Matrix Completion

Shihua Zhang

Fall 2019

Overview

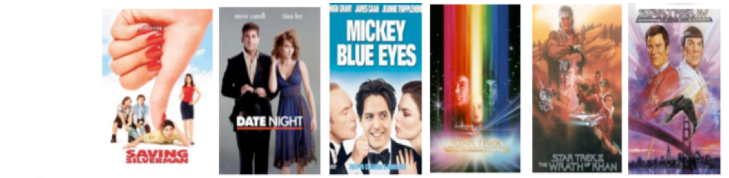
1. What is Matrix Completion?
2. Exact Matrix Completion
3. Approximate Matrix Completion
4. Follow-up Research

Outline

1. What is Matrix Completion?
2. Exact Matrix Completion
3. Approximate Matrix Completion
   - Nuclear Norm Minimization
   - Low-Rank Matrix Factorization
   - Distributed Matrix Completion
4. Follow-up Research

Movie Recommendation

- Websites recommend movies based on users' ratings.
- Users only saw a little part of the movies in the website.
- The ratings in the rating matrix are **very sparse**!!!

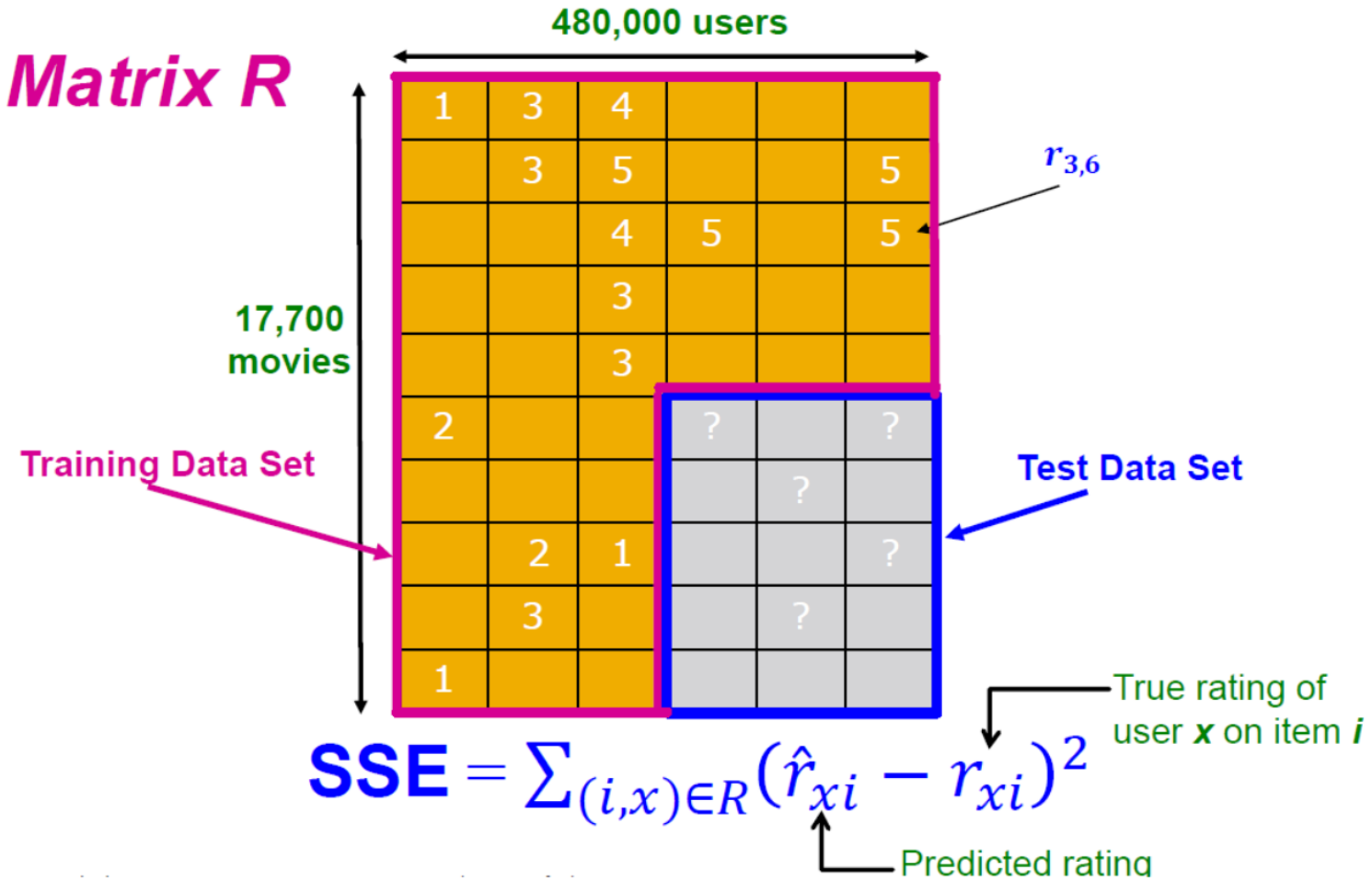


2			4	5	2.94*
5		4			1
		5		2	2.48*
	1		5		4
		4			2
4	5		1		1.12*

Netflix Prize

- Training data
  - 100 million ratings, 480,000 users, 17,770 movies
  - 6 years of data: 2000-2005
- Testing data
  - Last few ratings of each user (2.8 million)
  - Evaluation criterion – root mean squared error – RMSE=$\sqrt{\sum_{xi} (\hat{r}_{xi} - r_{xi})^2}$, where $\hat{r}_{xi}$ and r_{xi} are the predicted and true rating of x on i
  - Netflix Cinematch RMSE: 0.9514
- Competition
  - 2700+ teams
  - $1 million prize for 10% improvement on Cinematch

Netflix Prize



What is Matrix Completion?

Matrix Completion

Suppose you are given a matrix $M \in \mathbb{R}^{n_1 \times n_2}$ with some given entries $(M_{ij})_{ij \in \Omega}$, where $|\Omega| \ll n_1 n_2$

Problem: How to recover the missing elements in M?

What is Matrix Completion?

Matrix Completion

Suppose you are given a matrix $M \in \mathbb{R}^{n_1 \times n_2}$ with some given entries $(M_{ij})_{ij \in \Omega}$, where $|\Omega| \ll n_1 n_2$
Problem: How to recover the missing elements in M?

Lots of applications!!!

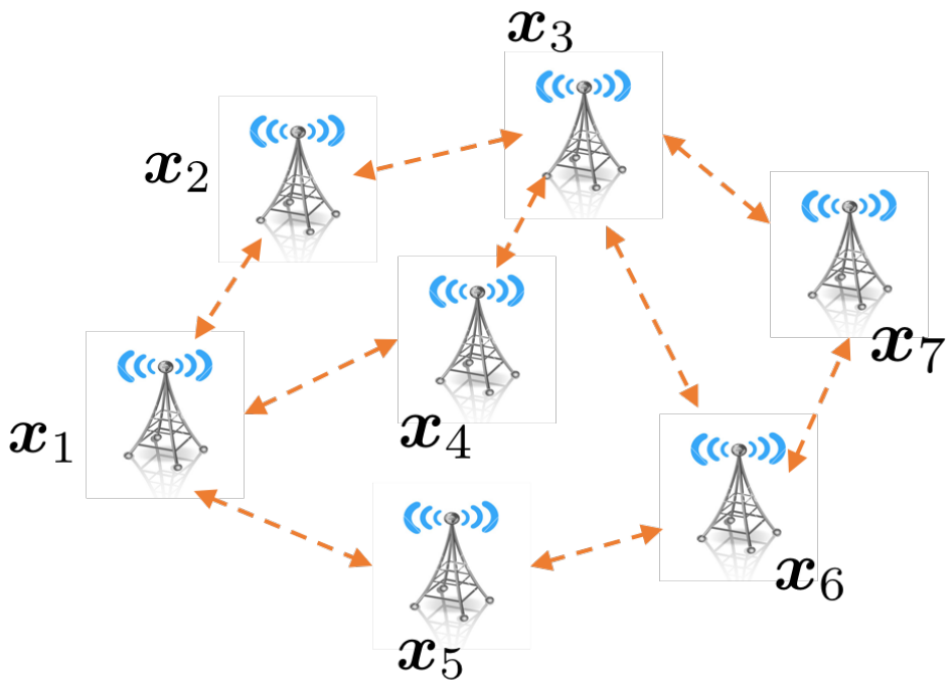
- news personalization
- sensor localization
- link prediction
- imputation
- ...

Application 1: Sensor Localization

- Given n sensors/points $\boldsymbol{x}_j \in R^3$ $(j = 1, \ldots, n)$
- Observe partial information about pairwise distances,

$$D_{i,j} = \|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2 = \|\boldsymbol{x}_i\|^2 + \|\boldsymbol{x}_j\|^2 - 2\boldsymbol{x}_i^\top \boldsymbol{x}_j \tag{1}$$

- **Aim:** infer the distances between any pair of locations



Application 1: Sensor Localization

Introduce

$$\boldsymbol{X} = \begin{bmatrix} \boldsymbol{x}_1^\top \\ \boldsymbol{x}_2^\top \\ \vdots \\ \boldsymbol{x}_n^\top \end{bmatrix} \in \mathbb{R}^{n\times 3} \tag{2}$$

then the distance matrix $D = [D_{i,j}]_{1\leqslant i,j\leqslant n}$ can be written as

$$\boldsymbol{D} = \underbrace{\boldsymbol{d}_2\boldsymbol{e}^\top + \boldsymbol{e}\boldsymbol{d}_2^\top - 2\boldsymbol{X}\boldsymbol{X}^\top}_{\text{low rank}} \tag{3}$$

where $\boldsymbol{d}_2 := \left[\|\boldsymbol{x}_1\|^2, \cdots, \|\boldsymbol{x}_n\|^2\right]^\top$.

Application 1: Sensor Localization

Introduce

$$\boldsymbol{X} = \begin{bmatrix} \boldsymbol{x}_1^\top \\ \boldsymbol{x}_2^\top \\ \vdots \\ \boldsymbol{x}_n^\top \end{bmatrix} \in \mathbb{R}^{n\times 3} \tag{2}$$

then the distance matrix $D = [D_{i,j}]_{1\leqslant i,j\leqslant n}$ can be written as

$$\boldsymbol{D} = \underbrace{\boldsymbol{d}_2\boldsymbol{e}^\top + \boldsymbol{e}\boldsymbol{d}_2^\top - 2\boldsymbol{X}\boldsymbol{X}^\top}_{\text{low rank}} \tag{3}$$

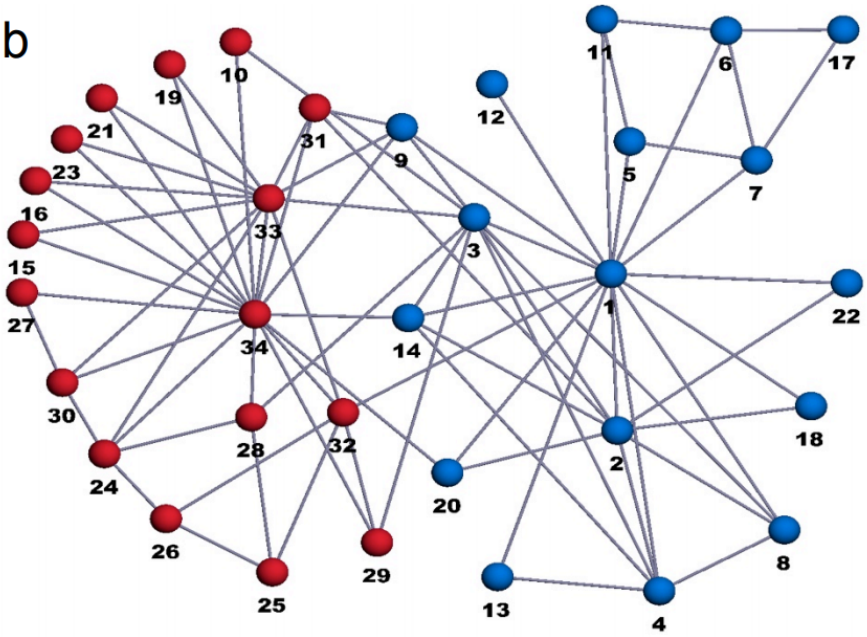
where $\boldsymbol{d}_2 := \left[\|\boldsymbol{x}_1\|^2, \cdots, \|\boldsymbol{x}_n\|^2\right]^\top$.

$\text{rank}(\boldsymbol{D}) \ll n \longrightarrow$ **low-rank matrix completion**

Application 2: Link Prediction

- Given a snapshot of a network $G = (V, E)$ (e.g., a Facebook social network among users)
- **Goal:** predict future possible links between any two users in the network

Zachary Karate Club



http://ifisc.uib-csic.es/~jramasco/ComplexNets.html

Application 3: Single-cell RNA-seq Data Imputation

- Given a gene expression matrix M of m genes and n cells
- **Dropout:** genes which are expressed even at a relatively high level may be undetected due to technical limitations
- **Goal:** impute the dropout events in M

	Cell 1	Cell2	Cell3	Cell4	...
Gene 1	0	0	3	10	
Gene 2	24	0	41	12	
Gene 3	175	284	93	162	
Gene 4	0	0	0	0	
Gene 5	36	0	32	21	
...	...	...	...	...	

Outline

1. What is Matrix Completion?
2. Exact Matrix Completion
3. Approximate Matrix Completion
   - Nuclear Norm Minimization
   - Low-Rank Matrix Factorization
   - Distributed Matrix Completion
4. Follow-up Research

Matrix Completion Problem

- Given a matrix M with some entries are missing
- **Goal:** Complete the matrix M

	1	2	3	4	5	6	7	8	9	10
1		4		2	4					
2	3		3	1			3	■	3	
3		3	2	3	■				4	
4			2	■		4		1	2	5
5	3			3		■	1			
6			2					3	2	■

- In general, it is impossible!!!
- But it can be possible with the **low-rank assumption**

Low-rank Assumption

Low-rank assumption

For a $n_1 \times n_2$ matrix M of rank r, assume that $\min(n_1, n_2) \gg r$.

- Why this assumption is needed?
- Assume M is a rating matrix for example,
  - Only a few factors contribute to anyone's taste or preference
- The matrix only has $(2n - r)r$ degrees of freedom (DoF) ($n = n_1 = n_2$).
- DoF is calculated by counting parameters in the SVD
  - (The number of singular values)
    + (degree of freedom of left singular vectors)
    + (degree of freedom of right singular vectors)
    $= r + ((2n - r - 1) \times r)/2 + ((2n - r - 1) \times r)/2$
- Considerably smaller than n^2.
  - With the low-rank assumption, DoF is reduced by about $2r/n$.

What Can Go Wrong?

Entire column missing

$$\begin{bmatrix} 1 & 2 & ? & 3 & \dots & 4 \\ 3 & 5 & ? & 4 & \dots & 1 \\ 2 & 5 & ? & 2 & \dots & 5 \end{bmatrix} \tag{4}$$

- No hope of recovery!

Standard solution: Uniform observation model

Assume that the set of s observed entries Ω is drawn uniformly at random:

$$\Omega \sim \text{Unif}(n_1, n_2, s) \tag{5}$$

What Can Go Wrong?

Bad spread of information

$$\mathbf{L} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{6}$$

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

Standard solution: Random orthogonal model

The family $\{u_k\}_{1 \leqslant k \leqslant r}$ is selected uniformly at random among all families of r orthonormal vectors, and similarly for the family $\{v_k\}_{1 \leqslant k \leqslant r}$. Here u_k and v_k are the left and right singular vectors.

EXACT Matrix Completion

Problem 1.1: Matrix Completion Problem

$$\begin{aligned} &\text{minimize} \ \ \text{rank}(X) \\ &\text{subject to} \ X_{ij} = M_{ij}, \quad (i,j) \in \Omega \end{aligned}$$

- Unfortunately, this problem is NP-hard and non-convex.
  - rank norm is not convex.
- All known algorithms require exponential time to n.

EXACT Matrix Completion

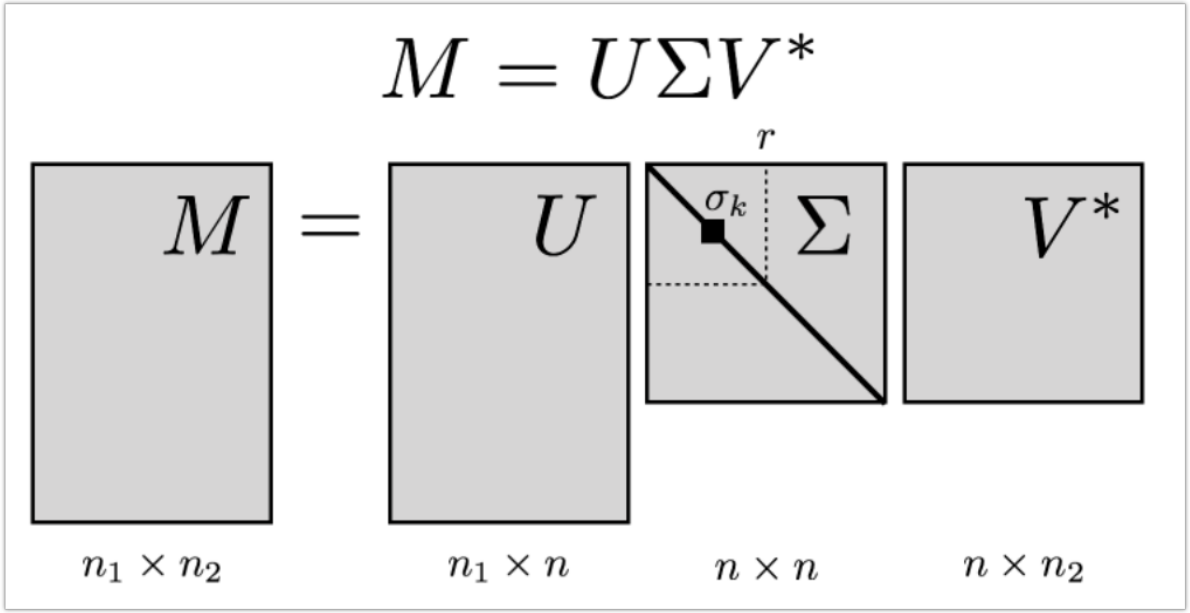
Problem 1.2: Convex Relaxation [Candes and Recht, 2009]

$$\begin{aligned} &\text{minimize } \|X\|_* \\ &\text{subject to } X_{ij} = M_{ij}, \quad (i,j) \in \Omega \end{aligned}$$

- The nuclear norm $\|X\|_*$ is the surrogate of the rank norm.
  - Also, the nuclear norm is a convex function.
  - This heuristic was introduced by [Recht, 2009].

Is Nuclear Norm Relaxation Reasonable?

- Recall that l_1-norm is the surrogate of l_0-norm.



- Let the vector $\sigma = (\sigma_1, \sigma_2, \cdots, \sigma_n)$, σ_i is the i-th singular value of the matrix M.
- The rank norm is the l_0-norm of the σ vector.
- The nuclear norm is the l_1-norm of the σ vector.

Connections with Compressed Sensing

General setup

Rank minimization

$$\begin{aligned} \text{minimize} \quad & \operatorname{rank}(X) \\ \text{subject to} \quad & \mathcal{A}(X) = b \end{aligned}$$

Convex relaxation

$$\begin{aligned} \text{minimize} \quad & \|X\|_* \\ \text{subject to} \quad & \mathcal{A}(X) = b \end{aligned}$$

Suppose $X = \operatorname{diag}(x), x \in \mathbb{R}^n$

- $\operatorname{rank}(X) = \sum_i 1_{(x_i \neq 0)} = \|x\|_{\ell_0}$
- $\|X\|_* = \sum_i |x_i| = \|x\|_{\ell_1}$

Rank minimization

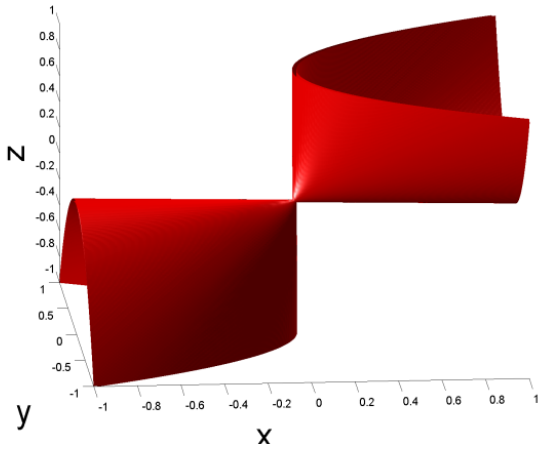
$$\begin{aligned} \text{minimize} \quad & \|x\|_{\ell_0} \\ \text{subject to} \quad & Ax = b \end{aligned}$$

Convex relaxation

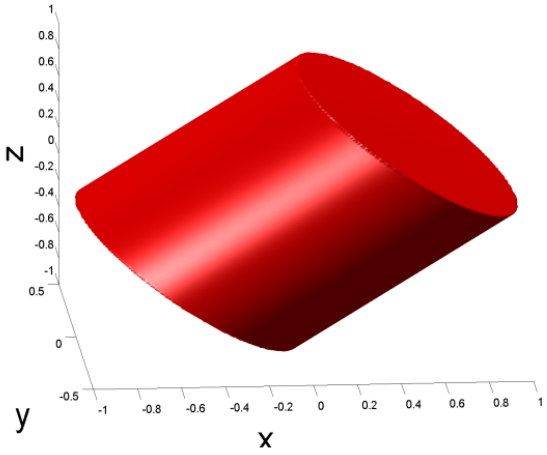
$$\begin{aligned} \text{minimize} \quad & \|x\|_{\ell_1} \\ \text{subject to} \quad & Ax = b \end{aligned}$$

This is compressed sensing!

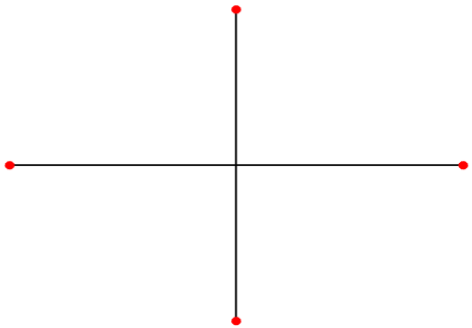
Connections with Compressed Sensing



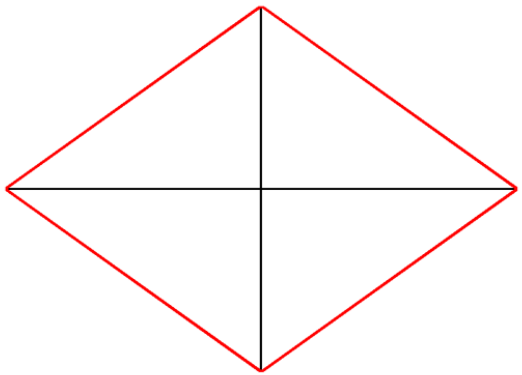
Rank penalty



Nuclear norm



ℓ_0 norm



ℓ_1 norm

Exact Unique Recovery Theorem

- If we want to recover a incomplete matrix, some assumptions are necessary.
- In fact, if some gentle conditions are met, this matrix can be recovered exactly and uniquely [Candes and Recht, 2009].
  - M: $n_1 \times n_2$ matrix of rank r, $n = \max(n_1, n_2)$, $m = |\Omega|$.

Theorem 1.1 (General)

- M obeys the random orthogonal model
- With uniformly random sampling assumption

Then, $\exists$ constants C, c such that if

$$m \geqslant Cn^{5/4} r \log n$$

the minimizer to the **Problem 1.2** is unique and equal to M with probability at least $1 - cn^{-3}$.

Exact Unique Recovery Theorem

- For the low-rank case, a tighter bound is given.

Theorem 1.1 (Low-rank)

if $r \leqslant n^{1/5}$, the recovery is exact with same probability provided that

$$m \geqslant Cn^{6/5} r \log n$$

How to Solve the Nuclear Norm Minimization?

- Convex Relaxation to Matrix Completion Problem (Nuclear Norm Minimization) can be solved by the Semi-definite Programming (SDP)
- Let SVD of matrix $X = U\Sigma V^*$
- Define $W_1 = U\Sigma U^*, W_2 = V\Sigma V^*, X' = \begin{bmatrix} W_1 & X \\ X^* & W_2 \end{bmatrix}$
- Under above settings, the following properties satisfy:
  - $\|X\|_* = \|W_1\|_* = \|W_2\|_*$
  - $X' \succeq 0$ (positive semidefinite matrix),
    $\because X' = \begin{bmatrix} U \\ V \end{bmatrix} \Sigma \begin{bmatrix} U \\ V \end{bmatrix}^* = \left(\begin{bmatrix} U \\ V \end{bmatrix} \sqrt{\Sigma} \right) \left(\begin{bmatrix} U \\ V \end{bmatrix} \sqrt{\Sigma} \right)^* \succeq 0$
  - $\forall$ symmetric matrix X, $\mathrm{trace}(X) = \|X\|_*$

How to Solve the Nuclear Norm Minimization?

- Therefore, the **Problem 1.2** can be reduced to

Reduced form of nuclear norm minimization

$$\begin{aligned} &\text{minimize} \ \ \text{trace}\,(X') \\ &\text{subject to} \ \mathcal{P}_\Omega(X) = \mathcal{P}_\Omega(M), X' \succeq 0 \end{aligned}$$

- $X' = \begin{bmatrix} W_1 & X \\ X^* & W_2 \end{bmatrix}$
- $\text{trace}\,(X') = \text{trace}\,(W_1) + \text{trace}\,(W_2) = \|X\|_* + \|X\|_* = 2\|X\|_*$

Now, the reduced form can be solved by SDP

- Many algorithms solving SDP have been proposed

How to Solve the Nuclear Norm Minimization?

- However, the off-the-shelf algorithms (such as interior point methods) are not directly amenable to large problems of this kind with over a million unknown entries.

- A faster and less memory-required algorithm is needed.

Singular Value Thresholding (SVT) Algorithm

- We consider the nuclear norm minimization problem,

$$\begin{array}{ll} \underset{X}{\text{minimize}} & \|X\|_* \\ \text{subject to} & \mathcal{P}_\Omega(X) = \mathcal{P}_\Omega(M) \end{array}$$

  with optimization variable $X \in R^{n_1 \times n_2}$.
- The SVT algorithm [Cai et al., 2010] can be stated as,

Singular Value Thresholding (SVT)

1. Input $\tau \geqslant 0$, $\{\delta_k\}_{k \geqslant 1}$, $Y^0 = 0$
2. $X^k = \mathcal{S}_\tau\left(Y^{k-1}\right)$
3. $Y^k = Y^{k-1} + \delta_k \mathcal{P}_\Omega\left(M - X^k\right)$

- $\mathcal{S}_\tau$ is the shrinkage operator.
- What's the meaning of the iteration?

Shrinkage Operator

For a matrix $Y \in \mathbb{R}^{n_1 \times n_2}$, consider:

$$\min_{X \in \mathbb{R}^{m \times n}} \tau \|X\|_* + \frac{1}{2}\|X - Y\|_F^2$$

The optimal solution is:

$$X := S_\tau(Y) = U\,\mathrm{Diag}\left(s_\tau(\sigma)\right) V^\top$$

- SVD: $Y = U\mathrm{Diag}(\sigma)V^T$
- $s_\tau(\sigma) = \max(\sigma - \tau, 0)$
- $S_\tau(Y)$ is a trade-off between minimize the rank and the distance with Y.

How to Understand SVT?

$$\begin{cases} \boldsymbol{X}^k = \mathcal{S}_\tau\left(\boldsymbol{Y}^{k-1}\right) \\ \boldsymbol{Y}^k = \boldsymbol{Y}^{k-1} + \delta_k \mathcal{P}_\Omega\left(\boldsymbol{M} - \boldsymbol{X}^k\right) \end{cases}$$

The meaning of the iterations is obvious,

- Proximity Optimization
  - The first step is a trade-off between minimize the rank of X^k and the distance with Y^k.
  - Only needs a SVD on a sparse matrix!!!
- Gradient Descent
  - The second step is a projected gradient descent step, so Y^k is gradually close to M.
  - Y^k is always sparse and easy to store.

Convergence Analysis

Theorem 3.1 [Cai et al., 2010]

The sequence $\{X^k\}$ generated by the Proximal gradient iterations converges to some $X^* \in \mathcal{X}^*$, where $\mathcal{X}^*$ is the optimal solution set.

Do We Need Exact Matrix Completion?

- We have recovered the matrix M with the SVT algorithm by solving:

$$\begin{aligned} \text{minimize} \quad & \|X\|_* \\ \text{subject to} \quad & P_\Omega(X) = P_\Omega(M) \end{aligned}$$

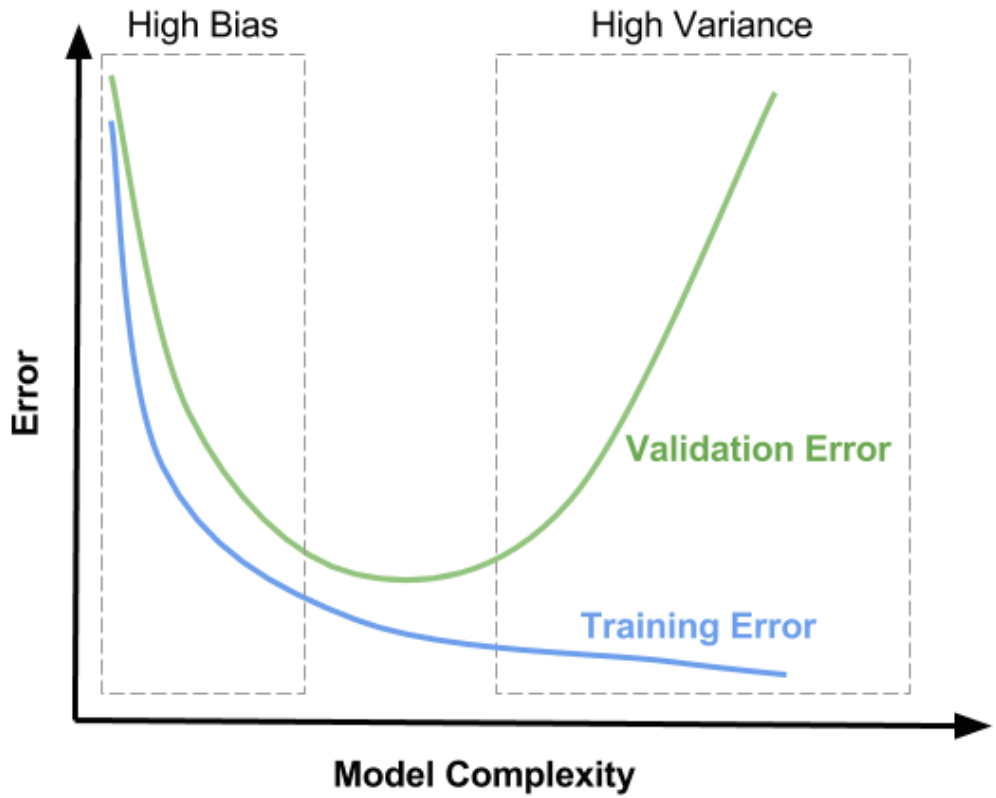
- $X = M$ in every observed entry
- But in real-life, there is always noise — fitting training data exactly incurs added variance
- Introduce bias to decrease variance like,

$$\begin{aligned} \text{minimize} \quad & \|X\|_* \\ \text{subject to} \quad & \|P_\Omega(M) - P_\Omega(X)\|_F \leqslant \delta \end{aligned}$$

where δ is a little positive number

Bias-Variance Trade-Off

- Low bias may incur high variance!!!



- We only need Approximate Matrix Completion!

Outline

1. What is Matrix Completion?
2. Exact Matrix Completion
3. Approximate Matrix Completion
   - Nuclear Norm Minimization
   - Low-Rank Matrix Factorization
   - Distributed Matrix Completion
4. Follow-up Research

Outline

1. What is Matrix Completion?
2. Exact Matrix Completion
3. Approximate Matrix Completion
   - Nuclear Norm Minimization
   - Low-Rank Matrix Factorization
   - Distributed Matrix Completion
4. Follow-up Research

Noise Assumption

Noise Assumption

For the observed $n_1 \times n_2$ matrix M of rank r, assume $M = X + E$, where X is the true data and E is the noise.

- Why this assumption is needed?
- For example in recommendation system,
  - We usually rate the movies only with scores 1-5
  - Our rating are influenced seriously by our mood
  - In many situations, we want to give a score between 3-4, for 3.6
  - Our rating are not that real
- Similar in other applications

Matrix Completion with Noise

SVT algorithm solves:

$$\begin{aligned} \text{minimize} \quad & \|X\|_* \\ \text{subject to} \quad & P_\Omega(M) = P_\Omega(X) \end{aligned}$$

there is always noise

$$M = X + E$$

Relax the constraint
[Mazumder et al., 2010]

$$\begin{aligned} \text{minimize} \quad & \|X\|_* \\ \text{subject to} \quad & \|P_\Omega(M) - P_\Omega(X)\|_F \leqslant \delta \end{aligned}$$

- First order algorithm to solve via a sparse SVD.
- No-noise reconstruction model seems too rigid.
- In real-life, there is noise - fitting training data exactly incurs added variance.
- Introduce bias to decrease variance.

The Lagrange Form

For a matrix $X_{n_1 \times n_2}$, let $\Omega \subset \{1, \ldots, n_1\} \times \{1, \ldots, n_2\}$ denote the indices of observed entries, let's consider the following problem:

$$\begin{array}{ll} \underset{X}{\text{minimize}} & \|X\|_* \\ \text{subject to} & \sum_{(i,j)\in\Omega} (X_{ij} - M_{ij})^2 \leqslant \delta \end{array} \qquad (*)$$

Equivalently we can reformulate (*) in the Lagrange form

$$\underset{X}{\text{minimize}} \sum_{(i,j)\in\Omega} (X_{ij} - M_{ij})^2 + \lambda \|X\|_* \qquad (**)$$

- Here $\lambda \geqslant 0$ is a regularization parameter controlling the nuclear norm of the minimizer $\hat{Z}_\lambda$ of $(*)$;
- There is a $1-1$ mapping between $\delta \geqslant 0$ and $\lambda \geqslant 0$ over their active domains.

Back to Shrinkage Operator

Let (fully observed) $M_{n_1 \times n_2}$ have SVD

$$M = U \cdot \text{diag}\left[\sigma_1, \ldots, \sigma_r\right] \cdot V^T$$

Consider the convex optimization problem

$$\underset{X}{\text{minimize}} \quad \|M - X\|_F^2 + \lambda \|X\|_*$$

Solution is shrinkage operator gets,

$$\mathcal{S}_\lambda(M) := U \cdot \text{diag}\left[\left(\sigma_1 - \lambda\right)_+, \ldots, \left(\sigma_r - \lambda\right)_+\right] \cdot V^T$$

Like lasso for SVD: singular values are shrunk to zero, with many set to zero. Smooth version of the best-rank approximation.

Soft-Impute Algorithm

Back to the nuclear norm minimization, in the lagrange form:

$$\underset{X}{\text{minimize}} \left\| P_\Omega(X) - P_\Omega(M) \right\|_F^2 + \lambda \|X\|_*$$

Soft-Impute [Mazumder et al., 2010]

1. Initialize $X^{\text{old}} = 0$ and create a decreasing grid $\wedge$ of values $\lambda_0 > \lambda_1 > \ldots > \lambda_r > 0$, with $\lambda_0 = \lambda_{\max}\left(P_\Omega(M)\right)$
2. For each $\lambda = \lambda_1, \lambda_2, \ldots \in \Lambda$ iterate $2\mathrm{a} - 2\mathrm{b}$ till convergence:
   - (2a) Compute $X^{\text{new}} \leftarrow \mathcal{S}_\lambda\left(P_\Omega(M) + P_\Omega^\perp\left(X^{\text{old}}\right)\right)$
   - (2b) Assign $X^{\text{old}} \leftarrow X^{\text{new}}$ and go to step $(2a)$
   - (2c) Assign $\hat{X}_\lambda \leftarrow X^{\text{new}}$ and go to 2
   - Output the sequence of solutions $\hat{X}_{\lambda_1}, \ldots, \hat{X}_{\lambda_K}$

Soft-Impute: Computational Bottleneck

Obtain the sequence $\{X_k\}$ of guesses

- Soft-Impute:
  $$X_{k+1} = \underset{X}{\operatorname{argmin}} \left\| P_\Omega(M) + P_\Omega^\perp(X_k) - X \right\|_F^2 + \lambda \|X\|_*$$
- SVT: $X_{k+1} = \underset{X}{\operatorname{argmin}} \|P_\Omega(X_k) - X\|_F^2 + \lambda \|X\|_*$

Computational bottleneck: Soft-Impute requires (low-rank) SVD of completed matrix in iterations:

$$\hat{X}_k = \mathcal{S}_\lambda(P_\Omega(M) + P_\Omega^\perp(X_k))$$

Trick:

$$P_\Omega(M) + P_\Omega^\perp(X_k) = \underbrace{\{P_\Omega(M) - P_\Omega(X_k)\}}_{\text{Sparse}} + \underbrace{X_k}_{\text{Low Rank}}$$

Computational Tricks in Soft-Impute

- Anticipate rank of $\hat{X}_{\lambda_{j+1}}$ based on rank of $\hat{X}_{\lambda_j}$, erring on generous side.
- Compute the low-rank SVD of $\hat{X}_k$ using orthogonal QR iterations with Reitz acceleration [Hastie et al., 2015].
- Iterations require left and right multiplications $U'\hat{X}_k$ and $\hat{X}_k V$. Ideal for **Sparse** $+$ **Low-rank structure**.
- Warm starts: $\mathbf{S}_\lambda\left(\hat{X}_k\right)$ provides excellent warm starts (U and V) for $\mathbf{S}_\lambda\left(\hat{X}_{k+1}\right)$. Likewise $\hat{Z}_{\lambda_j}$ for $\hat{Z}_{\lambda_{j+1}}$.
- Total cost per iteration $O[(m+n)\cdot r + |\Omega|]$ is less than that of the off-the-shelf methods.

Soft-Impute: Convergence Analysis

Theorem 3.2 [Mazumder et al., 2010]

Take $\lambda > 0$. The sequence of estimates $\{X_k\}_k$ given by:

$$X_{k+1} = \underset{Z}{\operatorname{argmin}} \left\| P_\Omega(M) + P_\Omega^\perp(X_k) - X \right\|_F^2 + \lambda \|X\|_* \tag{7}$$

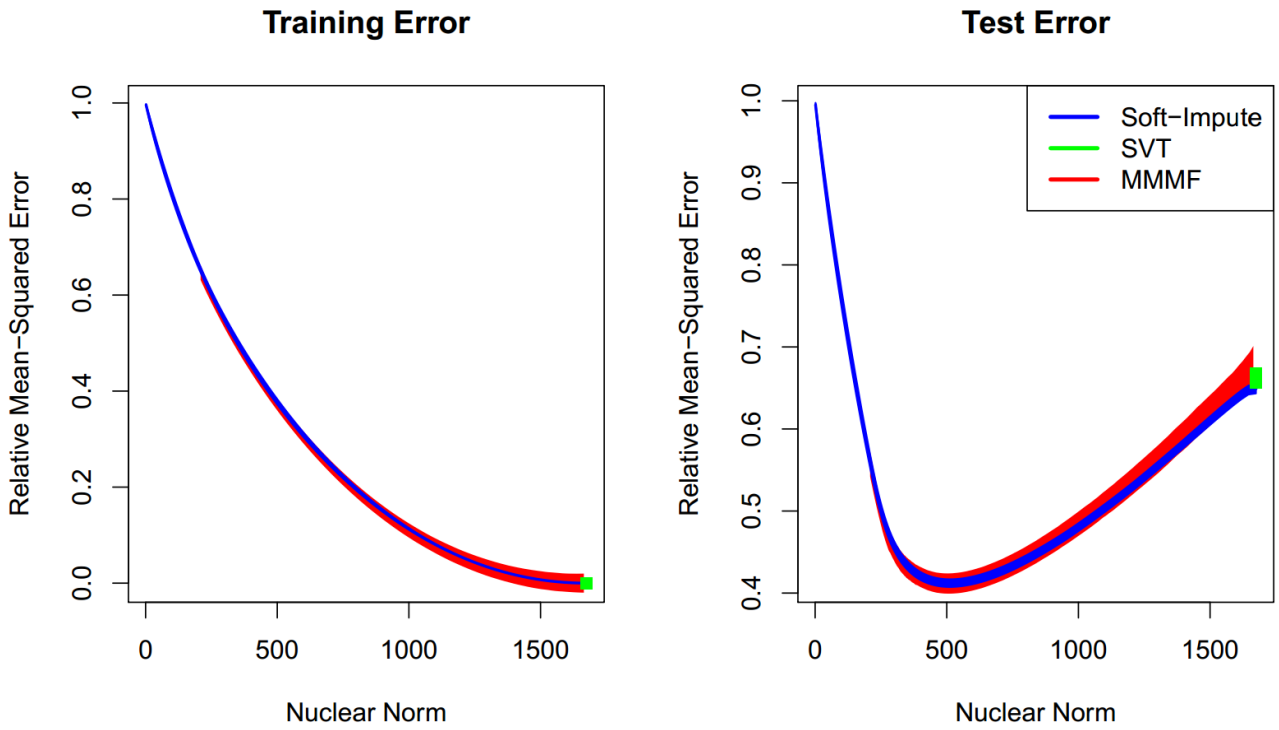
converges to X_∞, a fixed point of

$$X = \mathcal{S}_\lambda \left(P_\Omega(M) + P_\Omega^\perp(X) \right)$$

Hence, X_∞ minimizes

$$f_\lambda(X) := \|P_\Omega(M) - P_\Omega(X)\|_F^2 + \lambda \|X\|_*$$

Experimental Analysis



- MMMF (Maximum Margin Matrix Factorization) is a matrix factorization based method.
- SVT has very poor prediction error.
- Exactly fitting the training data is too rigid and overfitting.

Hard-Impute Algorithm

Let's consider another circumstance where X is low rank and its rank is known a priori,

$$\underset{\text{rank}(X)=r}{\text{minimize}} \left\| P_\Omega(X) - P_\Omega(M) \right\|_F$$

This is not convex in X, but by analogy with Soft-Impute, an iterative algorithm gives good solutions. Replace step:

$$Z^{\text{new}} \leftarrow \mathbf{S}_\lambda \left(P_\Omega(M) + P_\Omega^\perp \left(X^{\text{old}} \right) \right)$$

with

$$Z^{\text{new}} \leftarrow \mathbf{H}_r \left(P_\Omega(M) + P_\Omega^\perp \left(X^{\text{old}} \right) \right)$$

Here $\mathbf{H}_r(X^*)$ is the best rank-r approximation to X^*, i.e. the rank-r truncated SVD approximation.

Discussion

Nuclear norm minimization has many advantages,

- Convex problem
- No local minima

but,

- Storing a big dense matrix needs lots of memory

Since we assume it's a low rank or rank is known a priori, can we use less memory to store it?

Outline

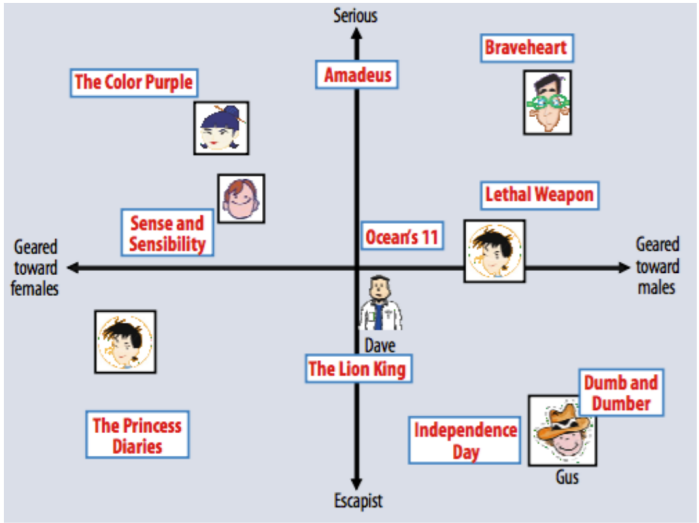
1. What is Matrix Completion?
2. Exact Matrix Completion
3. Approximate Matrix Completion
   - Nuclear Norm Minimization
   - Low-Rank Matrix Factorization
   - Distributed Matrix Completion
4. Follow-up Research

Latent Factor Model in Recommendation System

Assume that both movies and users live in some low dimensional space describing their properties,

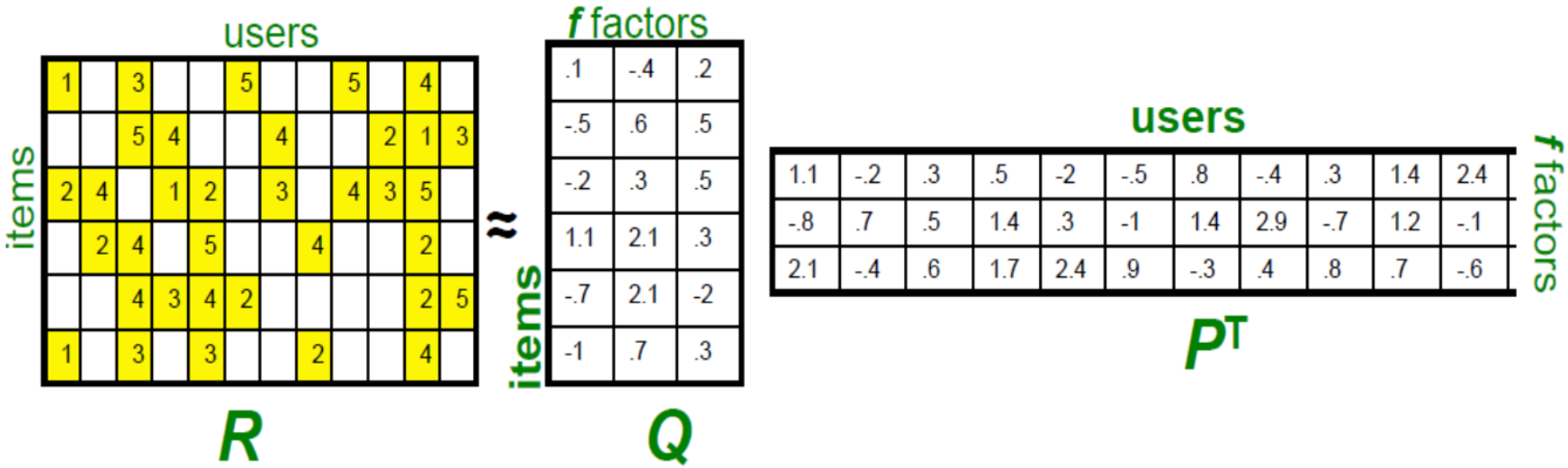
- characters, genres, tastes, ...

Recommend a movie based on its proximity to the user in the latent space.



Latent Factor Model in Recommendation System

- "SVD" on Netflix data: $R \approx Q \cdot P^T$



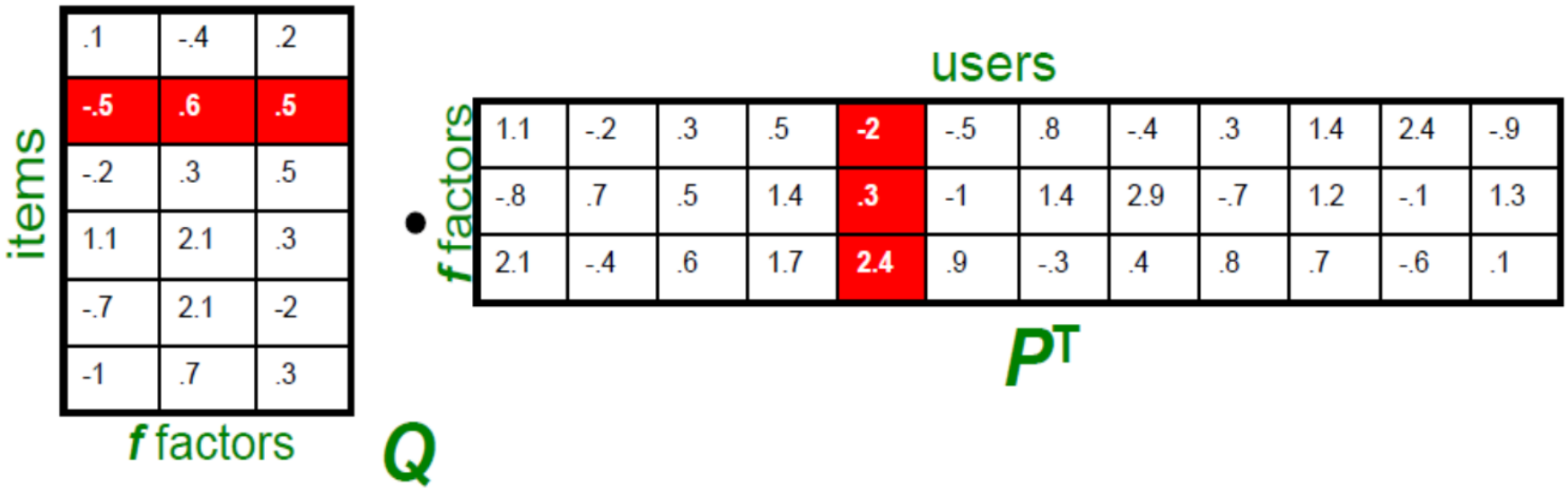
- For now let's assume we can approximate the rating matrix R as a product of "thin" $Q \cdot P^T R$ has missing entries, but let's ignore that for now! Basically, we will want the reconstruction error to be small on known ratings and we don't care about the values on the missing ones.

Ratings as Products of Factors

- How to estimate the missing rating of user x for item i?

$$\hat{r}_{xi} = q_i \cdot p_x^T = \sum_f q_{if} p_{xf}$$

where q_i is row i of Q and p_x is column x of P^T.



What is SVD?

Definition of SVD: if A is a real m-by-n matrix, then there exits $U = [u_1, \ldots, u_m] \in \mathbb{R}^{m \times m}$, and $V = [v_1, \ldots, v_n] \in \mathbb{R}^{n \times n}$, such that $U^T U = I$, $V^T V = I$ and $U^T A V = \text{diag}\,(\sigma_1, \ldots, \sigma_p) \in \mathbb{R}^{m \times n}$, $p = \min(m, n)$, where $\sigma_1 \geqslant \sigma_2 \geqslant \ldots \geqslant \sigma_p \geqslant 0$.

What is SVD?

Definition of SVD: if A is a real m-by-n matrix, then there exits $U = [u_1, \ldots, u_m] \in \mathbb{R}^{m \times m}$, and $V = [v_1, \ldots, v_n] \in \mathbb{R}^{n \times n}$, such that $U^T U = I$, $V^T V = I$ and $U^T A V = \text{diag}\,(\sigma_1, \ldots, \sigma_p) \in \mathbb{R}^{m \times n}$, $p = \min(m, n)$, where $\sigma_1 \geqslant \sigma_2 \geqslant \ldots \geqslant \sigma_p \geqslant 0$.

Theorem 3

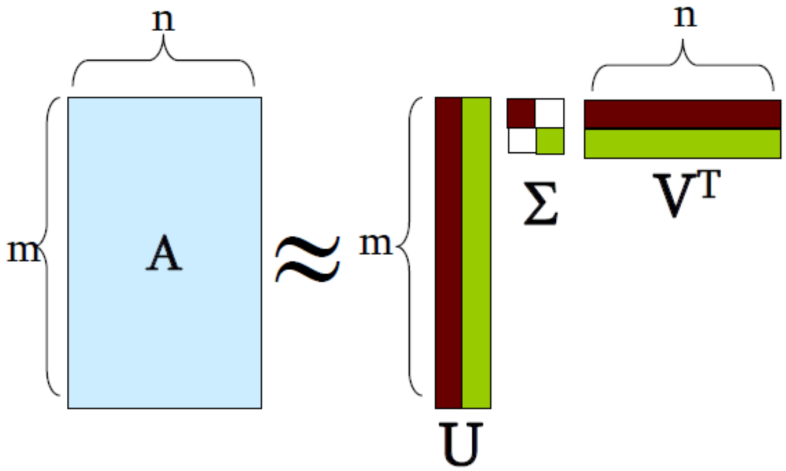
Let the SVD of $A \in \mathbb{R}^{m \times n}$ be given. If $k < r = \text{rank}(A)$ and $A_k = \sum_{i=1}^{k} \sigma_i u_i v_i^T$, then

$$\min_{\text{rank}(B)=k} \|A - B\|_2 = \|A - A_k\|_2 = \sigma_{k+1}$$

- The theorem says SVD gives a best solution minimizing mean square error given a certain rank.

What is SVD?

- A: Input data matrix
- U: Left singular vectors
- V: Right singular vectors
- $\sum$: Singular values



- SVD gives the minimum reconstruction error (SSE)!

$$\min_{U,V,\Sigma} \sum_{ij} \left(A_{ij} - \left[U \Sigma V^T \right]_{ij} \right)^2$$

- In our case, "SVD" on Netflix data: $R \approx Q \cdot P^T$, i.e.,$A = R$, $Q = U$, $P^T = V^T$
- But, we are not done yet! R has missing entries!

Regularized Matrix Factorization

- Minimize SSE on the training data!
- Use specialized methods to find P, Q such that $\hat{r}_{xi} = q_i \cdot p_x^T$

$$\min_{P,Q} \sum_{(i,x) \in \text{training}} \left(r_{xi} - q_i \cdot p_x^T\right)^2$$

We don't require cols of P, Q to be orthogonal/unit length.

- P, Q map users/movies to a latent space
- Add regularization [Srebro et al., 2004]:

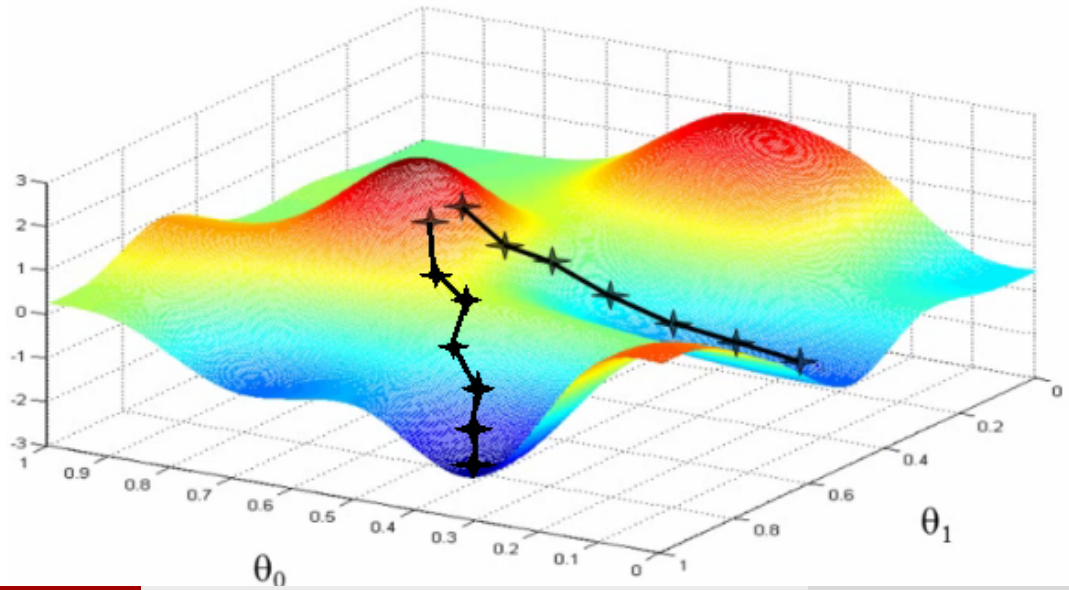
$$\min_{P,Q} \sum_{(i,x) \in \text{training}} \left(r_{xi} - q_i \cdot p_x^T\right)^2 + \lambda \left[\sum_x \|p_x\|_2^2 + \sum_i \|q_i\|_2^2\right]$$

λ is called regularization parameters.

How to Optimize?

$$\min_{P,Q} \sum_{(i,x)\in \text{ training}} \left(r_{xi} - q_i \cdot p_x^T\right)^2 + \lambda \left[\sum_x \|p_x\|_2^2 + \sum_i \|q_i\|_2^2\right]$$

- Non-convex, exactly biconvex: convex to p or q but non-convex to (p,q).
- Lots of local minimas, which means initializations are important.



Biconvex Set

Let $X \subseteq \mathbb{R}^n$ and $Y \subseteq \mathbb{R}^m$ be two non-empty, convex sets, and let $B \subseteq X \times Y$. We define x- and y-sections of B as follows:

$$B_x := \{y \in Y : (x, y) \in B\}$$
$$B_y := \{x \in X : (x, y) \in B\}$$

Definition: Biconvex set

The set $B \subseteq X \times Y$ is called a biconvex set on $X \times Y$ or biconvex for short, if B_x is convex for every $x \in X$ and B_y is convex for every $y \in Y$.

Biconvex Function

Definition: Biconvex function

A function $f : B \to \mathbb{R}$ on a biconvex set $B \subseteq X \times Y$ is called a biconvex function on B or biconvex for short, if

$$f_x(\bullet) := f(x, \bullet) : B_x \to \mathbb{R}$$

is a convex function on B_x for every fixed $x \in X$ and

$$f_y(\bullet) := f(\bullet, y) : B_y \to \mathbb{R}$$

is a convex function on B_y for every fixed $y \in Y$.

Alternating Least Square (ALS)

Since the problem is convex to P and Q separately, so we can use the alternating least square (ALS) method,

- Repeat
  - Fix P and solve $\min_Q \sum_{i \in \text{ training}} \left(r_{xi} - q_i \cdot p_x^T\right)^2 + \lambda \left[\sum_i \|q_i\|_2^2\right]$
  - Fix Q and solve $\min_P \sum_{x \in \text{ training}} \left(r_{xi} - q_i \cdot p_x^T\right)^2 + \lambda \left[\sum_x \|p_x\|_2^2\right]$

Alternating Least Square (ALS)

Since the problem is convex to P and Q separately, so we can use the alternating least square (ALS) method,

- Repeat
  - Fix P and solve $\min_Q \sum_{i \in \text{ training}} \left(r_{xi} - q_i \cdot p_x^T\right)^2 + \lambda \left[\sum_i \|q_i\|_2^2\right]$
  - Fix Q and solve $\min_P \sum_{x \in \text{ training}} \left(r_{xi} - q_i \cdot p_x^T\right)^2 + \lambda \left[\sum_x \|p_x\|_2^2\right]$

Here, the ALS algorithm is as follows,

$$q_i = \left(\sum_{r_{xi} \in r_{*i}} p_x p_x^\top + \lambda I_k\right)^{-1} \sum_{r_{xi} \in r_{*i}} r_{xi} p_x$$
$$p_x = \left(\sum_{r_{xi} \in r_{x*}} q_i q_i^\top + \lambda I_k\right)^{-1} \sum_{r_{xi} \in r_{x*}} r_{xi} q_i$$

- Involve inverting a matrix.
- Not only computationally expensive but also numerically unstable.

The Gradient Descent Method

$$\min_{P,Q} F(P,Q) := \sum_{(i,x)\in \text{training}} \left(r_{xi} - q_i \cdot p_x^T\right)^2 + \lambda \left[\sum_x \|p_x\|_2^2 + \sum_i \|q_i\|_2^2\right]$$

Gradient descent:

- Initialize P and Q (using SVD, pretend missing ratings are 0).
- Do gradient descent:
  - $P^{k+1} \leftarrow P^k - \tau \nabla_P F\left(P^k, Q^k\right)$
  - $Q^{k+1} \leftarrow Q^k - \tau \nabla_Q F\left(P^k, Q^k\right)$

  where $\left(\nabla_Q F\right)_{if} = -2 \sum_{x,i} \left(r_{xi} - q_i p_x^T\right) p_{xf} + 2\lambda q_{if}$.

Computing gradients is slow when the dimension is huge.

The Stochastic Gradient Descent (SGD)

Let q_{if} be entry f of row q_i of matrix Q

$$\left(\nabla_Q F\right)_{if} = \sum_{x,i} \left(-2\left(r_{xi} - q_i p_x^T\right) p_{xf} + 2\lambda q_{if}\right) = \sum_{x,i} \nabla_Q F\left(r_{xi}\right)$$

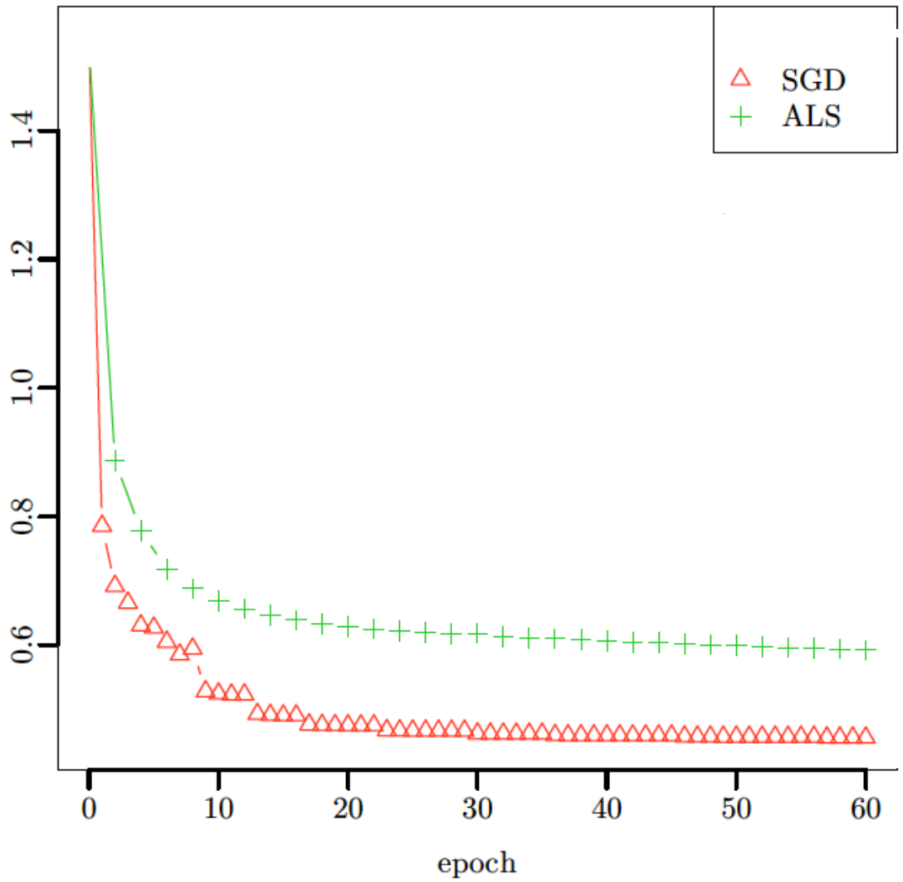
$$\left(\nabla_P F\right)_{xf} = \sum_{x,i} \left(-2\left(r_{xi} - q_i p_x^T\right) q_{xf} + 2\lambda p_{if}\right) = \sum_{x,i} \nabla_P F\left(r_{xi}\right)$$

Stochastic gradient decent:

- Instead of evaluating gradient over all ratings, evaluate it for each individual rating and make a step
  - $P \leftarrow P - \tau \nabla_P F\left(r_{xi}\right)$
  - $Q \leftarrow Q - \tau \nabla_Q F\left(r_{xi}\right)$
- More steps but each step is computed much faster.

Comparison of Different Optimization Algorithms

Small steps! Big Difference!!!



Connections with the Probabilistic Matrix Factorization (PMF)

- PMF [Salakhutdinov, 2008] is a simple probabilistic linear model with the Gaussian observation noise.
- Given the feature vectors for the user and the movie, the distribution of the corresponding rating is:

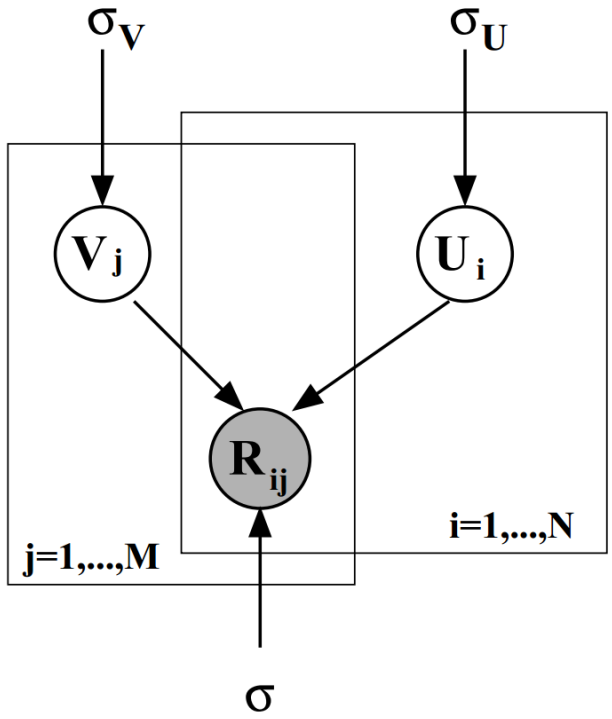
$$p\left(R_{ix}|Q_i, P_x, \sigma^2\right) = \mathcal{N}\left(R_{ix}|Q_i P_x^T, \boldsymbol{\sigma}^2\right)$$

- The user and movie feature vectors are given zero-mean spherical Gaussian priors:

$$p\left(Q|\sigma_q^2\right) = \prod_{i=1}^{N} \mathcal{N}\left(U_i|0, \sigma_q^2 \mathrm{I}\right), \quad p\left(P|\sigma_p^2\right) = \prod_{x=1}^{M} \mathcal{N}\left(P_x|0, \sigma_V^2 \mathrm{I}\right)$$

The Graphical Model View for the PMF

- MAP Learning: Maximize the log posterior over movie and user features with fixed hyperparameters.
- Equivalent to minimizing the sum-of-squared-errors with quadratic regularization terms:



$$E = \frac{1}{2}\sum_{i=1}^{N}\sum_{x=1}^{M} I_{ix}\left(R_{ix} - q_i p_x^T\right)^2 + \frac{\lambda_q}{2}\sum_{i=1}^{N}\|q_i\|_{Fro}^2 + \frac{\lambda_p}{2}\sum_{x=1}^{M}\|p_x\|_{Fro}^2$$

$\lambda_q = \sigma^2/\sigma_q^2, \lambda_p = \sigma^2/\sigma_{p'}^2$, and $I_{ix} = 1$ if user i rated movie x and is 0 otherwise.

MAP versus Regularized Least-Squares

- MAP under Gaussian Model:

$$E = \frac{1}{2}\sum_{i=1}^{N}\sum_{x=1}^{M} \boldsymbol{I}_{ix}\left(R_{ix} - q_i p_x^T\right)^2 + \frac{\lambda_q}{2}\sum_{i=1}^{N}\|q_i\|_{Fro}^2 + \frac{\lambda_p}{2}\sum_{x=1}^{M}\|p_x\|_{Fro}^2$$

- Least-squares matrix completion with L_2 regularization:

$$\min_{Q,P} \frac{1}{2}\sum_{r_{xi}}\left(q_i \cdot p_x^T - r_{xi}\right)^2 + \frac{\lambda_q}{2}\|Q\|_F^2 + \frac{\lambda_p}{2}\|P\|_F^2$$

- Understanding as a probabilistic model is very useful!
  - Change priors.
  - Incorporate other sources of information or dependencies.

A Probabilistic Model on Matrix Factorization

- Imposing different priors on U and V and assuming the underlying distribution of the residual will lead to different models.

$$X = UV + \varepsilon$$

- Probabilistic PCA [Tipping and Bishop, 1997]: isotropic Gaussian prior on V, ε follows i.i.d Gaussian.
- GLAD [Saddiki et al., 2014]: Laplace prior on U and Dirichlet prior on V, ε follows i.i.d Gaussian.
- Probabilistic approach provides a flexible way to address the uncertain nature of the data!!!

Summary

- In the matrix factorization model, the matrix is assumed to be low-rank.
- For example, in the recommendation system, we think users' tastes are decided by several features,
  - genders, characters, locations, ...
- Does it make sense to subtract 2 characters?
- Nonnegativity seems to be necessary!!!

Data is often Nonnegative by Nature

- pixel intensities
- amplitude spectra
- occurrence counts
- food or energy consumption
- user scores
- stock market values
- . . .

For the sake of interpretability, optimal processing of nonnegative data may call for processing under nonnegativity constraints.

Data is often Nonnegative by Nature

- pixel intensities
- amplitude spectra
- occurrence counts
- food or energy consumption
- user scores
- stock market values
- . . .

For the sake of interpretability, optimal processing of nonnegative data may call for processing under nonnegativity constraints. **Nonnegative Matrix Factorization (NMF)** provides an unsupervised linear representation of the data [Lee and Seung, 1999].

NMF vs SVD

Property	NMF	SVD
Formulation	A=WH	A=U$\sum V^T$
Optimality (in terms of squared distance)	×	✓
Speed and robustness	×	✓
Uniqueness	×	✓
Sensitivity to initialization	✓	×
Orthogonality	×	✓
Sparsity	✓	×
Non-nongativity	✓	×
Interpretability	✓	×

NMF Completion (NMFC)

Model [Xu et al., 2012]:

$$\begin{array}{ll} \min & \|\mathcal{P}_\Omega(XY - M)\|_F \\ \text{s.t.} & X \geqslant 0, Y \geqslant 0 \end{array} \iff \begin{array}{ll} \min & \frac{1}{2}\|XY - Z\|_F^2 \\ \text{s.t.} & X = U, Y = V \\ & U \geqslant 0, V \geqslant 0 \\ & \mathcal{P}_\Omega(Z - M) = 0 \end{array}$$

Augmented Lagrangian function:

$$\mathcal{L}_A(X, Y, Z, U, V, \Lambda, \Pi) = \frac{1}{2}\|XY - Z\|_F^2 + \Lambda \bullet (X - U) + \Pi \bullet (Y - V) + \frac{\alpha}{2}\|X - U\|_F^2 + \frac{\beta}{2}\|Y - V\|_F^2$$

where $A \bullet B := \sum_{i,j} a_{ij} b_{ij}$.

Optimization

Lots of algorithms for NMF have been developed,

- Projected Gradient
- Active Set
- Nesterov's optimal gradient
- Proximal Alternating Non-negative Least Square

The matrix is incomplete. Not all exiting algorithms are applicable.

Optimization

Lots of algorithms for NMF have been developed,

- Projected Gradient
- Active Set
- Nesterov's optimal gradient
- Proximal Alternating Non-negative Least Square

The matrix is incomplete. Not all exiting algorithms are applicable.

- **The Alternating Direction Method of Multipliers (ADMM)**
  - with good robustness of method of multipliers
  - which can support decomposition
- "robust dual decomposition" or "decomposable method of multipliers".
- proposed by Gabay, Mercier, Glowinski, Marrocco in 1976.

Basic Idea of ADMM

- ADMM problem form (with f, g convex),

$$\begin{array}{ll} \text{minimize} & f(x) + g(z) \\ \text{subject to} & Ax + Bz = c \end{array}$$

  - 2 sets of variables, with separable objective.

- $L_\rho(x, z, y) = f(x) + g(z) + y^T(Ax + Bz - c) + (\rho/2)\|Ax + Bz - c\|_2^2$
- ADMM:

$$\begin{aligned} x^{k+1} &:= \text{argmin}_x \, L_\rho\left(x, z^k, y^k\right) && \text{// x-minimization} \\ z^{k+1} &:= \text{argmin}_z \, L_\rho\left(x^{k+1}, z, y^k\right) && \text{// z-minimization} \\ y^{k+1} &:= y^k + \rho\left(Ax^{k+1} + Bz^{k+1} - c\right) && \text{// dual update} \end{aligned}$$

ADMM for NMFC

$$
\begin{aligned}
X_{k+1} &= \left(Z_k Y_k^T + \alpha U_k - \Lambda_k\right)\left(Y_k Y_k^T + \alpha I\right)^{-1} \\
Y_{k+1} &= \left(X_{k+1}^T X_{k+1} + \beta I\right)^{-1}\left(X_{k+1}^T Z_k + \beta V_k - \Pi_k\right) \\
Z_{k+1} &= X_{k+1} Y_{k+1} + \mathcal{P}_\Omega\left(M - X_{k+1} Y_{k+1}\right) \\
U_{k+1} &= \mathcal{P}_+\left(X_{k+1} + \Lambda_k/\alpha\right) \\
V_{k+1} &= \mathcal{P}_+\left(Y_{k+1} + \Pi_k/\beta\right) \\
\Lambda_{k+1} &= \mathcal{N}_k + \gamma\alpha\left(X_{k+1} - U_{k+1}\right) \\
\Pi_{k+1} &= \Pi_k + \gamma\beta\left(Y_{k+1} - V_{k+1}\right)
\end{aligned}
$$

where $\left(\mathcal{P}_+(A)\right)_{ij} = \max\{a_{ij}, 0\}$.

Outline

1. What is Matrix Completion?
2. Exact Matrix Completion
3. Approximate Matrix Completion
   - Nuclear Norm Minimization
   - Low-Rank Matrix Factorization
   - Distributed Matrix Completion
4. Follow-up Research

Distributed Matrix Completion

- Real applications can be large,
  - Millions of users, Millions of items, Billions of ratings.
  - e.g., Netflix: $\geqslant$20M users, $\geqslant$20k movies, $\geqslant$4B ratings (projected).

- Scalable algorithms are necessary!!!
- Existing MapReduce algorithms, e.g.,
  - DALS [Zhou et al., 2008]
  - DSGD-MR [Gemulla et al., 2011]
  - ASGD [Yun et al., 2013]
  - DSGD++ [Gemulla et al., 2011]
  - . . .

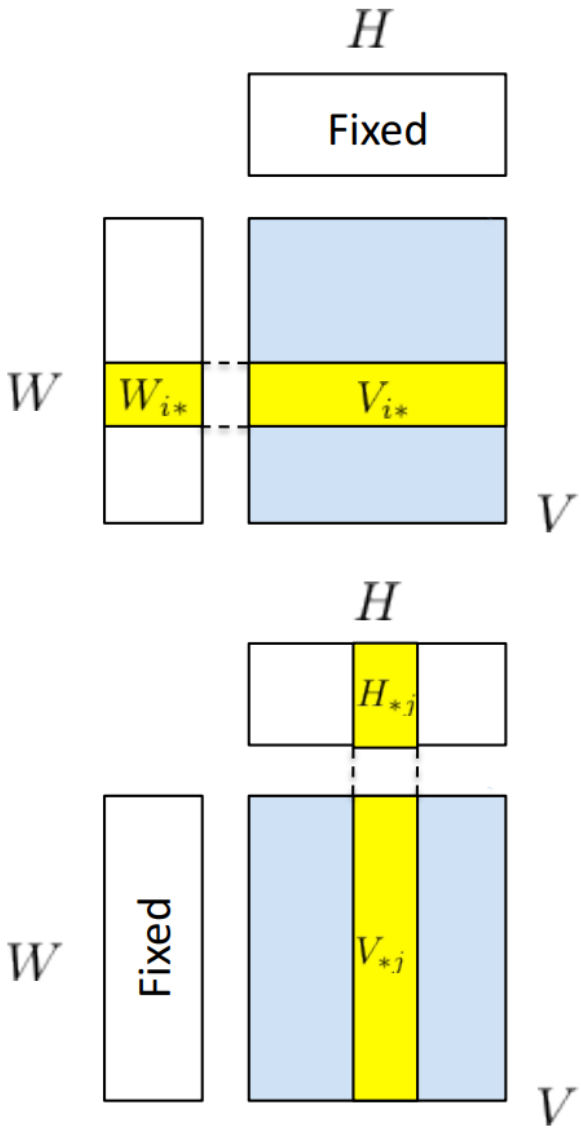
Distributed Alternating Least Squares (DALS)

Alternate

- Fix H optimize for W.
- Fix W optimize for H.
- For each column/row: solve a least squares problem.

Drawbacks

- Slow (cubic in rank).
- Memory intensive (stores data matrix twice).



SGD for Matrix Factorization

- Set $\theta = (W; H)$ and use

$$L(\theta) = \sum_{(i,j)\in Z} L_{ij}\left(\mathbf{W}_{i*}, \mathbf{H}_{*j}\right)$$

$$L'(\theta) = \sum_{(i,j)\in Z} L'_{ij}\left(\mathbf{W}_{i*}, \mathbf{H}_{*j}\right)$$

$$\hat{L}'(\theta, z) = NL'_{i,j_z}\left(\mathbf{W}_{i_z*}, \mathbf{H}_{*_j}\right)$$

  where N is all the observed values.
- SGD epoch
  1. Pick a random entry $z \in Z$
  2. Compute approximate gradient $\hat{L}'(\theta, z)$
  3. Update parameters

$$\theta_{n+1} = \theta_n - \epsilon_n \hat{L}'\left(\theta_n, z\right)$$

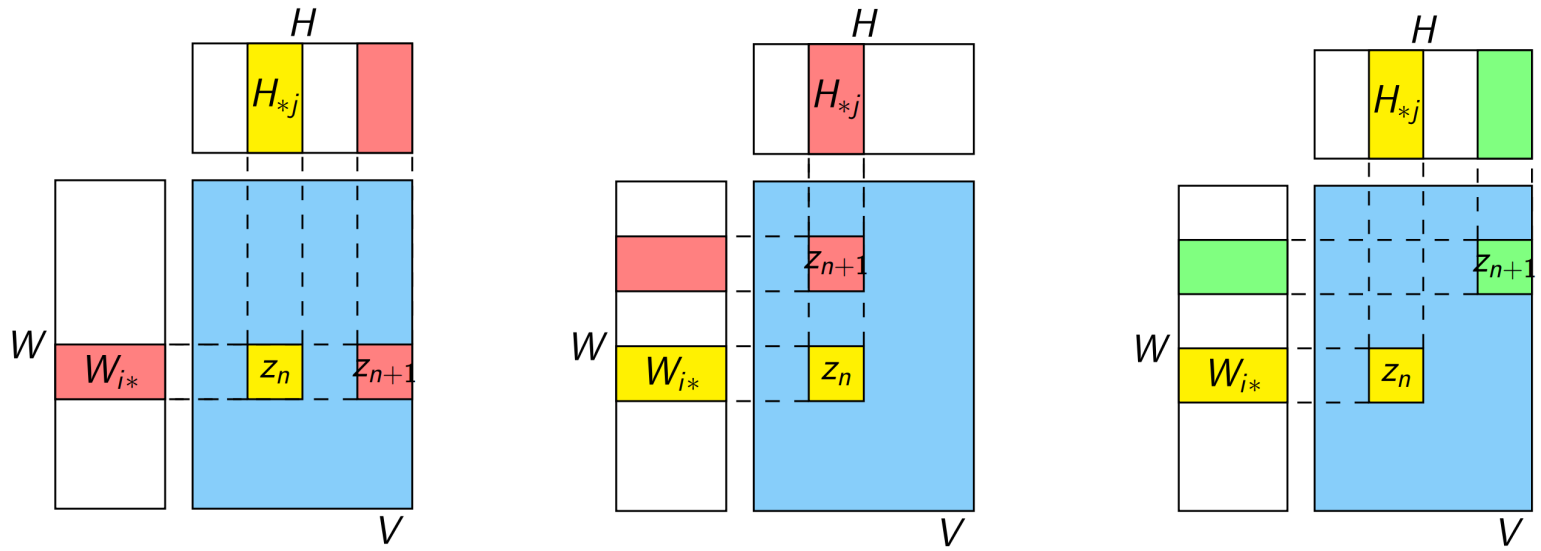
  4. Repeat N times

Distributed SGD (DSGD)

- SGD steps depend on each other,

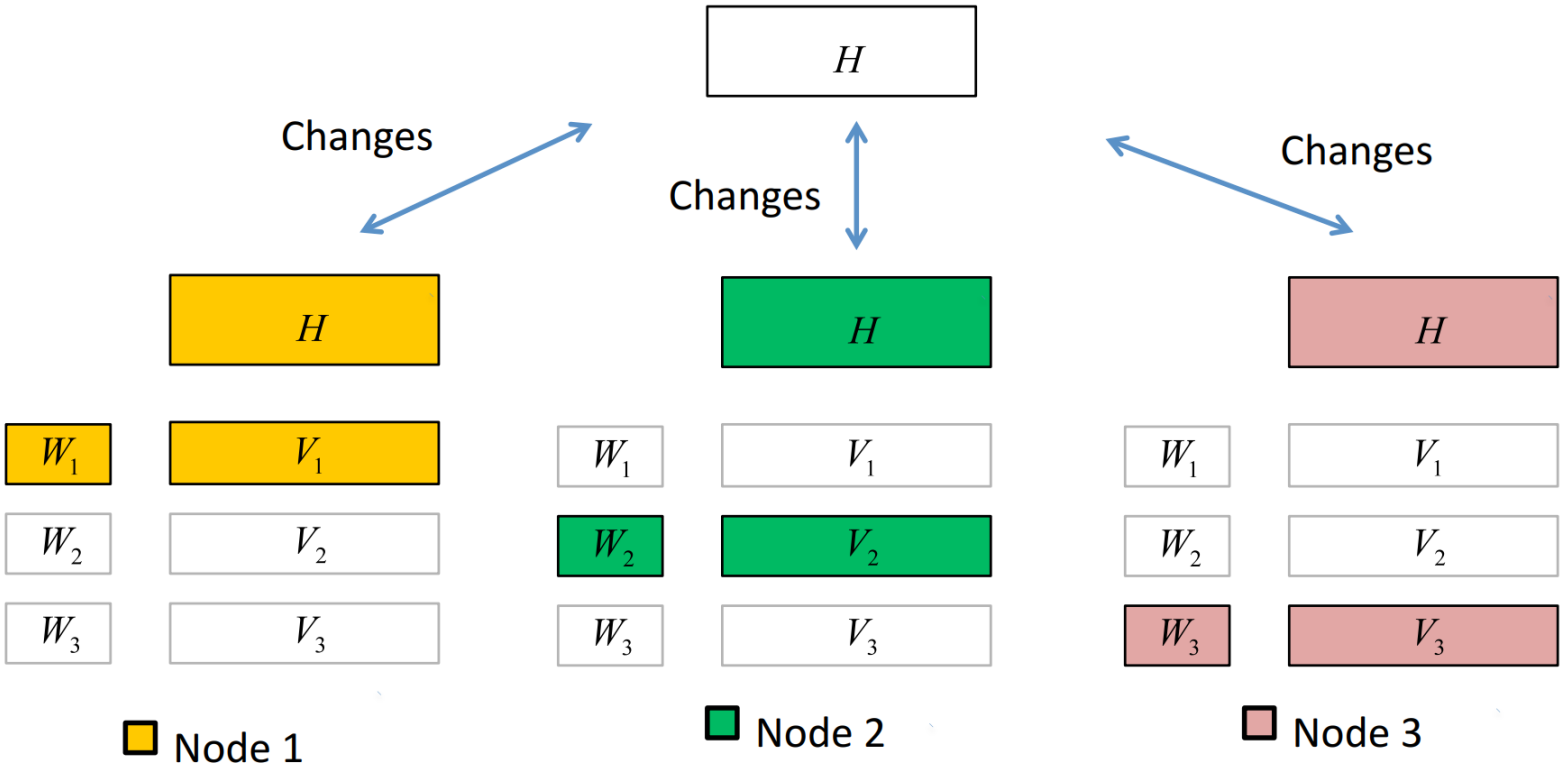
$$\theta_{n+1} = \theta_n - \epsilon_n \hat{L}'(\theta_n)$$

- An SGD step on example $z \in Z$,
  - Reads $W_{i_z *}$ and $H_{* j_z}$
  - Performs gradient computation $L'_{ij}\left(W_{i_z *}, H_{* * j_z}\right)$
  - Updates $W_{i_z *}$ and $H_{* j_z}$
- Not all steps are dependent!



Asynchronous SGD (ASGD)

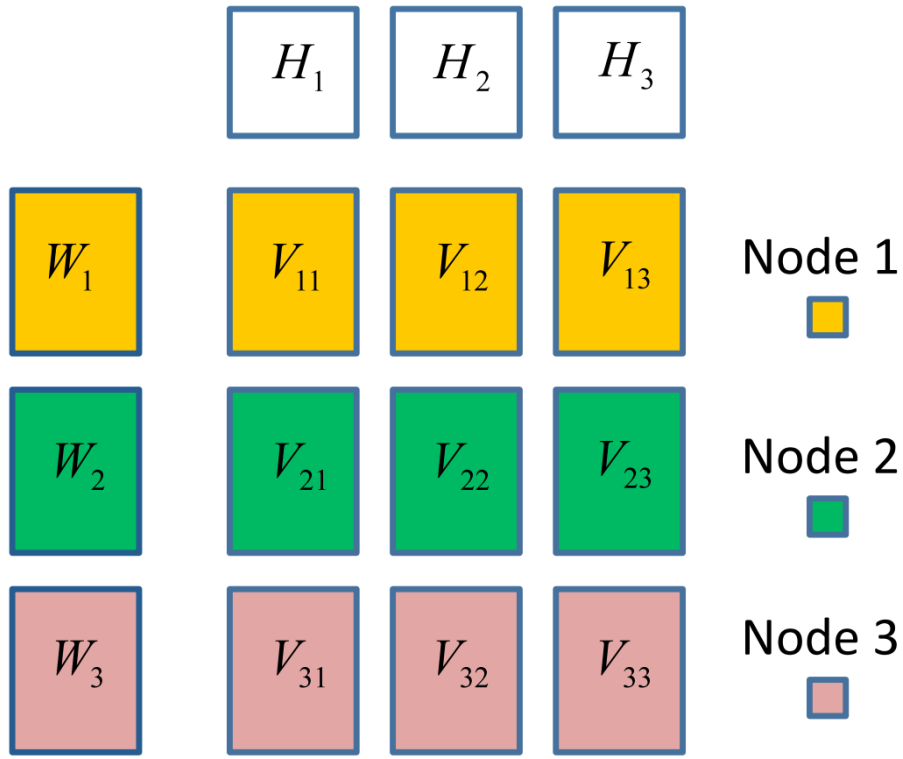
- Each node works on a local copy of the movies matrix H.
- Local copies are synchronized continuously.



Distributed SGD-MapReduce (DSGD-MR)

Block and distribute V

1. Pick a "diagonal".
2. Run SGD on the diagonal (in parallel).
3. Merge the results.
4. Move on to next "diagonal".

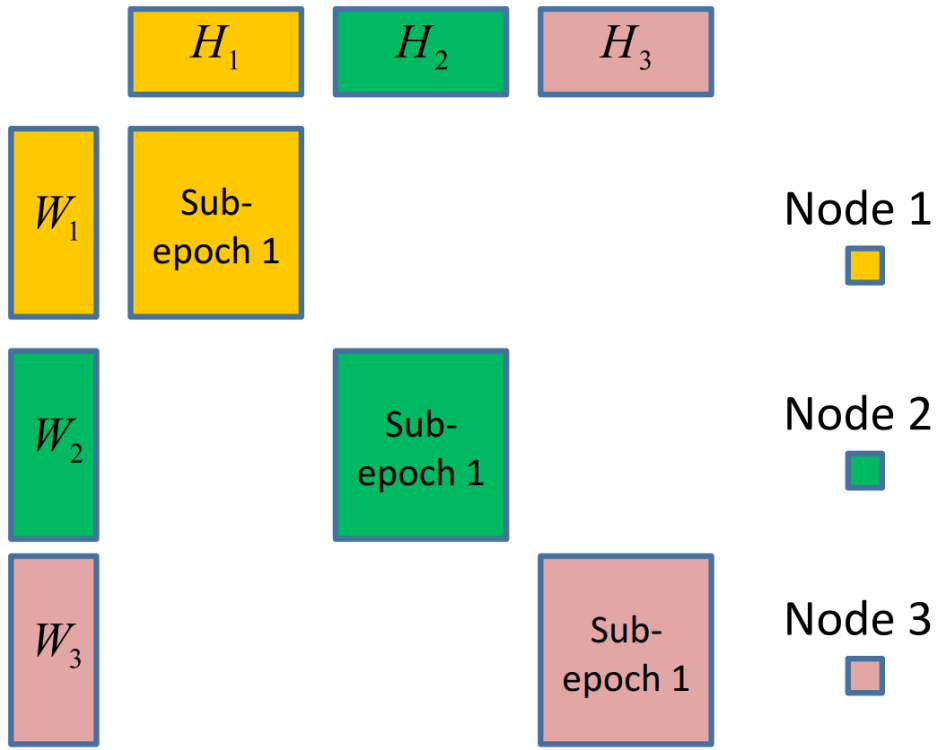


DSGD++: Direct Communication Between Nodes

How to do better in a shared-nothing environment?

DSGD-MR drawbacks:

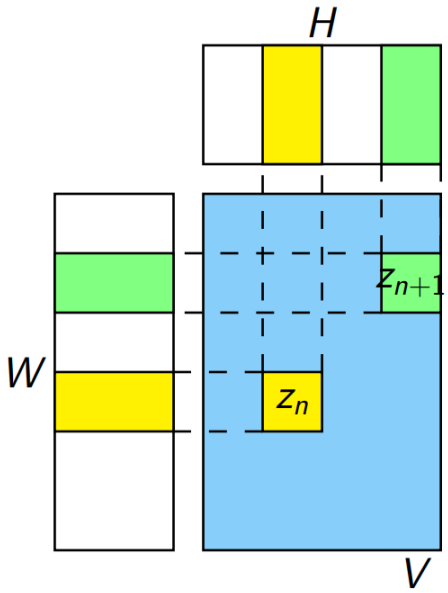
- Repeatedly reads/writes from/to disk
- Synchronous
- No overlapping of communication and computation
- No shared memory

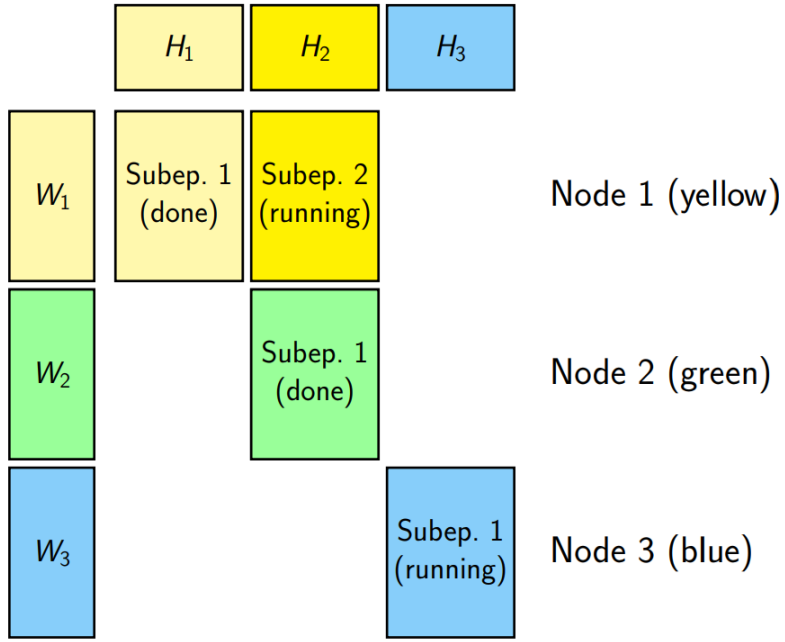


DSGD++

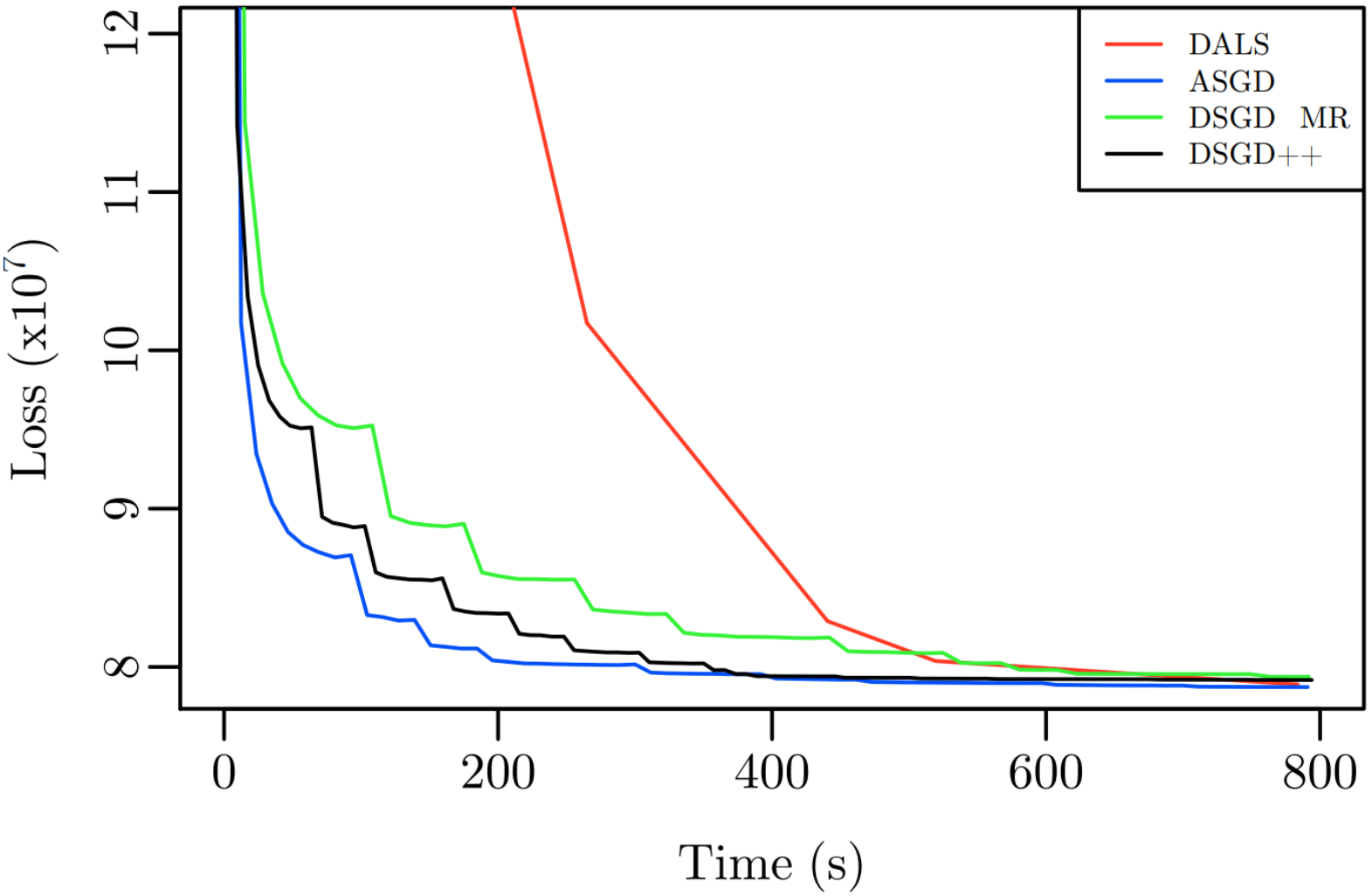
How to do better in a shared-nothing environment?

- Prefetch data/parameters for next SGD step(s).
- Exploit multi-core.
- Directly communicate parameters between nodes.
- Overlay subepochs.
- Overlay computation and communication.





Experiments with Netflix Data



Extended Studies

- There have been lots of techniques and studies on distributed matrix completion,
  - distributed non-negative matrix factorization [Liu et al., 2016]
  - distributed nuclear norm minimization [Mardani et al., 2012]
- Because of the variable separation property, ADMM has been adopted in distributed matrix factorization,
  - A distributed NMF with an ADMM algorithm [Sun and Fevotte, 2014]
  - A stochastic ADMM to matrix factorization [Yu et al., 2014]
  - . . .

Nuclear Norm Minimization vs Direct (Bilinear) Low Rank Solutions

- Nuclear norm minimization:

$$\min_X f(X - M) + \lambda \|X\|_* \qquad (*)$$

  - convex, global optima, close to truth.
  - X is very large, solvers are very slow and non-scalable.

- Low rank approximation:

$$\min_{U,V} f\left(M - UV^\top\right) + \frac{\lambda}{2}\left(\|U\|_F^2 + \|V\|_F^2\right) \qquad (**)$$

  - non-convex, many local optima, non-robust.
  - less parameters, faster solvers.

Nuclear Norm Minimization vs Direct (Bilinear) Low Rank Solutions

In fact, under some conditions, the local optima of (**) are global optima of (*).

[Boumal et al., 2016]

Let $f(X - M)$ be convex in M and M^* be an optimal solution of $(*)$ with $\text{rank}\,(M^*) = k^*$. Then, any solution $M = UV^\top$ of $(**)$ with $r \geqslant k^*$ is a solution of $(*)$.

Summary

- Though nuclear norm minimization is convex, it's largely limited by its memory requirements and nonscalable algorithms.
- Matrix Completion based on matrix factorization has made great success in last 10 years!
- Sometimes local minima is good enough for our problem.
- The winner of Netflix Prize win 1000000 dollars relying on matrix factorization.
- Matrix factorization is still very instructive in many domains, e.g., clustering,dimension reduction etc.

Outline

1. What is Matrix Completion?
2. Exact Matrix Completion
3. Approximate Matrix Completion
   - Nuclear Norm Minimization
   - Low-Rank Matrix Factorization
   - Distributed Matrix Completion
4. Follow-up Research
   - Graph-regularized Matrix Completion
   - Link Prediction on Graphs with Deep Learning
   - Matrix Completion under Missing Mechanisms

Outline

1. What is Matrix Completion?
2. Exact Matrix Completion
3. Approximate Matrix Completion
4. Follow-up Research
   - Graph-regularized Matrix Completion
   - Link Prediction on Graphs with Deep Learning
   - Matrix Completion under Missing Mechanisms

Build Graphs with Additional Information

- In real problems such as the Netflix task, additional information can be obtained (e.g., users' age, gender, hobbies, education, movies' genre, release year, actors, origin country, etc).
- This additional information can be taken advantage of with graphs to encode relationships between users and movies.

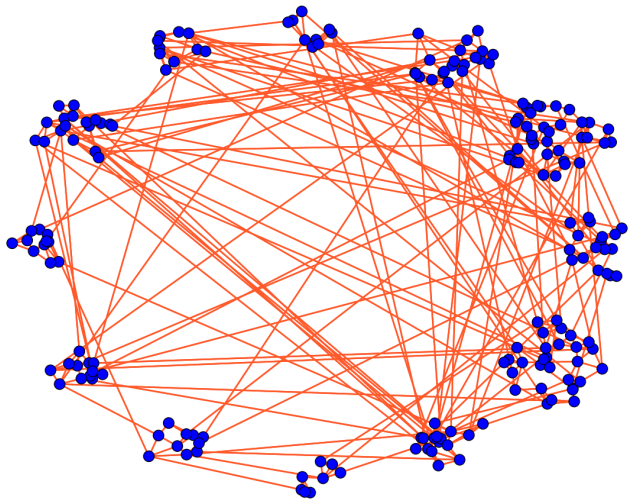


Figure: Graph of users: there will be an edge between 2 similar-tastes users

Graph-regularized Matrix Completion (GMC)

- In matrix completion, we want the reconstructed vectors x_j, x_j' in X $(= x_1, \ldots x_n)$ to be close if $e_{j,j'} \in E_c$.
- Stated differently, we want

$$\sum_{j,j'} w_{jj'}^c \left\| x_j - x_{j'} \right\|_2^2 = \mathrm{tr}\left(X L_c X^\top \right) = \|X\|_{\mathcal{D},c}^2$$

to be small, where $D_c = \mathrm{Diag}\left(\sum_{j'=1}^n w_{jj'}^c \right)$, $L_c = D_c - W_c$ is the Laplacian of the column graph G_c, and $\|\cdot\|_{\mathcal{D},c}$ is the graph Direchlet semi-norm for columns. Similarly, we can get $\|X\|_{\mathcal{D},r}^2$

- These smoothness terms can be seen as regularization terms [Benzi et al., 2016]

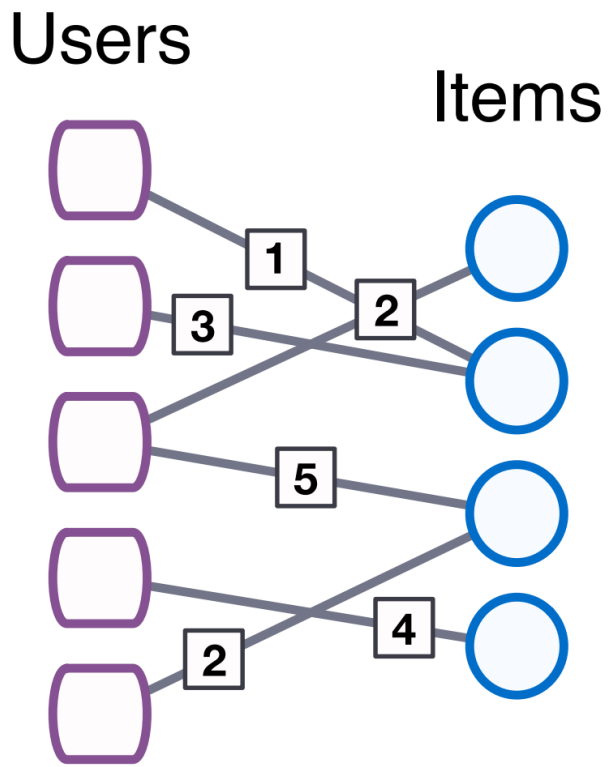
$$\min_X \gamma_n \|X\|_* + \ell\left(\mathcal{P}_\Omega(X), \mathcal{P}_\Omega(M)\right) + \frac{\gamma_r}{2}\|X\|_{\mathcal{D},r}^2 + \frac{\gamma_c}{2}\|X\|_{\mathcal{D},c}^2$$

Outline

1. What is Matrix Completion?
2. Exact Matrix Completion
3. Approximate Matrix Completion
4. Follow-up Research
   - Graph-regularized Matrix Completion
   - Link Prediction on Graphs with Deep Learning
   - Matrix Completion under Missing Mechanisms

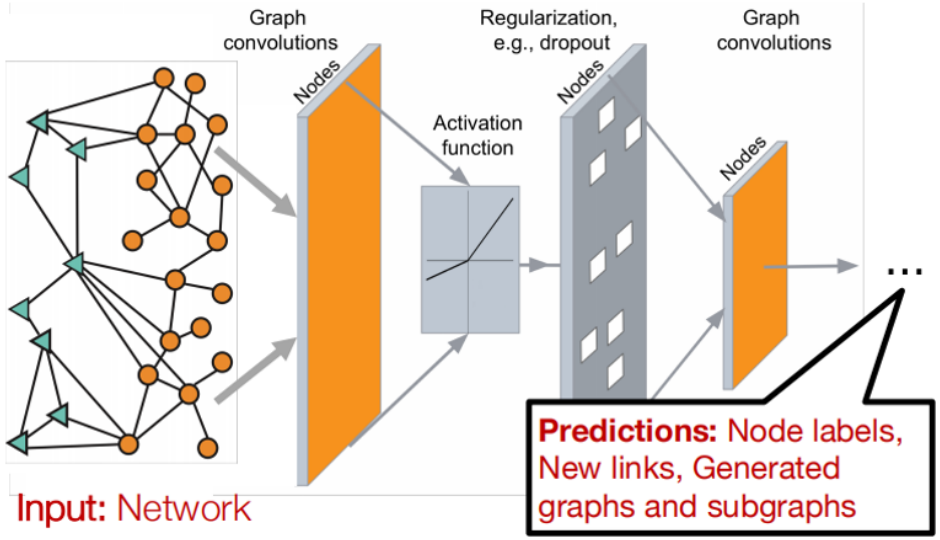
Link Prediction on Graphs

- Undirected weighted graphs can be built among users and items (in the recommendation systems).
- A bipartite graph can also be built between users and items.
- Thus, matrix completion problem can turn to be a link prediction problem.



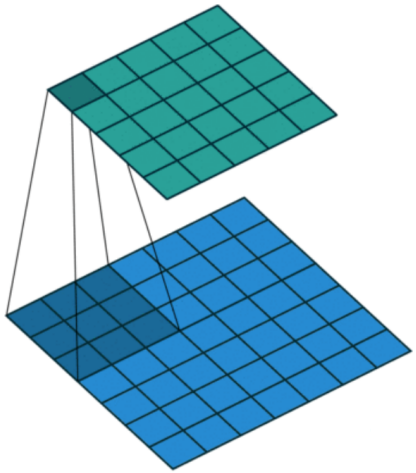
Graph Neural Network (GNN): Deep learning on Graphs

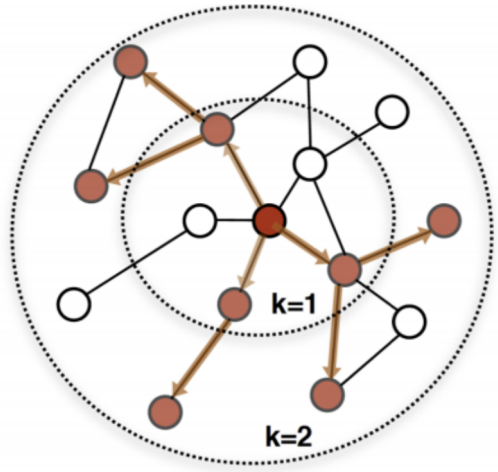
- Graph Neural Networks(GNN) [Scarselli et al., 2009] are designed to solve graphical problem with deep learning.
- Similar structure as deep feedforward neural network: convolution, activation, dropout...
- Convolution and output functions need to be designed specifically according to the data and the task.



Graph Convolutional Networks

- Define Graph Convolutional Networks (GCN) like CNN [Kipf and Welling, 2016].
- In CNN, convolutional operations are sliding on a rectangular tensor.
- In GCN, convolutional are sliding on the vertexes, where kernel covering all the neighbor vertexes.





- The above picture depicts when the kernel size is 1 and 2.

Graph Convolutional Networks

- Graph convolution: nodes aggregate information from their neighbors.
- Every vertex can be encoded into a representation vector with multi-layer graph convolution.
- In the Netflix task, one get the users and movies representation U and V respectively.

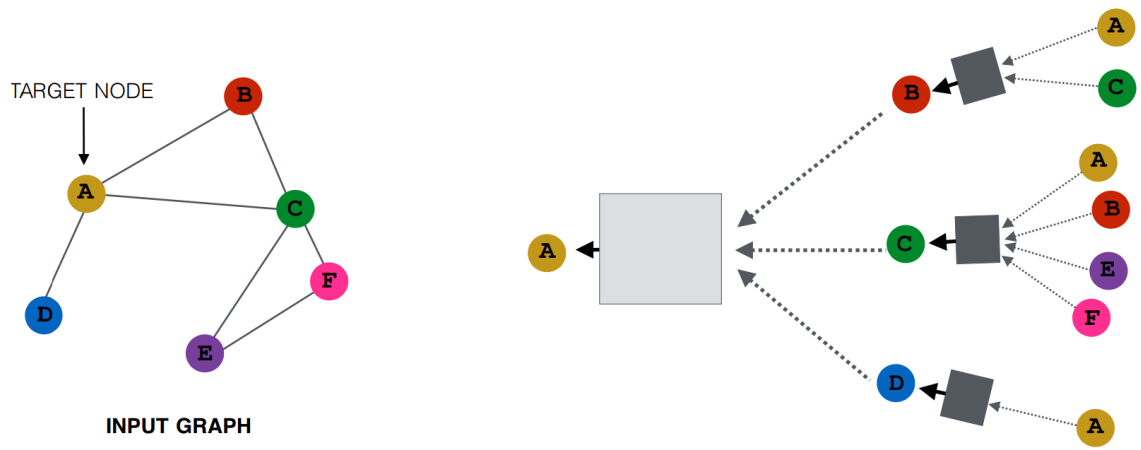


Figure: 2-order convolution: all 2 unit distance nodes information flows into the central node

What's the Output?

- Now we get users and movies representations U and V.
- How to predict the weighted edges between them?
- A commonly used activation function in link prediction is a little like softmax,

$$p\left(\check{M}_{ij} = r\right) = \frac{e^{u_i^T Q_r v_j}}{\sum_{s \in R} e^{u_i^T Q_s v_j}}$$

- In the Netflix task, r is 1-5.

Graph Convolutional Matrix Completion

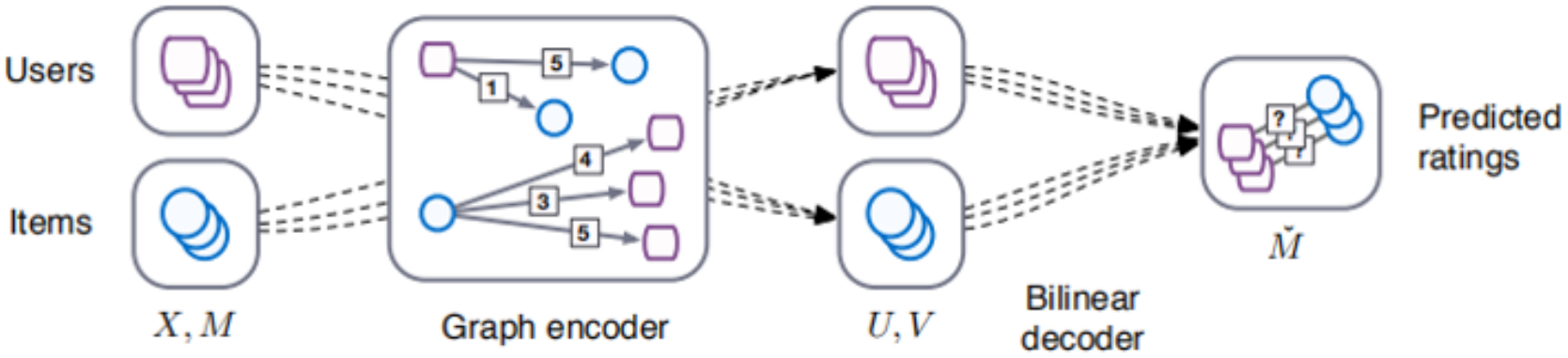


Figure: structure of graph neural network in link prediction

- Matrix Factorization is limited by its expressive ability, failing to mine deep information in the data.
- Graph Neural Network achieves the state-of-the-art results [Berg et al., 2017].

Outline

1. What is Matrix Completion?
2. Exact Matrix Completion
3. Approximate Matrix Completion
4. Follow-up Research
   - Graph-regularized Matrix Completion
   - Link Prediction on Graphs with Deep Learning
   - Matrix Completion under Missing Mechanisms

Matrix Completion under Missing Mechanisms

- Most existing studies adopt a uniform observation mechanism, where each entry has the same marginal probability of being observed.
- This leads to significant simplifications, and enables the domain to move forward rapidly with various theoretical breakthroughs in the last decade.
- Uniform mechanism is often unrealistic.
- Several studies have been devoted to relaxing such an restrictive assumption by adopting other missing structures.

Non-uniform Missing Mechanisms

- Why non-uniform Missing Mechanism is necessary?
- In movie recommender system,
  - users tend to rate movies that they prefer or dislike most, while often remain silent to other movies.
- In the online merchandising,
  - users may purchase certain items regularly without often rating them, but evaluate products that they rarely buy with a higher chance.
- Similar to the latent factor model, it is reasonable to believe that the missingness is also governed by a small and possibly different set of hidden factors.

How to Model the Missing Mechanism?

- In matrix completion setting, for the $(i; j)$-th entry, define the sampling indicator $w_{ij} = 1$ if it's observed, and 0 otherwise.
- As for the sampling mechanism, we adopt a Bernoulli model where w_{ij} are independent Bernoulli random variables with observation probabilities θ_{ij}, collectively denoted by a matrix Θ,

$$\Theta := (\theta_{ij})_{i,j=1}^{n_1,n_2} \in (0,1)^{n_1 \times n_2}$$

- Similar to the latent factor model, it is also believed Θ is low-rank.

Low-rank Modeling of Θ

- There have been lots of works on how to model the missing mechanism Θ.
  - [Srebro and Salakhutdinov, 2010] utilize an independent row and column sampling mechanism, leading to a rank-1 structure for Θ.
  - [Cai et al., 2016] consider a block structure for Θ and hence M can be regarded as a special case of the low-rank modeling.
  - [Mao et al., 2018] consider Θ is dependent on some hidden factors, which reflects situations when obvious covariates are unknown or not available.
  - Still a hot area currently...

Weighted Loss Function

- Once we get the probability matrix Θ, we induce the weighted nuclear norm minimization loss,

$$min_X ||\Theta \circ (X - M)||_2 + \lambda ||X||_*$$

- or weighted low rank matrix factorization loss,

$$min_X ||\Theta \circ (M - WH)||_2 + \lambda (||W||_2 + ||H||_2)$$

- the meaning is simple: attention will be focused on the entries with high probability!!!
- Optimization is similar to the original problem.

References I

Benzi, K., Kalofolias, V., Bresson, X., and Vandergheynst, P. (2016).

Song recommendation with non-negative matrix factorization and graph total variation.
In *IEEE International Conference on Acoustics*.

Berg, R. V. D., Kipf, T. N., and Welling, M. (2017).

Graph convolutional matrix completion.

Boumal, N., Voroninski, V., and Bandeira, A. S. (2016).

The non-convex burer–monteiro approach works on smooth semidefinite programs.
In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, pages 2765–2773, USA. Curran Associates Inc.

Cai, J. F., Candes, E. J., and Shen, Z. (2010).

A singular value thresholding algorithm for matrix completion.
SIAM Journal on Optimization, 20(4):1956–1982.

Cai, T., Cai, T. T., and Zhang, A. (2016).

Structured matrix completion with applications to genomic data integration.
Journal of the American Statistical Association, 111(514):621–633.

Candes, E. J. and Recht, B. (2009).

Exact matrix completion via convex optimization.
Foundations of Computational mathematics, 9(6):717.

Gemulla, R., Nijkamp, E., Haas, P. J., and Sismanis, Y. (2011).

Large-scale matrix factorization with distributed stochastic gradient descent.
In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '11, pages 69–77, New York, NY, USA. ACM.

References II

Hastie, T., Mazumder, R., Lee, J. D., and Zadeh, R. (2015).

Matrix completion and low-rank svd via fast alternating least squares.
The Journal of Machine Learning Research, 16(1):3367–3402.

Kipf, T. N. and Welling, M. (2016).

Semi-supervised classification with graph convolutional networks.

Lee, D. and Seung, H. (1999).

Learning the parts of objects by non-negative matrix factorization.
Nature, 401(6755):788–791.

Liu, C., Yang, H., Fan, J., He, L. W., and Wang, Y. M. (2016).

Distributed nonnegative matrix factorization for web-scale dyadic data analysis on mapreduce.
Inproceedings.

Mao, X., Wong, R. K., and Chen, S. X. (2018).

Matrix completion under low-rank missing mechanism.
arXiv preprint arXiv:1812.07813.

Mardani, M., Mateos, G., and Giannakis, G. B. (2012).

Distributed nuclear norm minimization for matrix completion.
In *IEEE International Workshop on Signal Processing Advances in Wireless Communications*.

Mazumder, R., Hastie, T. J., and Tibshirani, R. (2010).

Spectral regularization algorithms for learning large incomplete matrices.
Journal of machine learning research : JMLR, 11:2287–2322.

Recht, B. (2009).

A Simpler Approach to Matrix Completion.

References III

Saddiki, H., McAuliffe, J., and Flaherty, P. (2014).

GLAD: a mixed-membership model for heterogeneous tumor subtype classification.
Bioinformatics, 31(2):225–232.

Salakhutdinov, R. (2008).

Probabilistic matrix factorization.
Advances in Neural Information Processing Systems, 20:1257–1264.

Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., and Monfardini, G. (2009).

The graph neural network model.
IEEE Transactions on Neural Networks, 20(1):61–80.

Srebro, N., Rennie, J. D. M., and Jaakkola, T. S. (2004).

Maximum-margin matrix factorization.
In *NIPS*.

Srebro, N. and Salakhutdinov, R. R. (2010).

Collaborative filtering in a non-uniform world: Learning with the weighted trace norm.
In *Advances in Neural Information Processing Systems*, pages 2056–2064.

Sun, D. L. and Fevotte, C. (2014).

Alternating direction method of multipliers for non-negative matrix factorization with the beta-divergence.
In *IEEE International Conference on Acoustics*.

Tipping, M. and Bishop, C. (1997).

Probabilistic principal component analysis.
Workingpaper, Aston University.

References IV

Xu, Y., Yin, W., Wen, Z., and Zhang, Y. (2012).

An alternating direction algorithm for matrix completion with nonnegative factors.
Frontiers of Mathematics in China, 7(2):365–384.

Yu, Z. Q., Shi, X. J., Yan, L., and Li, W. J. (2014).

Distributed stochastic admm for matrix factorization.
In *Acm International Conference on Conference on Information & Knowledge Management*.

Yun, H., Yu, H. F., Hsieh, C. J., Vishwanathan, S. V. N., and Dhillon, I. (2013).

Nomad: Non-locking, stochastic multi-machine algorithm for asynchronous and decentralized matrix completion.
Proceedings of the Vldb Endowment, 7(11):975–986.

Zhou, Y., Wilkinson, D. M., Schreiber, R., and Rong, P. (2008).

Large-scale parallel collaborative filtering for the netflix prize.
In *Proc Intl Conf Algorithmic Aspects in Information & Management, Lncs*.