Recommender systems

- **Problem**
  - Set of users
  - Set of items (movies, books, jokes, products, stories, ...)
  - Feedback (ratings, purchase, click-through, tags, ...)
  - Sometimes: metadata (user profiles, item properties, ...)

- **Goal**: Predict preferences of users for items
- **Ultimate goal**: Create item recommendations for each user

- **Example**

<table>
<thead>
<tr>
<th></th>
<th>Avatar</th>
<th>The Matrix</th>
<th>Up</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td>?</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>Bob</td>
<td>3</td>
<td>2</td>
<td>?</td>
</tr>
<tr>
<td>Charlie</td>
<td>5</td>
<td>?</td>
<td>3</td>
</tr>
</tbody>
</table>
Outline

1. Collaborative Filtering

2. Matrix Completion

3. Algorithms

4. Summary
Collaborative filtering

- Key idea: Make use of past user behavior
- No domain knowledge required
- No expensive data collection needed
- Allows discovery of complex and unexpected patterns
- Widely adopted: Amazon, TiVo, Netflix, Microsoft
- Key techniques: neighborhood models, latent factor models

<table>
<thead>
<tr>
<th></th>
<th>Avatar</th>
<th>The Matrix</th>
<th>Up</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td>?</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>Bob</td>
<td>3</td>
<td>2</td>
<td>?</td>
</tr>
<tr>
<td>Charlie</td>
<td>5</td>
<td>?</td>
<td>3</td>
</tr>
</tbody>
</table>

Leverage past behavior of other users and/or on other items.
A simple baseline

- **m** users, **n** items, **m** × **n** rating matrix **D**
- Revealed entries Ω = \{(i, j) \mid \text{rating } D_{ij} \text{ is revealed}\}, N = |Ω|
- **Baseline predictor**: \(b_{ui} = \mu + b_i + b_j\)
  - \(\mu = \frac{1}{N} \sum_{(i,j) \in \Omega} D_{ij}\) is the overall average rating
  - \(b_i\) is a **user bias** (user’s tendency to rate low/high)
  - \(b_j\) is an **item bias** (item’s tendency to be rated low/high)
- Least squares estimates: \(\arg\min_{b} \sum_{(i,j) \in \Omega} (D_{ij} - \mu - b_i - b_j)^2\)

<table>
<thead>
<tr>
<th>D</th>
<th>Avatar</th>
<th>Matrix</th>
<th>Up</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>(1.01)</td>
<td>(0.34)</td>
<td>(−1.32)</td>
</tr>
<tr>
<td>Alice</td>
<td>?</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>(0.32)</td>
<td>(4.5)</td>
<td>(3.8)</td>
<td>(2.1)</td>
</tr>
<tr>
<td>Bob</td>
<td>3</td>
<td>2</td>
<td>?</td>
</tr>
<tr>
<td>(−1.34)</td>
<td>(2.8)</td>
<td>(2.2)</td>
<td>(0.5)</td>
</tr>
<tr>
<td>Charlie</td>
<td>5</td>
<td>?</td>
<td>3</td>
</tr>
<tr>
<td>(0.99)</td>
<td>(5.2)</td>
<td>(4.5)</td>
<td>(2.8)</td>
</tr>
</tbody>
</table>

- \(m = 3\)
- \(n = 3\)
- \(\Omega = \{(1, 2), (1, 3), (2, 1), \ldots\}\)
- \(N = 6\)
- \(\mu = 3.17\)
- \(b_{32} = 3.17 + 0.99 + 0.34 = 4.5\)

Baseline does not account for personal tastes.
When does a user like an item?

- **Neighborhood models** (kNN): When he likes similar items
  - Find the top-\(k\) most similar items the user has rated
  - Combine the ratings of these items (e.g., average)
  - Requires a similarity measure (e.g., Pearson correlation coefficient)

  ![The Matrix](Image102x142 to 131x184) is similar to

  Unrated by Bob → predict 4

  ![Avatar](Image138x679 to 140x679) Bob rated 4

- **Latent factor models** (LFM): When similar users like similar items
  - More holistic approach
  - Users and items are placed in the same “latent factor space”
  - Position of a user and an item related to preference (via dot products)

![Mapping of movies and users](Image257x142 to 285x184)
Intuition behind latent factor models (1)

Matrix factorization models map both users and items to a joint latent factor space of dimensionality \( d \), such that

\[
\begin{align*}
    \mathbf{u}_i & \in \mathbb{R}^d, \\
    \mathbf{v}_j & \in \mathbb{R}^d,
\end{align*}
\]

where \( \mathbf{u}_i \) and \( \mathbf{v}_j \) are the factor vectors of item \( i \) and user \( j \), respectively. Each factor vector is decomposed as a linear combination of basis vectors, and the interaction between each user and item is represented by the inner product of their corresponding factor vectors.

The major challenge is computing the mapping of each item and user to factor vectors \( \mathbf{u}_i \) and \( \mathbf{v}_j \) due to the high portion of missing values caused by sparsity. In collaborative filtering, user feedback is typically presented by a densely populated user-item matrix, where each element \( r_{ui} \) represents the rating given by user \( u \) to item \( i \). The goal is to predict the unknown ratings and complete the rating matrix.

One strength of matrix factorization is that it allows computation of predicted ratings \( \hat{r}_{ui} \) for any item \( i \) and user \( u \) without the need for explicit feedback as

\[
\hat{r}_{ui} = \mathbf{u}_i^T \mathbf{v}_j
\]

Moreover, the ratings are usually non-negative, and the system can avoid predicting negative feedback.

To learn the factor vectors \( \mathbf{u}_i \) and \( \mathbf{v}_j \), the system minimizes the regularized squared error on the set of observed ratings:

\[
\min \sum_{ui} (r_{ui} - \hat{r}_{ui})^2 + \lambda (\|\mathbf{u}_i\|^2 + \|\mathbf{v}_j\|^2)
\]

where \( \lambda \) is the regularization constant that controls the complexity of the model.

The figure illustrates the latent factor approach, which characterizes both users and movies using two axes—male versus female and serious versus escapist. The movies are positioned according to their latent factors, where the position of a movie reflects the extent of interest the user has in items that are high in either axis. This visualization helps in understanding the user's preferences and in making recommendations.
Intuition behind latent factor models (2)

- Does user $u$ like item $v$?

  - Quality: measured via **direction** from origin ($\cos \angle(u, v)$)
    - Same direction $\rightarrow$ attraction: $\cos \angle(u, v) \approx 1$
    - Opposite direction $\rightarrow$ repulsion: $\cos \angle(u, v) \approx -1$
    - Orthogonal direction $\rightarrow$ oblivious: $\cos \angle(u, v) \approx 0$

- Strength: measured via **distance** from origin ($\|u\|\|v\|$)
  - Far from origin $\rightarrow$ strong relationship: $\|u\|\|v\|$ large
  - Close to origin $\rightarrow$ weak relationship: $\|u\|\|v\|$ small

- Overall preference: measured via **dot product** ($u \cdot v$)

  $$u \cdot v = \|u\|\|v\| \frac{u \cdot v}{\|u\|\|v\|} = \|u\|\|v\| \cos \angle(u, v)$$

  - Same direction, far out $\rightarrow$ strong attraction: $u \cdot v$ large positive
  - Opposite direction, far out $\rightarrow$ strong repulsion: $u \cdot v$ large negative
  - Orthogonal direction, any distance $\rightarrow$ oblivious: $u \cdot v \approx 0$

But how to select dimensions and where to place items and users?

Key idea: Pick dimensions that explain the known data well.
SVD and missing values

Input data

Rank-10 truncated SVD

10% of input data

Rank-10 truncated SVD

SVD treats missing entries as 0.
Latent factor models and missing values

Input data

Rank-10 LFM

10% of input data

Rank-10 LFM

LFMs “ignore” missing entries.
Latent factor models (simple form)

- Given rank $r$, find $m \times r$ matrix $L$ and $r \times n$ matrix $R$ such that
  \[ D_{ij} \approx [LR]_{ij} \quad \text{for} \quad (i, j) \in \Omega \]

- Least squares formulation
  \[ \min_{L, R} \sum_{(i,j) \in \Omega} (D_{ij} - [LR]_{ij})^2 \]

- Example ($r = 1$)

<table>
<thead>
<tr>
<th></th>
<th>Avatar</th>
<th>The Matrix</th>
<th>Up</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td>(1.98)</td>
<td>(4.4)</td>
<td>(2.3)</td>
</tr>
<tr>
<td></td>
<td>(2.24)</td>
<td>(3.8)</td>
<td>(1.4)</td>
</tr>
<tr>
<td>Bob</td>
<td>3</td>
<td>2</td>
<td>?</td>
</tr>
<tr>
<td></td>
<td>(1.21)</td>
<td>(2.7)</td>
<td>(2.3)</td>
</tr>
<tr>
<td>Charlie</td>
<td>5</td>
<td>?</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>(2.30)</td>
<td>(5.2)</td>
<td>(2.7)</td>
</tr>
</tbody>
</table>
Our winning entries consist of more than 100 different predictor sets, the majority of which are factorization models using some variants of the methods described here. Our discussions with other top teams and postings on the contest website (www.netflixprize.com) indicate that these are the most popular and successful methods for predicting ratings. We can identify the first few most important dimensions from a matrix decomposition and explore the movies’ location in this new space. Figure 3 shows the first two factors from the Netflix data matrix decomposition involving such schemes, which let it give less weight to less meaningful observations. If confidence is observed, then the model enhances the cost function (Equation 5) to account for confidence in observing levels, which let it give less weight to events; however, a recurring event is more likely to reflect user opinion. Implicit Feedback Datasets.”10

Example: Netflix prize data

(≈ 500k users, ≈ 17k movies, ≈ 100M ratings)
Latent factor models (summation form)

- Least squares formulation prone to overfitting
- More general summation form:

\[ L = \sum_{(i,j) \in \Omega} l_{ij}(L_{i*}, R_{*j}) + R(L, R), \]

- \( L \) is global loss
- \( L_{i*} \) and \( R_{*j} \) are user and item parameters, resp.
- \( l_{ij} \) is local loss, e.g., \( l_{ij} = (D_{ij} - [LR]_{ij})^2 \)
- \( R \) is regularization term, e.g., \( R = \lambda(\|L\|_F^2 + \|R\|_F^2) \)

- Loss function can be more sophisticated
  - Improved predictors (e.g., include user and item bias)
  - Additional feedback data (e.g., time, implicit feedback)
  - Regularization terms (e.g., weighted depending on amount of feedback)
  - Available metadata (e.g., demographics, genre of a movie)
Computer matrix factorization techniques have become a dominant methodology within the mainstream crowd-pleasers, is indeed, the third dimension in the factorization does end up separating these two. Moreover, the more refined factor models, whose descriptions involve more distinct sets of parameters, are more accurate. For comparison, the Netflix system achieves RMSE = 0.9514 on the same dataset, while the grand prize's required accuracy is RMSE = 0.8563.

Example: Netflix prize data

Root mean square error of predictions

- Plain
- With biases
- With implicit feedback
- With temporal dynamics (v.1)
- With temporal dynamics (v.2)

With temporal dynamics (v.2)

Koren et al., 2009.
The matrix completion problem

Complete these matrices!

\[
\begin{pmatrix}
1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 \\
1 & 1 & ? & 1 & 1 \\
1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 \\
\end{pmatrix}
\quad \begin{pmatrix}
1 & 1 & 1 & 1 & 1 \\
\end{pmatrix}
\]

Matrix completion is impossible without additional assumptions!

Let’s assume that underlying full matrix is “simple” (here: rank 1).

\[
\begin{pmatrix}
1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 \\
\end{pmatrix}
\quad \begin{pmatrix}
1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 \\
\end{pmatrix}
\]

When/how can we recover a low-rank matrix from a sample of its entries?
Rank minimization

Definition (rank minimization problem)

Given an \( n \times n \) data matrix \( D \) and an index set \( \Omega \) of revealed entries. The rank minimization problem is

\[
\begin{align*}
\text{minimize} & \quad \text{rank}(X) \\
\text{subject to} & \quad D_{ij} = X_{ij} \quad (i, j) \in \Omega \\
& \quad X \in \mathbb{R}^{n \times n}.
\end{align*}
\]

- Seeks for “simplest explanation” fitting the data
- If unique and sufficient samples, recovers \( D \) (i.e., \( X = D \))
- NP-hard

Time complexity of existing rank minimization algorithms double exponential in \( n \) (and also slow in practice).
Nuclear norm minimization

- **Rank**: \( \text{rank}(D) = |\{ \sigma_k(D) > 0 : 1 \leq k \leq n \}| = \sum_{k=1}^{n} I_{\sigma_k(D) > 0} \)
- **Nuclear norm**: \( \|D\|_* = \sum_{k=1}^{n} \sigma_k(D) \)

**Definition (nuclear norm minimization)**

Given an \( n \times n \) data matrix \( D \) and an index set \( \Omega \) of revealed entries. The **nuclear minimization problem** is

\[
\begin{align*}
\text{minimize} & \quad \|X\|_* \\
\text{subject to} & \quad D_{ij} = X_{ij} \quad (i,j) \in \Omega \\
& \quad X \in \mathbb{R}^{n \times n}.
\end{align*}
\]

- A heuristic for rank minimization
- Nuclear norm is convex function (thus local optimum is global opt.)

Can be optimized (more) efficiently via semidefinite programming.
Why nuclear norm minimization?

Figure 1. Unit ball of the nuclear norm for symmetric $2 \times 2$ matrices. The red line depicts a random one-dimensional affine space. Such a subspace will generically intersect a sufficiently large nuclear norm ball at a rank one matrix.

- Consider SVD of $D = U \Sigma V^T$
- Unit nuclear norm ball = convex combination $(\sigma_k)$ of rank-1 matrices of unit Frobenius $(U_{*k}V_{*k}^T)$
- Extreme points have low rank (in figure: rank-1 matrices of unit Frobenius norm)
- Nuclear norm minimization: inflate unit ball as little as possible to reach $D_{ij} = X_{ij}$
- Solution lies at extreme point of inflated ball $\rightarrow$ (hopefully) low rank
Relationship to LFM

- Recall regularized LFM ($L$ is $m \times r$, $R$ is $r \times n$):

  $$\min_{L,R} \sum_{(i,j) \in \Omega} (D_{ij} - [LR]_{ij})^2 + \lambda (\|L\|_F^2 + \|R\|_F^2)$$

- View as matrix completion problem by enforcing $D_{ij} = [LR]_{ij}$:

  minimize $$\frac{1}{2} (\|L\|_F^2 + \|R\|_F^2)$$

  subject to $D_{ij} = X_{ij}$, $(i,j) \in \Omega$

  $LR = X$.

- One can show: for $r$ chosen larger than rank of nuclear norm optimum, equivalent to nuclear norm minimization

- For some intuition, suppose $X = U\Sigma V^T$ at optimum $L$ and $R$:

  $$\frac{1}{2} (\|L\|_F^2 + \|R\|_F^2) \leq \frac{1}{2} (\|U\Sigma^{1/2}\|_F^2 + \|\Sigma^{1/2}V^T\|_F^2)$$

  $$= \frac{1}{2} \sum_{i=1}^n \sum_{k=1}^r (U_{i k}^2 \sigma_k + V_{i k}^2 \sigma_k)$$

  $$= \sum_{k=1}^r \sigma_k = \|X\|_*$$
When can we hope to recover \( \mathbf{D} \)? (1)

Assume \( \mathbf{D} \) is the \( 5 \times 5 \) all-ones matrix (rank 1).

\[
\begin{pmatrix}
1 & 1 & 1 & 1 & 1 \\
1 & ? & ? & ? & ?
\end{pmatrix}
\]

\[
\begin{pmatrix}
1 & ? & ? & 1 & ? \\
? & 1 & ? & ? & 1 \\
1 & ? & 1 & ? & ? \\
? & 1 & 1 & ? & ?
\end{pmatrix}
\]

Ok

\[
\begin{pmatrix}
1 & 1 & 1 & 1 & ? \\
1 & 1 & ? & ? & ? \\
1 & ? & ? & 1 & ? \\
1 & ? & ? & ? & ?
\end{pmatrix}
\]

Not unique
(column missed)

\[
\begin{pmatrix}
\end{pmatrix}
\]

Not unique
(insufficient samples)

Sampling strategy and sample size matter.
When can we hope to recover $\mathbf{D}$? (2)

Consider the following rank-1 matrices and assume few revealed entries.

\[
\begin{pmatrix}
1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 \\
\end{pmatrix}
\]

Ok ("incoherent")

\[
\begin{pmatrix}
20 & 20 & 22 & 20 & 20 \\
20 & 20 & 22 & 20 & 20 \\
22 & 22 & 24 & 22 & 22 \\
20 & 20 & 22 & 20 & 20 \\
20 & 20 & 22 & 20 & 20 \\
\end{pmatrix}
\]

Ok ("incoherent")

\[
\begin{pmatrix}
1 & 1 & 1 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
\end{pmatrix}
\]

Bad ("coherent")

→ first row required

\[
\begin{pmatrix}
1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
\end{pmatrix}
\]

Bad ("coherent")

→ $(1, 1)$-entry required

Properties of $\mathbf{D}$ matter.
When can we hope to recover D? (3)

Exact conditions under which matrix completion “works” is active research area:

- Which sampling schemes? (e.g., random, WR/WOR, active)
- Which sample size?
- Which matrices? (e.g., “incoherent” matrices)
- Noise (e.g., independent, normally distributed noise)

Theorem (Candès and Recht, 2009)

Let $D = U\Sigma V^T$. If $D$ is incoherent in that

$$\max_{ij} U_{ij}^2 \leq \frac{\mu_B}{n} \quad \text{and} \quad \max_{ij} V_{ij}^2 \leq \frac{\mu_B}{n}$$

for some $\mu_B = O(1)$, and if $\text{rank}(D) \leq \mu_B^{-1} n^{1/5}$, then $O(n^{6/5} r \log n)$ random samples without replacement suffice to recover $D$ exactly with high probability.
Outline

1. Collaborative Filtering
2. Matrix Completion
3. Algorithms
4. Summary
Overview

Latent factor models in practice

- Millions of users and items
- Billions of ratings
- Sometimes quite complex models

Many algorithms have been applied to large-scale problems

- Gradient descent and quasi-Newton methods
- Coordinate-wise gradient descent
- Stochastic gradient descent
- Alternating least squares
Continuous gradient descent

- Find minimum $\theta^*$ of function $L$
- Pick a starting point $\theta_0$
- Compute gradient $L'(\theta_0)$
- Walk downhill
- Differential equation

$$\frac{\partial \theta(t)}{\partial t} = -L'(\theta(t))$$

with boundary cond. $\theta(0) = \theta_0$

- Under certain conditions

$$\theta(t) \to \theta^*$$
Discrete gradient descent

- Find minimum \( \theta^* \) of function \( L \)
- Pick a starting point \( \theta_0 \)
- Compute gradient \( L'(\theta_0) \)
- Jump downhill
- Difference equation
  \[
  \theta_{n+1} = \theta_n - \epsilon_n L'(\theta_n)
  \]
- Under certain conditions, approximates CGD in that
  \[
  \theta^n(t) = \theta_n + "\text{steps of size } t"
  \]
satisfies the ODE as \( n \to \infty \)
Gradient descent for LFM
ts

Set $\theta = (L, R)$ and write

$$L(\theta) = \sum_{(i,j) \in \Omega} L_{ij}(L_i^*, R_j^*)$$

$$\nabla_{L_i^*} L(\theta) = \sum_{j \in \{ j' | (i,j') \in \Omega \}} \nabla_{L_i^*} L_{ij}(L_i^*, R_j^*)$$

GD epoch

1. Compute gradient
   - Initialize zero matrices $L^\nabla$ and $R^\nabla$
   - For each entry $(i,j) \in \Omega$, update gradients
     $$L_i^\nabla \leftarrow L_i^\nabla + \nabla_{L_i^*} L_{ij}(L_i^*, R_j^*)$$
     $$R_j^\nabla \leftarrow R_j^\nabla + \nabla_{R_j^*} L_{ij}(L_i^*, R_j^*)$$

2. Update parameters

$$L \leftarrow L - \epsilon_n L^\nabla$$

$$R \leftarrow R - \epsilon_n R^\nabla$$
Computing the gradient (example)

Simplest form (unregularized)

\[ L_{ij}(L_{i*}, R_{*j}) = (D_{ij} - L_{i*}R_{*j})^2 \]

Gradient computation

\[ \nabla_{L_{i'k}} L_{ij}(L_{i*}, R_{*j}) = \begin{cases} 
0 & \text{if } i' \neq i \\
-2R_{kj}(D_{ij} - L_{i*}R_{*j}) & \text{if } i' = i 
\end{cases} \]

Local gradient of entry \((i, j) \in \Omega\) nonzero only on row \(L_{i*}\) and column \(R_{*j}\).
Stochastic gradient descent

- Find minimum $\theta^*$ of function $L$
- Pick a starting point $\theta_0$
- Approximate gradient $\hat{L}'(\theta_0)$
- Jump “approximately” downhill
- Stochastic difference equation
  \[ \theta_{n+1} = \theta_n - \epsilon_n \hat{L}'(\theta_n) \]
- Under certain conditions, asymptotically approximates (continuous) gradient descent
Stochastic gradient descent for LFM

- Set $\theta = (L, R)$ and use
  
  $$L(\theta) = \sum_{(i,j) \in \Omega} L_{ij}(L_{i*}, R_{*j})$$

  $$L'(\theta) = \sum_{(i,j) \in \Omega} L'_{ij}(L_{i*}, R_{*j})$$

  $$\hat{L}'(\theta, z) = NL'_{izjz}(L_{iz*}, R_{*jz}),$$

  where $N = |\Omega|$ and $z = (i_z, j_z) \in \Omega$

- SGD epoch
  1. Pick a random entry $z \in \Omega$
  2. Compute approximate gradient $\hat{L}'(\theta, z)$
  3. Update parameters
     $$\theta_{n+1} = \theta_n - \epsilon_n \hat{L}'(\theta_n, z)$$
  4. Repeat $N$ times

SGD step affects only current row and column.
SGD in practice

Step size sequence \( \{ \epsilon_n \} \) needs to be chosen carefully
- Pick initial step size based on sample (of some rows and columns)
- Reduce step size gradually
- **Bold driver heuristic**: After every epoch
  - Increase step size slightly when loss decreased (by, say, 5%)
  - Decrease step size sharply when loss increased (by, say, 50%)

```
epoch
0 10 20 30 40 50 60
Mean Loss
0.6 0.8 1.0 1.2 1.4
```

Netflix data (unregularized)
Lessons learned

- Collaborative filtering methods learn from past user behavior

- Latent factor models are best-performing single approach for collaborative filtering
  - But often combined with other methods

- Users and items are represented in common latent factor space
  - Holistic matrix-factorization approach
  - Similar users/item placed at similar positions
  - Low-rank assumption = few “factors” influence user preferences

- Close relationship to matrix completion problem
  - Reconstruct a partially observed low-rank matrix

- SGD is simple and practical algorithm to solve LFM s in summation form
Suggested reading

- Y. Koren, R. Bell, C. Volinsky
  *Matrix factorization techniques for recommender systems*
  IEEE Computer, 42(8), p. 30–37, 2009
  [http://research.yahoo.com/pub/2859](http://research.yahoo.com/pub/2859)

- E. Candès, B. Recht
  *Exact matrix completion via convex optimization*
  [http://doi.acm.org/10.1145/2184319.2184343](http://doi.acm.org/10.1145/2184319.2184343)

- And references in the above articles