Why Biased Matrix Factorization Works Well?

2018. 8. 29.

Joonyoung Yi
joonyoung.yi@kaist.ac.kr
Machine Learning & Intelligence Laboratory
School of Computing
Korea Advanced Institute of Science and Technology

KAIST

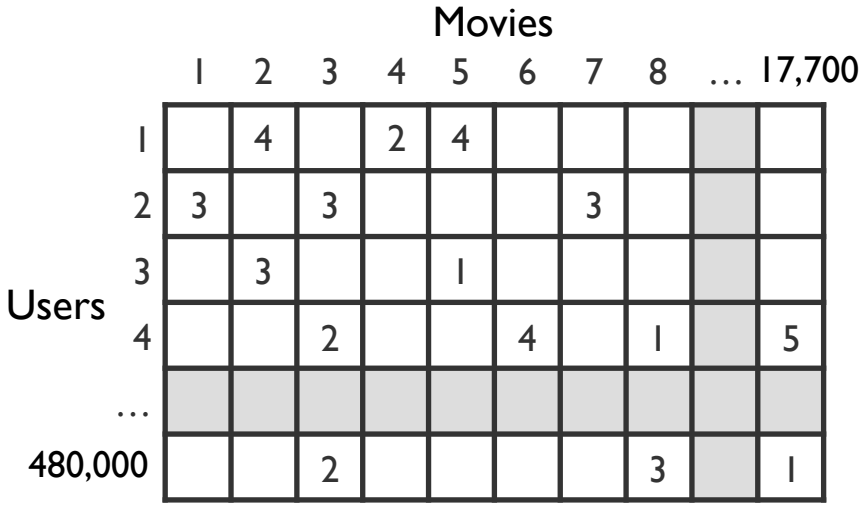
MLILAB
Machine Learning & Intelligence

Table of Contents

Table of Contents

NETFLIX PRIZE

- In October 2016, Netflix open a $1M Prize called Netflix Prize.
- To improve recommendation accuracy by 10%.
- They splitted the data for prize.
  - Training set: 100M
  - Test set: 1.4M
- Training set contains 1.2 % of entries.
  - 480,000 users and 17,700 movies.
- From a matrix viewpoint, training set contains only 1.2% entries of the matrix.



Data Description of Netflix Prize

- Each set(training set, test set) is a set of (i, j, M_{ij})s.
  - i: user index
  - j: item(or movie) index
  - M_{ij}: true rating of user u to item i.
  - Ω : the set of (i, j) corresponding to the training set(=The set of known entries).
  - Ω': the set of (i, j) corresponding to the test set(=The set of entries to predict).
  - m is the number of users.
  - n is the number of items.

Data Description of Netflix Prize

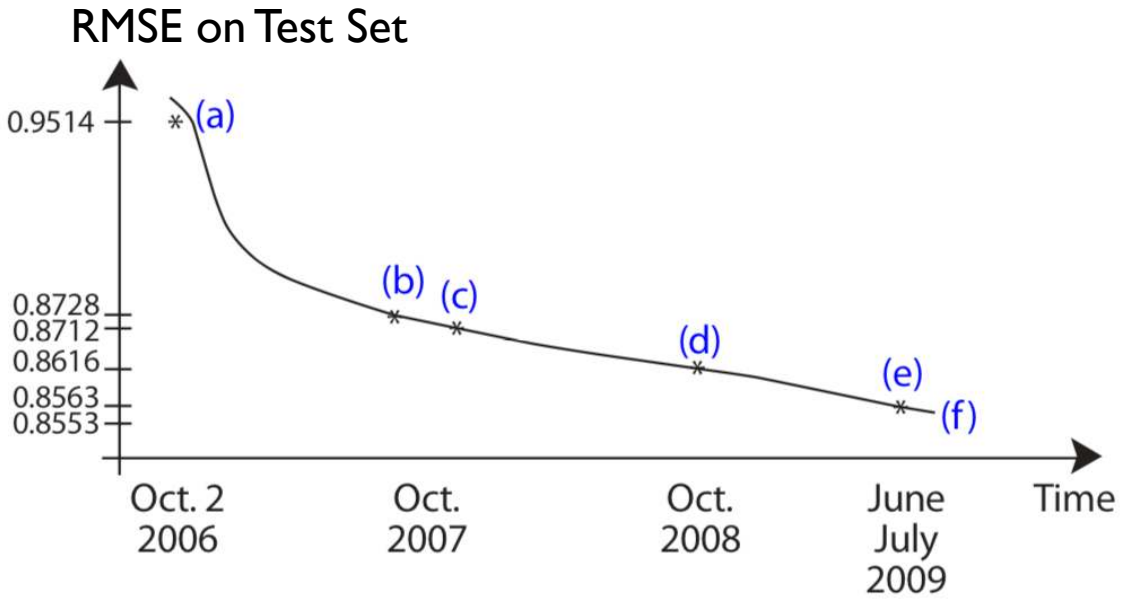
- RMSE(Root Mean Square Error) was used for measuring accuracy.

$$\text{train RMSE} = \sqrt{\sum_{\Omega} \frac{(M_{ij} - R_{ij})^2}{|\Omega|}} = \frac{\|P_{\Omega}(M - R)\|_F}{\sqrt{\Omega}}$$

$$\text{test RMSE} = \sqrt{\sum_{\Omega'} \frac{(M_{ij} - R_{ij})^2}{|\Omega'|}} = \frac{\|P_{\Omega'}(M - R)\|_F}{\sqrt{\Omega'}}$$

  - R_{ij}: predicted rating of user *i* to item *j*.
  - $|\Omega'|$ is the cardinality of the Ω'.
  - Projection Operator $P_{\Omega}(M) = \begin{cases} M_{ij} & \text{if } (i,j) \in \Omega, \\ 0 & \text{otherwise.} \end{cases}$
  - *M*, *R* are rating matrices.
  - $M, R \in \mathbb{R}^{m \times n}$

RMSE of Netflix, Prize Winner and Biased-MF



- RMSE of Cinematch (Original Netflix Recommendation Engine): 0.9514
- RMSE of Winner: 0.8553 (**10%** enhancement)
  - They ensembled a lot of algorithms.
- RMSE of Biased-MF **ONLY**: 0.8799 (**7.5%** enhancement)
  - Some materials said 6% enhancement only by Biased-MF.
- RMSE of Biased-MF(with tuned hyper-parameters): 0.844 (**11.3%** enhancement)
  - From the AutoRec Paper in WWW'15.

BIASED-MF IS FAST AND WORKS WELL

- Many Algorithms are presented after Biased-MF.
- Biased-MF is the fastest algorithm among them.
  - The convex relaxation(nuclear norm minimization) algorithm is slower than Biased-MF.
- Biased-MF works quite well even these days.
  - And, the some of algorithms with good performance are based on Biased-MF.
  - SMA and MRMA are ensemble algorithms based on Biased-MF.
  - LLoRMA and ABCF are modification of Biased-MF.

	Movielens 1M	Movielens 10M	Netflix(100M)
Biased MF(IEEE'09)	**0.845**	**0.803**	**0.844**
LLoRMA(ICML'13)	0.8333	0.7815	0.8337
AutoRec(WWW'15)	0.831	0.782	0.823
CF-NADE(ICML '16)	0.829	0.771	0.803
SMA(ICML'16)	**-**	**0.7682**	**0.8036**
ABCF(Neurocomputing'18)	0.836	0.766	0.795
MRMA(NIPS'17)	**-**	**0.7634**	**0.7973**

Table of Contents

FUNK SVD (SINGULAR VECTOR DECOMPOSITION)

- Before introducing Biased-MF, Let's see Funk SVD(suggested by Funk).
- Goal of Netflix Prize: Minimize RMSE

$$\arg\min_{R} \sqrt{\sum_{(i,j)\in\Omega} \frac{(M_{ij} - R_{ij})^2}{|\Omega|}} = \arg\min_{R} \sum_{(i,j)\in\Omega} (M_{ij} - R_{ij})^2$$

  - There is no way to solve this optimization form without any assumption.
  - In rating recommendation, it is natural to assume the matrix $\hat{r}_{ui}$ is low-rank.

- New Optimization Form with Low-rank Assumption:

$$\arg\min_{\hat{U},\hat{V}} \sum_{(i,j)\in\Omega} (M_{ij} - \hat{U}_i \hat{V}_j^{\dagger})^2$$

  - *p* and *q* are low-rank matrices: $\max(\mathrm{rank}(\hat{U}), \mathrm{rank}(\hat{V})) \ll \dim(M)$
  - To avoid overfitting:

$$\arg\min_{\hat{U},\hat{V}} \sum_{(i,j)\in\Omega} (M_{ij} - \hat{U}_i \hat{V}_j^{\dagger})^2 + \lambda_U \|\hat{U}\|_F^2 + \lambda_V \|\hat{V}\|_F^2$$

Solution of the Funk SVD

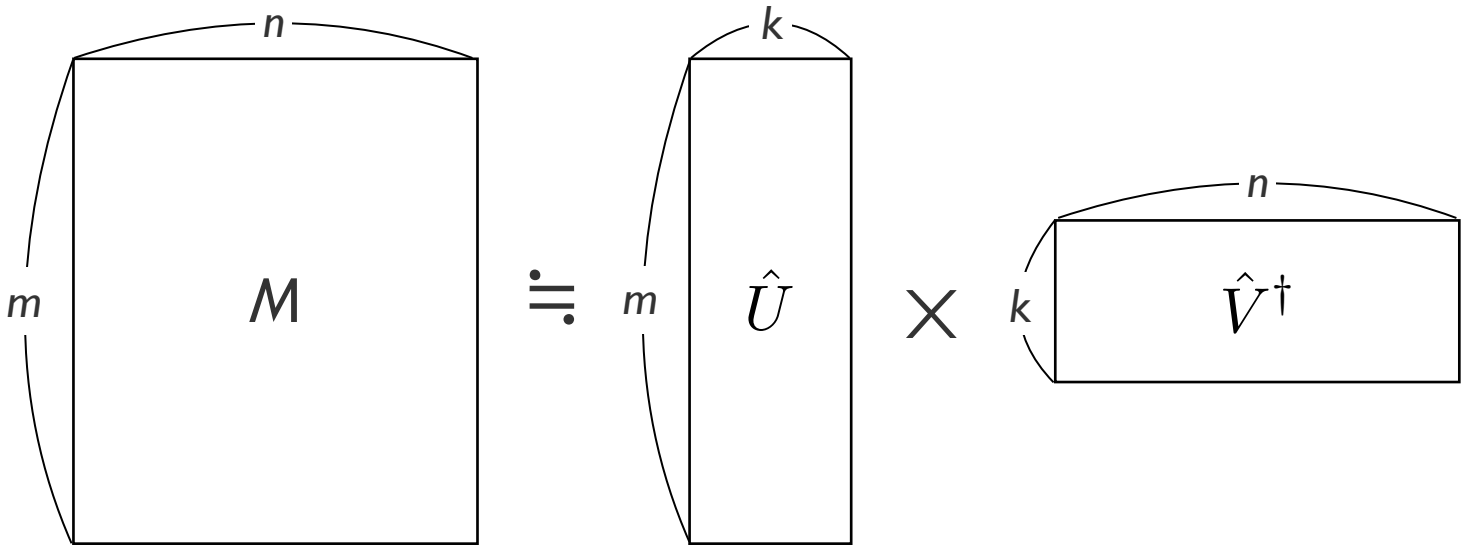
- Optimization form of the Funk SVD:

$$\arg\min_{\hat{U},\hat{V}} \sum_{(i,j)\in\Omega} (M_{ij} - \hat{U}_i \hat{V}_j^{\dagger})^2 + \lambda_U \|\hat{U}\|_F^2 + \lambda_V \|\hat{V}\|_F^2$$

- If we know all entries (= Ω contains all (*u, i*) pairs), we can solve by SVD (Singular vector decomposition).
  - However, we can't know the all entries.

- If we don't know all entries, this is NP-hard problem.
  - Then, how to solve this optimization problem?
  - One of approaches is convex relaxation.
  - But, the Funk-SVD use another algorithm.

Solve by Alternating Minimization



1: Input: observed set Ω, values $P_\Omega(M)$

2: Initialize $\hat{V}^0$ randomly.

3: `for` $t = 1, \cdots, T$:

4: $\hat{U}^t \leftarrow \arg\min_{U \in \mathbb{R}^{m \times k}} \|P_\Omega(M - U(\hat{V}^{t-1})^\dagger)\|_F^2 + \lambda_U \|U\|_F^2$

5: $\hat{V}^t \leftarrow \arg\min_{V \in \mathbb{R}^{k \times n}} \|P_\Omega(M - \hat{U}^t V^\dagger)\|_F^2 + \lambda_V \|V\|_F^2$

These can be solved by SVD! (pseudo inverse)

6: Return $R = \hat{U}^t(\hat{V}^t)^\dagger$

What is Biased Matrix Factorization?

- The Biased Matrix Factorization is based on Funk SVD.
  - Biased MF = Funk SVD + Bias Terms
- Some users tend to give higher ratings, and some users tend to give lower ratings.
  - The tendencies of movies are similar to that of users.
  - Some movies tend to get higher ratings, and some movies tend to get lower ratings.

- Biased Matrix Factorization wants to handle this phenomenon.
  - To introduce bias terms in the optimization form.

Optimization Form of Biased-MF

- Therefore, Biased-MF introduce biased terms related to user and item respectively.

$$\min_{\hat{U},\hat{V},b^{user},b^{item}} \sum_{(i,j)\in\Omega} (M_{ij} - \mu - b_i^{user} - b_j^{item} - \hat{U}_i\hat{V}_j^{\dagger})^2 + \lambda_U \|\hat{U}\|_F^2 + \lambda_V \|\hat{V}\|_F^2 + \lambda_{b^{user}} \|b^{user}\|_2^2 + \lambda_{b^{item}} \|b^{item}\|_2^2$$

  - μ is the average rating of training set(constant).
  - b^{user} is the bias vector related to user.
  - b^{item} is the bias vector related to item.

- This optimization form can be solved by alternating minimization.
  - Similar to the solution of Funk SVD.

- The Biased Matrix Factorization is also called SVD++.

The Role of Bias Terms in Biased-MF

- The bias terms serve the role of normalization.
  - The bias terms serve to make the rating matrix *r* be zero mean.

- Meanwhile, the Biased-MF without bias terms(=Funk SVD) also works quite well.
  - The RMSE performance on Netflix Dataset is similar to original recommendation engine of Netflix.

- Therefore, knowing why Biased-MF works well is equivalent to knowing why Funk SVD works well.
- Now, we will investigate why Funk SVD works well on low-rank matrices.

Table of Contents

Optimization Form of Funk SVD

- Recall: Optimization form of Funk SVD

$$\arg\min_{\hat{U},\hat{V}} \sum_{(i,j)\in\Omega} (M_{ij} - \hat{U}_i \hat{V}_j^{\dagger})^2 + \lambda_U \|\hat{U}\|_F^2 + \lambda_V \|\hat{V}\|_F^2$$

- Probabilistic Matrix Factorization (PMF, NIPS'08) interprets Funk SVD in the view point of posterior.
  - Funk SVD can be interpreted as an algorithm that finds the most probable low-rank matrices *p* and *q* when a star is observed.

Probabilistic Matrix Factorization

- Optimization form of PMF:

$$\arg\max_{\hat{U},\hat{V}} \Pr[\hat{U},\hat{V}|P_\Omega(M),\Omega,\sigma^2,\sigma_U^2,\sigma_V^2]$$

  - To find $\hat{U}$ and $\hat{V}$ via MAP(Maximum A Posteriori).

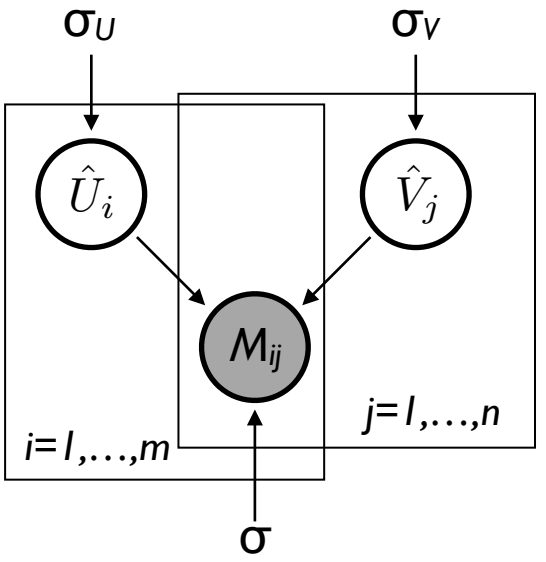
- A low-rank assumption ($M_{ij} = \hat{U}_i\hat{V}_j^\dagger$) and Gaussian noise assumptions.

$$\Pr[M|\hat{U},\hat{V},\sigma^2] = \prod_{i=1}^{m}\prod_{j=1}^{n}[\mathcal{N}[M_{ij}|\hat{U}_i\hat{V}_j^\dagger,\sigma^2]]^{\mathcal{I}_{ij}^\Omega},$$

$$\Pr[\hat{U}|\sigma_U^2] = \prod_{i=1}^{m}\mathcal{N}[\hat{U}_i|0,\sigma_U^2\mathbb{I}],$$

$$\Pr[\hat{V}|\sigma_V^2] = \prod_{j=1}^{n}\mathcal{N}[\hat{V}_j|0,\sigma_V^2\mathbb{I}],$$

$$\mathcal{I}_{ij}^\Omega = \begin{cases} 1 & (i,j)\in\Omega \\ 0 & \text{otherwise.} \end{cases}$$



The Log Posteriori of PMF

- The log posteriori of PMF is as follows:

$$\begin{aligned}
&\ln \Pr[\hat{U}, \hat{V} | P_\Omega(M), \Omega, \sigma^2, \sigma_U^2, \sigma_V^2] \\
&= -\frac{1}{2\sigma^2} \sum_{i=1}^{m} \sum_{j=1}^{n} \mathcal{I}_{ij}^{\Omega} (M_{ij} - \hat{U}_i \hat{V}_j^{\dagger})^2 - \frac{1}{2\sigma_U^2} \sum_{i=1}^{m} \hat{U}_i^{\dagger} \hat{U}_i - \frac{1}{2\sigma_V^2} \sum_{j=1}^{n} \hat{V}_j^{\dagger} \hat{V}_j \\
&\quad - \frac{1}{2} ((\sum_{i=1}^{m} \sum_{j=1}^{n} \mathcal{I}_{ij}^{\Omega}) \ln \sigma^2 + mk \ln \sigma_U^2 + nk \ln \sigma_V^2) + C
\end{aligned}$$

- Hence, the following equation holds:

$$\begin{aligned}
&\arg \max_{\hat{U}, \hat{V}} \Pr[\hat{U}, \hat{V} | P_\Omega(M), \Omega, \sigma^2, \sigma_U^2, \sigma_V^2] \\
&= \arg \min_{\hat{U}, \hat{V}} \sum_{(i,j) \in \Omega} (M_{ij} - \hat{U}_i \hat{V}_j^{\dagger})^2 + \lambda_U \|\hat{U}\|_F^2 + \lambda_V \|\hat{V}\|_F^2
\end{aligned}$$

Table of Contents

Why We Use Alternating Minimization?

- We can just use Gradient Descent with random initialization without alternating minimization.
    - Why we should use alternating minimization?

- First of all, Gradient Descent can oscillate without alternating minimization.

- The MRMA paper(NIPS'17) said that it is possible to overfit without alternating minimization.
- Alternating Minimization is not the only way to avoid overfitting.
- However, Besag(1986) said that it is a good way to avoid overfitting.
- Glendinning(1989) said that alternating minimization is robust to initial point empirically.

- However, the initial point is nonetheless important.
- Because it can fall into the local minima.

Table of Contents

Local Minima Problem of AM

- In otherwise, Jain et al(2013) showed that AM has no local minima problem.
  - Noiseless case **ONLY**. Exact Completion Setting.
  - With the slightly modified algorithm.
  - With some assumptions.

- Let's look at the modified algorithm first.

The Modified Algorithm

Algorithm 2 AltMinComplete: Alternating minimization for matrix completion

1: Input: observed set Ω, values $P_\Omega(M)$
2: Partition Ω into $2T+1$ subsets $\Omega_0, \cdots, \Omega_{2T}$ with each element of Ω belonging to one of the Ω_t with equal probability (sampling with replacement)
3: $\widehat{U}^0 = SVD(\frac{1}{p}P_{\Omega_0}(M), k)$ i.e., top-k left singular vectors of $\frac{1}{p}P_{\Omega_0}(M)$
4: Clipping step : Set all elements of $\widehat{U}^0$ that have magnitude greater than $\frac{2\mu\sqrt{k}}{\sqrt{n}}$ to zero and orthonormalize the columns of $\widehat{U}^0$
5: **for** $t = 0, \cdots, T-1$ **do**
6: $\quad \widehat{V}^{t+1} \leftarrow \text{argmin}_{V \in \mathbb{R}^{n \times k}} \ \|P_{\Omega_{t+1}}(\widehat{U}^t V^\dagger - M)\|_F^2$
7: $\quad \widehat{U}^{t+1} \leftarrow \text{argmin}_{U \in \mathbb{R}^{m \times k}} \ \|P_{\Omega_{T+t+1}}(U \left(\widehat{V}^{t+1}\right)^\dagger - M)\|_F^2$
8: **end for**
9: Return $X = \widehat{U}^T(\widehat{V}^T)^\dagger$

The Modified Algorithm

2: Partition Ω into $2T+1$ subsets $\Omega_0, \cdots, \Omega_{2T}$ with each element of Ω belonging to one of the Ω_t with equal probability (sampling with replacement)

Mini-batch

The Modified Algorithm

5: **for** $t = 0, \cdots, T-1$ **do**
6: $\quad \widehat{V}^{t+1} \leftarrow \text{argmin}_{V \in \mathbb{R}^{n \times k}} \ \|P_{\Omega_{t+1}}(\widehat{U}^t V^\dagger - M)\|_F^2$
7: $\quad \widehat{U}^{t+1} \leftarrow \text{argmin}_{U \in \mathbb{R}^{m \times k}} \ \|P_{\Omega_{T+t+1}}(U \left(\widehat{V}^{t+1}\right)^\dagger - M)\|_F^2$
8: **end for**

Alternating Minimization similar to Funk SVD

The Modified Algorithm

3: $\hat{U}^0 = SVD(\frac{1}{p}P_{\Omega_0}(M), k)$ i.e., top-k left singular vectors of $\frac{1}{p}P_{\Omega_0}(M)$

4: Clipping step : Set all elements of $\widehat{U}^0$ that have magnitude greater than $\frac{2\mu\sqrt{k}}{\sqrt{n}}$ to zero and orthonormalize the columns of $\widehat{U}^0$

When performing initialization using SVD + Clipping,
the local optimum found by the SVD method
is close enough to the global optimum!

Motivation of The Incoherence Assumption

- Consider the rank-1 matrix M:

$$M = e_1 e_n^* = \begin{bmatrix} 0 & 0 & \cdots & 0 & 1 \\ 0 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & 0 \end{bmatrix}$$

  - Let $|\Omega|$ be the number of observed entries of *M*.
  - Then, we can see only *0* with probability *1* - $|\Omega|$ / *(mn)*.
  - If sample set doesn't contain *1*, we can not complete matrix exactly.

- Therefore, it is impossible to recover all low-rank matrices.

The Incoherence Assumption

- More generally, it is hard to recover if the singular vectors of the matrix M are similar to standard basis.
  - Because, information is highly concentrated on specific region.
  - Hence, the singular vectors need to be sufficiently spread.

- This is why the paper introduce the incoherence assumption.

Definition 2.4. *[3] A matrix $M \in \mathbb{R}^{m \times n}$ is incoherent with parameter μ if:*

$$\left\|u^{(i)}\right\|_2 \leq \frac{\mu\sqrt{k}}{\sqrt{m}} \; \forall \, i \in [m], \quad \left\|v^{(j)}\right\|_2 \leq \frac{\mu\sqrt{k}}{\sqrt{n}} \; \forall \, j \in [n], \tag{3}$$

where $M = U\Sigma V^T$ is the SVD of M and $u^{(i)}$, $v^{(j)}$ denote the i^{th} row of U and the j^{th} row of V respectively.

- The EMCCO paper said that the random low-rank (orthogonal) matrices satisfy the incoherent assumption.

Main Results

Theorem 2.5. *Let $M = U^*\Sigma^*V^{*\dagger} \in \mathbb{R}^{m\times n}$ $(n \geq m)$ be a rank-k incoherent matrix, i.e., both U^* and V^* are μ-incoherent (see Definition 2.4). Also, let each entry of M be observed uniformly and independently with probability,*

$$p > C\frac{\left(\frac{\sigma_1^*}{\sigma_k^*}\right)^2 \mu^2 k^{2.5} \log n \log \frac{k\|M\|_F}{\epsilon}}{m\delta_{2k}^2},$$

where $\delta_{2k} \leq \frac{\sigma_k^*}{12k\sigma_1^*}$ and $C > 0$ is a global constant. Then w.h.p. for $T = C' \log \frac{\|M\|_F}{\epsilon}$, the outputs $\widehat{U}^T$ and V^T of Algorithm 2, with input $(\Omega, P_\Omega(M))$ (see Equation (2)) satisfy: $\left\|M - \widehat{U}^T \left(V^T\right)^\dagger\right\|_F \leq \epsilon$.

- Required entries: $|\Omega| = O((k^{4.5} \log k)\, n \log n)$
- Required steps: $O(\log (1/\varepsilon))$

- The EMCCO paper said that the optimum number of required entries is $O(n \log n)$

Comparison to Nuclear Norm Minimization

- Alternating Minimization
  - Required entries: $|\Omega| = O((k^{4.5} \log k)\, n \log n)$
  - Required steps: $O(\log(1/\varepsilon))$

- Nuclear Norm Minimization (Convex Relaxation)
  - Required entries: $|\Omega| = O((k)\, n \log n)$
  - Required steps: $O(\varepsilon^{-1/2})$

- Nuclear Norm Minimization requires smaller the number of samples.
- Alternating Minimization converges faster than Nuclear Norm Minimization.
- Theoretical bound is not tight in Nuclear Norm Minimization.

Proof Sketch

- *Base Case*: Show that $u^0 = \widehat{u}^0 / \|\widehat{u}^0\|_2$ is incoherent and have small distance to u^* (see Lemma 5.2).
- *Induction Step (distance)*: Assuming $u^t = \widehat{u}^t / \|\widehat{u}^t\|_2$ to be incoherent and that u^t has a small distance to u^*, v^{t+1} decreases distances to v^* by at least a constant factor.
- *Induction Step (incoherence)*: Show incoherence of v^{t+1}, while assuming incoherence of u^t (see Lemma 5.4)

PRELIMINARY

Definition 4.1. *[8] Given two matrices* $\widehat{U}, \widehat{W} \in \mathbb{R}^{m \times k}$, *the (principal angle) distance between the subspaces spanned by the columns of* $\widehat{U}$ *and* $\widehat{W}$ *is given by:*

$$\operatorname{dist}\left(\widehat{U}, \widehat{W}\right) \stackrel{def}{=} \left\| U_{\perp}^{\dagger} W \right\|_2 = \left\| W_{\perp}^{\dagger} U \right\|_2$$

where U *and* W *are orthonormal bases of the spaces* $Span\left(\widehat{U}\right)$ *and* $Span\left(\widehat{W}\right)$, *respectively. Similarly,* $U_{\perp}$ *and* $W_{\perp}$ *are any orthonormal bases of the perpendicular spaces* $Span\,(U)^{\perp}$ *and* $Span\,(W)^{\perp}$, *respectively.*

Base Case

Lemma 5.2. *Let M, Ω, p be as defined in Theorem 2.5. Also, let U^0 be the initial iterate obtained by step 4 of Algorithm 2. Then, w.h.p. we have*

- $\text{dist}\left(U^0, U^*\right) \leq \frac{1}{2}$ *and*
- U^0 *is incoherent with parameter* $4\mu\sqrt{k}$.

Induction Step (Distance)

Theorem 5.1. *Under the assumptions of Theorem 2.5, the* $(t+1)^{th}$ *iterates* $\widehat{U}^{t+1}$ *and* $\widehat{V}^{t+1}$ *satisfy the following property w.h.p.:*

$$\text{dist}\left(\widehat{V}^{t+1}, V^*\right) \leq \frac{1}{4}\text{dist}\left(\widehat{U}^{t}, U^*\right) \quad \textit{and}$$
$$\text{dist}\left(\widehat{U}^{t+1}, U^*\right) \leq \frac{1}{4}\text{dist}\left(\widehat{V}^{t+1}, V^*\right), \quad \forall\ 1 \leq t \leq T.$$

Induction Step (Incoherence)

Lemma 5.4. *Let M, p, Ω be as defined in Theorem 2.5. Also, let u^t be a unit vector with incoherence parameter $\mu_1 = \frac{6(1+\delta_2)\mu}{1-\delta_2}$. Then, w.p. at least $1 - \frac{1}{n^3}$, v^{t+1} is also μ_1 incoherent.*

Table of Contents

Why Random Initialization Works?

- In prior paper, initialization steps matters to guarantee global optimality.
- In practice, it is hard to determine hyper-parameters for clipping.
- However, random initialization works well practically.

- In this section, I'll show you to why random initialization works.
- It can be view as removing hyper-parameters.
  - Similar to motivation of WGAN-GP.
  - WGAN is really hard to determine hyper-parameter that performance is highly sensitive to.

PRELIMINARY: MATRIX SENSING PROBLEM

- Quite similar to the matrix completion problem.

$$\text{Find } X \in \mathbb{R}^{m \times n}, \quad s.t \;\; \mathcal{A}(X) = b, \quad rank(X) \leq k.$$

$$\min_{U \in \mathbb{R}^{m \times k}, V \in \mathbb{R}^{n \times k}} \quad \|\mathcal{A}(UV^{\dagger}) - b\|_2^2$$

  - A Low-rank version of linear regression ($y = x\beta$ problem).

- RIP condition:

Definition 2.1. *[19] A linear operator $\mathcal{A}(\cdot) : \mathbb{R}^{m \times n} \to \mathbb{R}^d$ is said to satisfy k-RIP, with δ_k RIP constant, if for all $X \in \mathbb{R}^{m \times n}$ s.t. $rank(X) \leq k$, the following holds:*

$$(1 - \delta_k) \|X\|_F^2 \leq \|\mathcal{A}(X)\|_2^2 \leq (1 + \delta_k) \|X\|_F^2 . \tag{1}$$

- Actually, the Jain et al's work (2013) proved in low-rank matrix sensing problem first.
- And then, they extended their work to low-rank matrix completion problem.

No Spurious Local Minima

- Rong et al (NIPS 2016), Matrix Completion has No Spurious Local Minimum
  - Matrix Completion problem with Semi-definite matrix assumption.
  - Showed why random initialization works with proper regularizer.

- Srinadh et al (NIPS 2016), Global Optimality of Local Search for Low Rank Matrix Recovery
  - Matrix Sensing Problem.
  - Showed all local minima are very close to a global optimum with noisy measurements.
  - With a curvature bound (RIP condition),
    a polynomial time global convergence is guaranteed by SGD from **random initialization**.

- Rong et al (ArXiv 2017), No Spurious Local Minima in Non-convex Low Rank Problems: A Unified Geometric Analysis
  - Extended Srinadh et al's works from matrix sensing to matrix completion and robust PCA.
  - Showed **no high-order saddle point exists** if being with proper regularizer.

Any Question?

REFERENCES

[1] Koren, Yehuda, Robert Bell, and Chris Volinsky. "Matrix factorization techniques for recommender systems." Computer 8 (2009): 30-37.

[2] Mnih, Andriy, and Ruslan R. Salakhutdinov. "Probabilistic matrix factorization." Advances in neural information processing systems. 2008.

[3] Jain, Prateek, Praneeth Netrapalli, and Sujay Sanghavi. "Low-rank matrix completion using alternating minimization." Proceedings of the forty-fifth annual ACM symposium on Theory of computing. ACM, 2013.

[4] Ge, Rong, Jason D. Lee, and Tengyu Ma. "Matrix completion has no spurious local minimum." Advances in Neural Information Processing Systems. 2016.

[5] Bhojanapalli, Srinadh, Behnam Neyshabur, and Nati Srebro. "Global optimality of local search for low rank matrix recovery." Advances in Neural Information Processing Systems. 2016.

[6] Ge, Rong, Chi Jin, and Yi Zheng. "No spurious local minima in nonconvex low rank problems: A unified geometric analysis." arXiv preprint arXiv:1704.00708 (2017)

[7] how does Netflix recommend movies?

[8] Li, Dongsheng, et al. "Mixture-Rank Matrix Approximation for Collaborative Filtering." Advances in Neural Information Processing Systems. 2017.

REFERENCES

[9] Candès, Emmanuel J., and Benjamin Recht. "Exact matrix completion via convex optimization." Foundations of Computational mathematics 9.6 (2009): 717.

[10] Glendinning, R. H. "An evaluation of the ICM algorithm for image reconstruction." Journal of Statistical Computation and Simulation 31.3 (1989): 169-185.

[11] http://sifter.org/~simon/journal/20061211.html

[12] https://www.slideshare.net/ssuser62b35f/exact-matrix-completion-via-convex-optimization-slideppt

[13] Lee, Joonseok, et al. "Local low-rank matrix approximation." *International Conference on Machine Learning*. 2013.

[14] Sedhain, Suvash, et al. "Autorec: Autoencoders meet collaborative filtering." *Proceedings of the 24th International Conference on World Wide Web*. ACM, 2015.

[15] Zheng, Yin, et al. "A neural autoregressive approach to collaborative filtering." *arXiv preprint arXiv:1605.09477* (2016).

[16] Fu, Mingsheng, et al. "Attention based collaborative filtering." *Neurocomputing* (2018).

[17] Li, Dongsheng, et al. "Low-rank matrix approximation with stability." *International Conference on Machine Learning*. 2016.