.

$$
M = \begin{bmatrix}
5 & ? & 3 & 1 & \ldots & 4 \\
5 & ? & 4 & ? & \ldots & ? \\
? & 5 & ? & 3 & \ldots & 5 \\
\end{bmatrix} \rightarrow \begin{bmatrix}
1 & 3 & 3 & 1 & \ldots & 3 \\
5 & 4 & 4 & 5 & \ldots & 4 \\
3 & 5 & 5 & 3 & \ldots & 5 \\
\end{bmatrix} = L_0
$$

State of the art MF algorithms

- Strong recovery guarantees
- Plagued by expensive subroutines (e.g., truncated SVD)

This talk

- Present divide and conquer approaches for scaling up any MF algorithm while maintaining strong recovery guarantees
Roadmap

1. Introduction
2. Matrix Completion
   - Background
   - Divide-Factor-Combine
   - Simulations
   - Collaborative filtering
3. Robust Matrix Factorization
   - Background
   - Simulations
   - Video background modeling
4. Future Directions
**Goal:** Given entries from a matrix $\mathbf{L}_0 \in \mathbb{R}^{m \times n}$ with rank $r \ll m, n$, recover $\mathbf{L}_0$. 
Noisy Matrix Completion

**Goal:** Given entries from a matrix \( \mathbf{M} = \mathbf{L}_0 + \mathbf{Z} \in \mathbb{R}^{m \times n} \) where \( \mathbf{L}_0 \) has rank \( r \ll m, n \) and \( \mathbf{Z} \) is entrywise noise, recover \( \mathbf{L}_0 \).

- **Good news:** \( \mathbf{L}_0 \) has \( \sim (m + n)r \ll mn \) degrees of freedom

\[ \mathbf{L}_0 = \mathbf{A} \mathbf{B}^\top \]

- Factored form: \( \mathbf{A} \mathbf{B}^\top \) for \( \mathbf{A} \in \mathbb{R}^{m \times r} \) and \( \mathbf{B} \in \mathbb{R}^{n \times r} \)

- **Bad news:** Not all low-rank matrices can be recovered

**Question:** What can go wrong?
What can go wrong?

Entire column missing

\[
\begin{bmatrix}
1 & 2 & ? & 3 & \ldots & 4 \\
3 & 5 & ? & 4 & \ldots & 1 \\
2 & 5 & ? & 2 & \ldots & 5
\end{bmatrix}
\]

- No hope of recovery!

Solution: Uniform observation model

Assume that the set of \( s \) observed entries \( \Omega \) is drawn uniformly at random:

\[ \Omega \sim \text{Unif}(m, n, s) \]
Matrix Completion

Background

What can go wrong?

Bad spread of information

\[
L = \begin{bmatrix}
1 \\
0 \\
0
\end{bmatrix} \begin{bmatrix}
1 & 1 & 0 & 0
\end{bmatrix} = \begin{bmatrix}
1 & 0 & 0 \\
0 & 0 & 0 \\
0 & 0 & 0
\end{bmatrix}
\]

- Can only recover \( L \) if \( L_{11} \) is observed

Solution: Incoherence with standard basis (Candès and Recht, 2009)

A matrix \( L = U \Sigma V^\top \in \mathbb{R}^{m \times n} \) with \( \text{rank}(L) = r \) is \((\mu, r)\)-coherent if

Singular vectors are not too sparse:

\[
\begin{align*}
\max_i \|UU^\top e_i\|^2 &\leq \frac{\mu r}{m} \\
\max_i \|VV^\top e_i\|^2 &\leq \frac{\mu r}{n}
\end{align*}
\]

and not too cross-correlated:

\[
\|UV^\top\|_\infty \leq \sqrt{\frac{\mu r}{mn}}
\]
How do we recover $L_0$?

First attempt:

\[
\begin{align*}
\text{minimize}_A & \quad \text{rank}(A) \\
\text{subject to} & \quad \sum_{(i,j) \in \Omega} (A_{ij} - M_{ij})^2 \leq \Delta^2.
\end{align*}
\]

Problem: Intractable to solve!

Solution: Solve convex relaxation (Fazel, Hindi, and Boyd, 2001; Candès and Plan, 2010)

\[
\begin{align*}
\text{minimize}_A & \quad \|A\|_* \\
\text{subject to} & \quad \sum_{(i,j) \in \Omega} (A_{ij} - M_{ij})^2 \leq \Delta^2
\end{align*}
\]

where $\|A\|_* = \sum_k \sigma_k(A)$ is the trace/nuclear norm of $A$.

Questions:

- Will the nuclear norm heuristic successfully recover $L_0$?
- Can nuclear norm minimization scale to large MC problems?
Noisy Nuclear Norm Heuristic: Does it work?

Yes, with high probability.

**Typical Theorem**

If $L_0$ is $(\mu, r)$-coherent, $s = O(\mu r n \log^2(n))$ entries of $M \in \mathbb{R}^{m \times n}$ are observed uniformly at random, and $\hat{L}$ solves the noisy nuclear norm heuristic, then

$$\|\hat{L} - L_0\|_F \leq f(m, n) \Delta$$

with high probability when $\|M - L_0\|_F \leq \Delta$.

- See Candès and Plan (2010); Mackey, Talwalkar, and Jordan (2011); Keshavan, Montanari, and Oh (2010); Negahban and Wainwright (2010)
- Implies **exact** recovery in the noiseless setting ($\Delta = 0$)
Not quite...

- Standard interior point methods (Candès and Recht, 2009):
  \[ O(|\Omega| (m + n)^3 + |\Omega|^2 (m + n)^2 + |\Omega|^3) \]

- More efficient, tailored algorithms:
  - Singular Value Thresholding (SVT) (Cai, Candès, and Shen, 2010)
  - Augmented Lagrange Multiplier (ALM) (Lin, Chen, Wu, and Ma, 2009a)
  - Accelerated Proximal Gradient (APG) (Toh and Yun, 2010)
  - All require rank-$k$ truncated SVD on every iteration

**Take away**: Provably accurate MC algorithms are still too expensive for large-scale or real-time matrix completion

**Question**: How can we scale up a given matrix completion algorithm and still retain recovery guarantees?
**Divide-Factor-Combine (DFC)**

**Idea:** Divide and conquer

1. Divide $M$ into submatrices.
2. Factor each submatrix in parallel.
3. Combine submatrix estimates to recover $L_0$.

**Advantages**

- Factoring a submatrix is often much cheaper than factoring $M$
- Multiple submatrix factorizations can be carried out in parallel
- DFC works with **any** base MC algorithm
- With the right choice of division and recombination, yields recovery guarantees comparable to those of the base algorithm
DFC-Nys: Generalized Nyström Decomposition

1. Choose a random column submatrix $C \in \mathbb{R}^{m \times l}$ and a random row submatrix $R \in \mathbb{R}^{d \times n}$ from $M$. Call their intersection $W$.

$$M = \begin{bmatrix} W & M_{12} \\ M_{21} & M_{22} \end{bmatrix} \quad C = \begin{bmatrix} W \\ M_{21} \end{bmatrix} \quad R = \begin{bmatrix} W & M_{12} \end{bmatrix}$$

2. Recover the low rank components of $C$ and $R$ in parallel to obtain $\hat{C}$ and $\hat{R}$.
   - Reduced cost: Expect $\min(n/l, m/d)$ speed-up per iteration

3. Recover $L_0$ from $\hat{C}$, $\hat{R}$, and their intersection $\hat{W}$
   $$\hat{L}^{nys} = \hat{C}\hat{W} + \hat{R}$$
   - Generalized Nyström method (Goreinov, Tyrtyshnikov, and Zamarashkin, 1997)
   - Minimal cost: $O(mk^2 + lk^2 + dk^2)$ where $k = \text{rank}(\hat{L}^{nys})$

4. Ensemble: Run $p$ times in parallel and average estimates
DFC-Proj: Partition and Project

1. Randomly partition $M$ into $n/l$ column submatrices $M = \begin{bmatrix} C_1 & C_2 & \cdots & C_{n/l} \end{bmatrix}$ where each $C_i \in \mathbb{R}^{m \times l}$

2. Complete the submatrices \textbf{in parallel} to obtain

$$\begin{bmatrix} \hat{C}_1 & \hat{C}_2 & \cdots & \hat{C}_{n/l} \end{bmatrix}$$

3. Recover a single factorization for $M$ by projecting each submatrix onto the column space of $\hat{C}_1$

$$\hat{L}^{proj} = \hat{C}_1 \hat{C}_1^+ \begin{bmatrix} \hat{C}_1 & \hat{C}_2 & \cdots & \hat{C}_{n/l} \end{bmatrix}$$

- \textbf{Minimal cost:} $O(mk^2 + lk^2)$ where $k = \text{rank}(\hat{L}^{proj})$

4. \textbf{Ensemble:} Project onto column space of each $\hat{C}_j$ and average

\textbf{Advantages} over DFC-Nys

- Utilizes the entire matrix $M$
- Column matrices are easier to factor when $n > m$
Yes, with high probability.

**Theorem** (Mackey, Talwalkar, and Jordan, 2011)

If $L_0$ is $(\mu, r)$-coherent and $s$ entries of $M \in \mathbb{R}^{m \times n}$ are observed uniformly at random, then

$$l = O\left(\frac{\mu^2 r^2 n^2 \log^2(n)}{s \epsilon^2}\right)$$

random columns suffice to have

$$\|\hat{L}^{proj} - L_0\|_F \leq (2 + \epsilon) f(m, n) \Delta$$

with high probability when $\|M - L_0\|_F \leq \Delta$ and the noisy nuclear norm heuristic is used as a base algorithm.

- Can sample vanishingly small fraction of columns $(l/n \to 0)$ whenever $s = \omega(n \log^2(n))$
- Implies exact recovery for noiseless $(\Delta = 0)$ setting
**DFC: Does it work?**

Yes, with high probability.

**Proof Ideas:**

1. Uniform column/row sampling yields **submatrices with low coherence** (high spread of information) w.h.p.
2. Each submatrix has **sufficiently many observed entries** w.h.p.  
   ⇒ Submatrix completion succeeds
3. Uniform sampling of columns/rows **captures the full column/row space** of $\mathbf{L}_0$ w.h.p.  
   - Noisy analysis builds on randomized $\ell_2$ regression work of Drineas, Mahoney, and Muthukrishnan (2008)  
   ⇒ Generalized Nyström method and column projection succeed
DFC Noisy Recovery Error

Figure: Recovery error of DFC relative to base algorithms with \((m = 10K, r = 10)\).
DFC Speed-up

Figure: Speed-up over APG for random matrices with $r = 0.001m$ and 4% of entries revealed.
Application: Collaborative filtering

**Task:** Given a sparsely observed matrix of user-item ratings, predict the unobserved ratings

**Issues**
- Full-rank rating matrix
- Noisy, non-uniform observations

**The Data**
- **Netflix Prize Dataset**
  - 100 million ratings in \{1, \ldots, 5\}
  - 17,770 movies, 480,189 users

\(^1\text{http://www.netflixprize.com/}\)
## Application: Collaborative filtering

<table>
<thead>
<tr>
<th>Method</th>
<th>Netflix</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>RMSE</td>
</tr>
<tr>
<td>APG</td>
<td>0.8433</td>
</tr>
<tr>
<td>DFC-NYS-25%</td>
<td>0.8832</td>
</tr>
<tr>
<td>DFC-NYS-10%</td>
<td>0.9224</td>
</tr>
<tr>
<td>DFC-NYS-ENS-25%</td>
<td>0.8486</td>
</tr>
<tr>
<td>DFC-NYS-ENS-10%</td>
<td>0.8613</td>
</tr>
<tr>
<td>DFC-PROJ-25%</td>
<td>0.8436</td>
</tr>
<tr>
<td>DFC-PROJ-10%</td>
<td>0.8484</td>
</tr>
<tr>
<td>DFC-PROJ-ENS-25%</td>
<td>0.8411</td>
</tr>
<tr>
<td>DFC-PROJ-ENS-10%</td>
<td>0.8433</td>
</tr>
</tbody>
</table>
**Goal:** Given a matrix $M = L_0 + S_0 + Z$ where $L_0$ is low-rank, $S_0$ is sparse, and $Z$ is entrywise noise, recover $L_0$

(Chandrasekaran, Sanghavi, Parrilo, and Willsky, 2009; Candès, Li, Ma, and Wright, 2011; Zhou, Li, Wright, Candès, and Ma, 2010)

- $S_0$ can be viewed as an outlier/gross corruption matrix
  - Ordinary PCA breaks down in this setting
- **Harder than MC:** outlier locations are unknown
- **More expensive than MC:** dense, fully observed matrices
How do we recover $L_0$?

First attempt:

$$\begin{align*}
\text{minimize}_{L,S} & \quad \text{rank}(L) + \lambda \text{card}(S) \\
\text{subject to} & \quad \|M - L - S\|_F \leq \Delta.
\end{align*}$$

**Problem:** Intractable to solve!

**Solution:** Convex relaxation

$$\begin{align*}
\text{minimize}_{L,S} & \quad \|L\|_* + \lambda \|S\|_1 \\
\text{subject to} & \quad \|M - L - S\|_F \leq \Delta.
\end{align*}$$

where $\|S\|_1 = \sum_{ij} S_{ij}$ is the $\ell_1$ entrywise norm of $S$.

**Question:** Does it work? Yes!

**Question:** Does it scale? Not quite.

**Idea:** Divide and conquer
Figure: Recovery error of DFC relative to base algorithms with \((m = 1K, r = 10)\).
DFC Speed-up

Figure: Speed-up over APG for random matrices with $r = 0.01m$ and 10% of entries corrupted.
Application: Video background modeling

Task
- Each video frame forms one column of matrix $M$
- Decompose $M$ into stationary background $L_0$ and moving foreground objects $S_0$

Issues
- Video is noisy
- Foreground corruption is often clustered, not uniform
Example: Changes in illumination

Specs

- 1.5 minutes of lobby surveillance (Li, Huang, Gu, and Tian, 2004)
- 1546 frames, 20480 pixels
- ALM running time: 5135.3s
- DFC-PROJ running time: 476.9s
Example: Significant foreground variation

Specs

- 1 minute of airport surveillance (Li, Huang, Gu, and Tian, 2004)
- 1000 frames, 25344 pixels
- ALM running time: 1871.0s
- DFC-PROJ running time: 392.8s
Future Directions

New Theory

- Analyze statistical implications of divide and conquer algorithms
  - Trade-off between statistical and computational efficiency
  - Impact of ensembling
- Analysis of alternative optimization problems
  - Weighted trace norm for non-uniform sampling (Salakhutdinov and Srebro, 2010; Negahban and Wainwright, 2010)
  - Non-convex matrix completion with recovery guarantees (Keshavan, Montanari, and Oh, 2010)

New Applications

- Practical problems with large-scale or real-time MF requirements
Thanks!


Yes, with high probability.

**Theorem (Zhou, Li, Wright, Candès, and Ma, 2010)**

If $\mathbf{L}_0$ is $(\mu, r)$-coherent and $\mathbf{S}_0 \in \mathbb{R}^{m \times n}$ contains $s$ non-zero entries with uniformly distributed locations, then if

$$r \leq \frac{\rho_r m}{\mu \log^2 n} \quad \text{and} \quad s \leq \rho_s mn$$

the minimizer to the problem

$$\min_{\mathbf{L}, \mathbf{S}} \quad \|\mathbf{L}\|_* + \lambda \|\mathbf{S}\|_1$$

subject to

$$\|\mathbf{M} - \mathbf{L} - \mathbf{S}\|_F \leq \Delta.$$ 

with $\lambda = 1/\sqrt{n}$ satisfies

$$\|\hat{\mathbf{L}} - \mathbf{L}_0\|_F \leq f(m, n)\Delta$$

with high probability when $\|\mathbf{M} - \mathbf{L}_0 - \mathbf{S}_0\|_F \leq \Delta$. 

Not quite...

- Standard interior point methods: $O(n^6)$ (Chandrasekaran, Sanghavi, Parrilo, and Willsky, 2009)

- More efficient, tailored algorithms:
  - Accelerated Proximal Gradient (APG) (Lin, Ganesh, Wright, Wu, Chen, and Ma, 2009b)
  - Augmented Lagrange Multiplier (ALM) (Lin, Chen, Wu, and Ma, 2009a)
  - Require rank-$k$ truncated SVD on every iteration
  - Best case $\text{SVD}(m, n, k) = O(mnk)$

Take away: Provably accurate RMF algorithms are still too expensive for large-scale or real-time matrix factorization

Idea: Leverage the divide-and-conquer techniques developed for MC in the RMF setting
**Idea:** Same divide-and-conquer strategies apply

1. Divide $M$ into submatrices.
2. Factor each submatrix **in parallel**.
   - Using any base RMF algorithm
3. Combine submatrix estimates to recover $L_0$.
   - Using DFC-NYS or DFC-PROJ

**Question:** Is it efficient?
- **Yes**, as in the MC case.

**Question:** Does it work?
Yes, with high probability.

**Theorem** (Mackey, Talwalkar, and Jordan, 2011)

If $L_0$ is $(\mu, r)$-coherent and $S_0 \in \mathbb{R}^{m \times n}$ contains $s \leq \rho_s mn$ non-zero entries with uniformly distributed locations, then

$$l = O\left(\frac{\mu^2 r^2 \log^2(n)}{\epsilon^2}\right)$$

columns suffice to have

$$\|\hat{L}^{proj} - L_0\|_F \leq (2 + \epsilon)f(m, n)\Delta$$

with high probability when $\|M - L_0 - S_0\|_F \leq \Delta$ and noisy principal component pursuit is used as the base algorithm.

- Implies exact recovery for noiseless ($\Delta = 0$) setting
- Similar result holds for DFC-Nys