Mixture-Rank Matrix Approximation for Collaborative Filtering

2019. 06. 25.

Joonyoung Yi
joonyoung.yi@kaist.ac.kr

KAIST

CONTENTS

KAIST

MLILAB
Machine Learning & Intelligence

CONTENTS

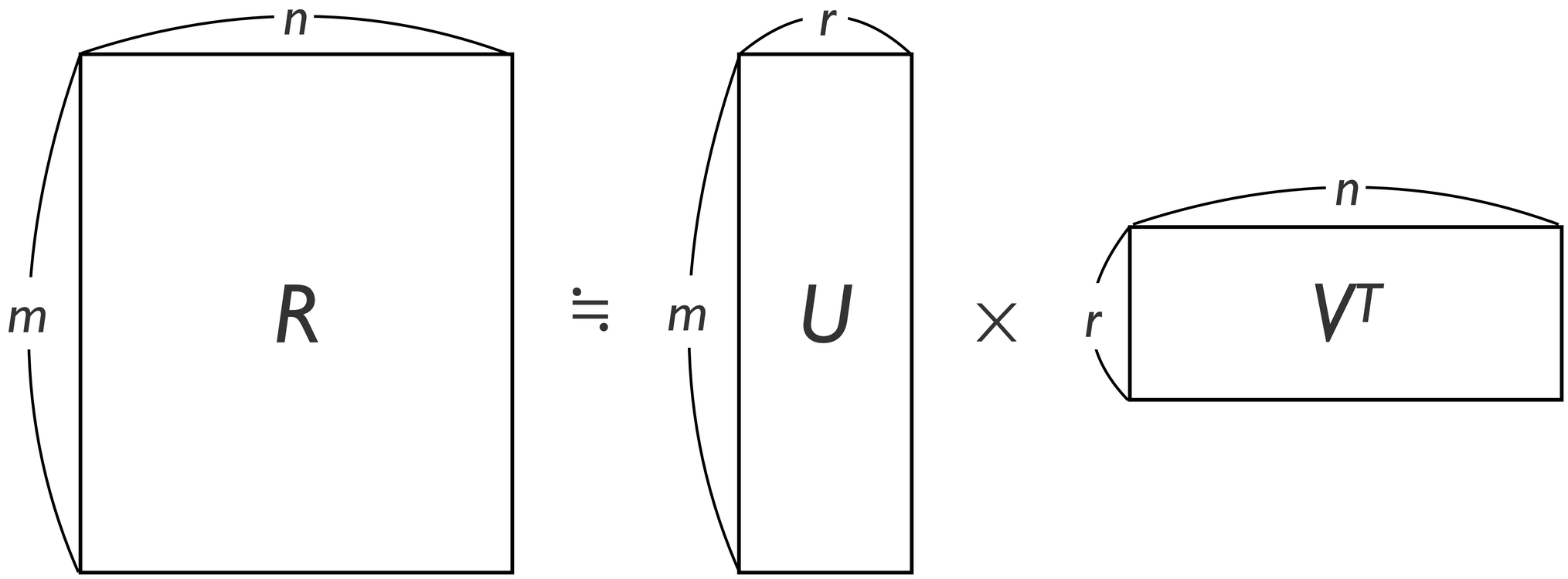
KAIST

MLILAB
Machine Learning & Intelligence

Low-rank Matrix Factorization (LRMF)

- Let $R \in \mathbb{R}^{m \times n}, U \in \mathbb{R}^{m \times r}, V \in \mathbb{R}^{n \times r}$ are matrices.
- **Low-rank Matrix Factorization** Problem:
  We know only 1% of entries in R, how to complete low-rank matrix R?
- Optimization form:

$$\hat{R} = \arg\min_{X} Loss(R, X),\ rank(X) = r.$$



Notation

- Each set(training set, test set) is a set of (i, j, R_{ij})s.
  - i: row index, j: column index
  - R_{ij}: true rating of row i to column j.
  - Ω: the set of (i, j) corresponding to the training set (=The set of known entries).
  - m is the number of rows, n is the number of columns.

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

$$\mathrm{RMSE} = \sqrt{\frac{1}{|\Omega|}\sum_{(i,j)\in\Omega}\left(\hat{R}_{i,j} - R_{i,j}\right)^2} = \sqrt{\frac{1}{|\Omega|}\sum_{i=1}^{m}\sum_{j=1}^{n}\mathbf{1}_{i,j}\left(\hat{R}_{i,j} - R_{i,j}\right)^2}$$

  - $|\Omega'|$ is the cardinality of the Ω'.
  - Indicator Operator: $\mathbf{1}_{i,j} = \begin{cases} 1 & \text{if } (i,j) \in \Omega \\ 0 & \text{otherwise} \end{cases}$
  - $\hat{R}, R \in \mathbb{R}^{m\times n}$

PROBABILISTIC MATRIX FACTORIZATION (PMF)

- Optimization form of PMF:

$$\arg\max_{U,V} \Pr\left(R|U,V,\sigma^2\right) \text{ s.t. } \operatorname{rank}(U) = \operatorname{rank}(V) = k$$

  - To find *U* and *V* via MAP (Maximum A Posteriori).

- A low-rank assumption ($R \simeq UV^T$) and Gaussian noise assumptions.

$$\Pr\left(R|U,V,\sigma^2\right) = \prod_{i=1}^{m}\prod_{j=1}^{n}\left[\mathcal{N}\left(R_{i,j}|U_iV_j^T,\sigma^2\right)\right]^{1_{i,j}}$$

- Also, place zero-mean spherical Gaussian priors on *U* and *V*,

$$\Pr\left(U|\sigma_U^2\right) = \prod_{i=1}^{m}\mathcal{N}\left(U_i|0,\sigma_U^2\mathbf{I}\right), \quad \Pr\left(V|\sigma_V^2\right) = \prod_{j=1}^{n}\mathcal{N}\left(V_j|0,\sigma_V^2\mathbf{I}\right)$$

PROBABILISTIC MATRIX FACTORIZATION (PMF)

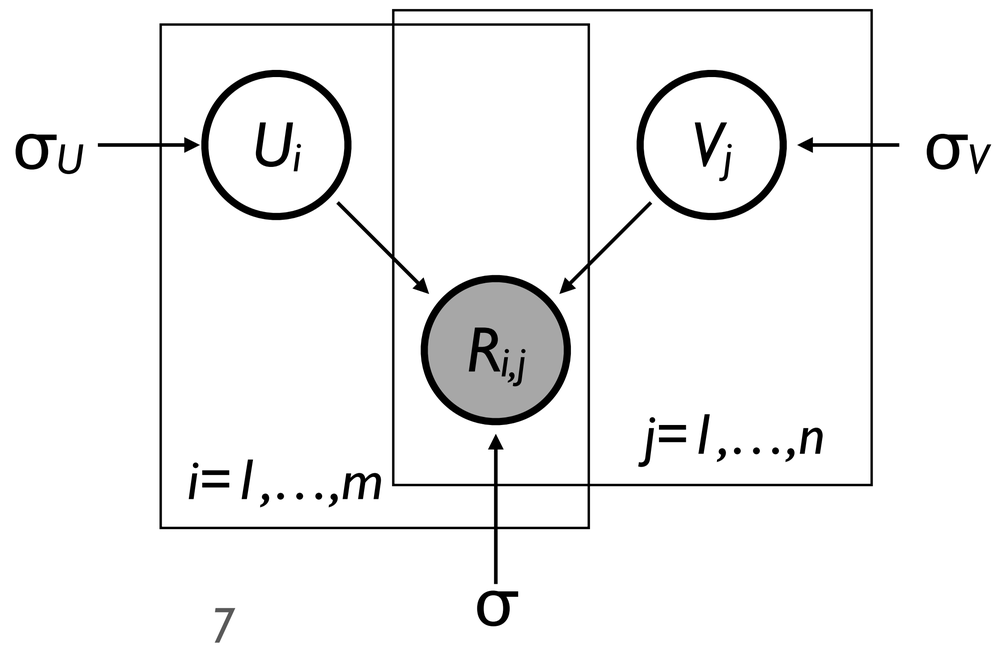
- The log of the posterior distribution over *U* and *V*,

$$\ln \Pr\left(U, V | R, \sigma^2, \sigma_V^2, \sigma_U^2\right) = -\frac{1}{2\sigma^2} \sum_{i=1}^{m} \sum_{j=1}^{n} \mathbf{1}_{i,j} \left(R_{i,j} - U_i V_j^T\right)^2 - \frac{1}{2\sigma_U^2} \|U\|_F - \frac{1}{2\sigma_V^2} \|V\|_F$$
$$- \frac{1}{2} \left(\left(\sum_{i=1}^{m} \sum_{j=1}^{n} \mathbf{1}_{i,j} \right) \ln \sigma^2 + mk \ln \sigma_U^2 + nk \ln \sigma_V^2 \right) + C$$

- We can see sigmas as hyper-parameters.

$$l = -\frac{1}{2\sigma^2} \sum_{i=1}^{m} \sum_{j=1}^{n} \mathbf{1}_{i,j} \left(R_{i,j} - U_i V_j^T\right)^2 - \frac{1}{2\sigma_U^2} \|U\|_F - \frac{1}{2\sigma_V^2} \|V\|_F$$

- Graphical Model for PMF:



Low-rank Matrix Approximation

- In vanilla Low-rank Matrix Factorization Problem,
  a solution find one pair of *U* and *V* **(single model)**.
- To enhance performance of single model,
  we can use **ensemble methods**.
- We can say some algorithm is low-rank matrix approximation
  when it finds multiple low-rank matrices to complete final matrix.

- For example,
  - [ICML'13] LLoRMA: Local Low-rank Matrix Approximation
  - [ICML'16] SMA: Low-rank Matrix Approximation with Stability
  - [AAAI'17] GLOMA: Global Information in Local Matrix Approximation
  - **[NIPS'17] MRMA: Mixture Rank Matrix Approximation**
    - The current states-of-the-arts model in Matrix Completion
      on Movielens 10M and Netflix dataset.

CONTENTS

KAIST

MLILAB
Machine Learning & Intelligence

Observation of MRMA

- Matrix Approximation methods proceed with ensembles in their own way.
- MRMA ***ensembles matrices with different ranks***.
  - Because we do not know which rank is appropriate.

- **Observation**:
  - For rows and columns with fewer samples, a lower rank is more appropriate.
  - Otherwise, a higher rank is more appropriate.

	rank = 5	rank = 50
#user ratings < 10	0.9058	0.9165
#user ratings > 50	0.8416	0.8352
#item ratings < 10	0.9338	0.9598
#item ratings > 50	0.8520	0.8418
All	0.8614	0.8583

Conditional Probability

- Probabilistic Matrix Factorization (PMF):

$$\Pr\left(R|U,V,\sigma^2\right)=\prod_{i=1}^{m}\prod_{j=1}^{n}\left[\mathcal{N}\left(R_{i,j}|U_iV_j^T,\sigma^2\right)\right]^{1_{i,j}}$$

- Mixture-rank Matrix Approximation (MRMA):

$$\Pr\left(R|U,V,\alpha,\beta,\sigma^2\right)=\prod_{i=1}^{m}\prod_{j=1}^{n}\left[\sum_{k=1}^{K}\alpha_i^k\beta_j^k\mathcal{N}\left(R_{i,j}|U_i^kV_j^{k^T},\sigma^2\right)\right]^{\mathbf{1}_{i,j}}$$

- α_i^k : the weight for rank *k* of row *i*.
- β_i^k : the weight for rank *k* of column *j*.
- U^k, V^k are the factorized matrices of the rank-*k* matrix approximation model.

Conditional Probability

- By placing a zero mean isotropic **Gaussian prior** similar to PMF,

$$\Pr\left(U^k|\sigma_U^2\right) = \prod_{i=1}^{m} \mathcal{N}\left(U_i^k|0,\sigma_U^2 I\right), \quad \Pr\left(V^k|\sigma_V^2\right) = \prod_{j=1}^{n} \mathcal{N}\left(V_j^k|0,\sigma_V^2 I\right)$$

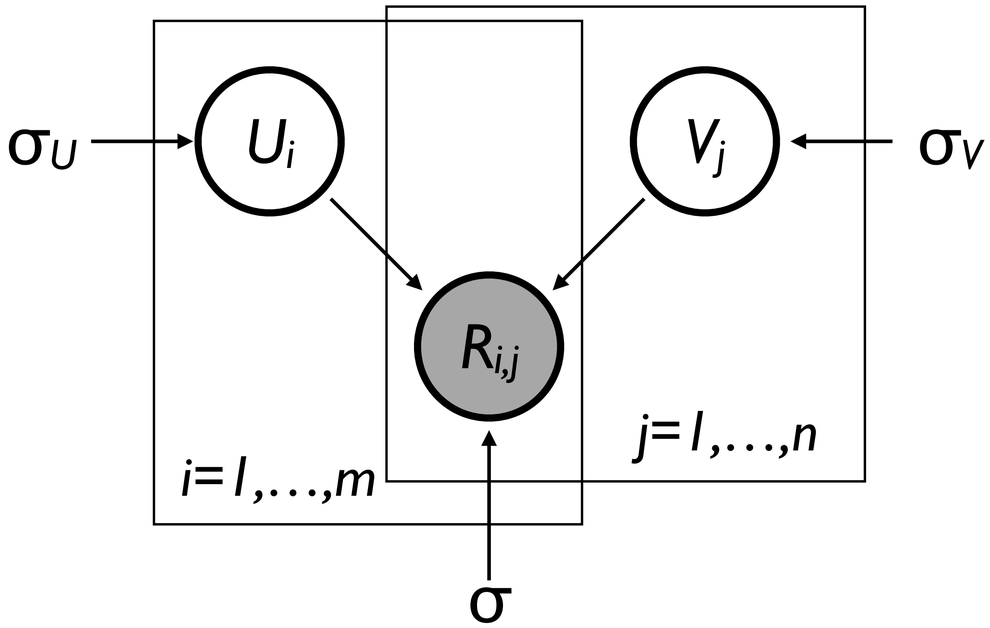
- For α^k and β^k, the paper choose a **Laplacian prior**,

$$\Pr\left(\alpha^k|\mu_\alpha,b_\alpha\right) = \prod_{i=1}^{m} \mathcal{L}\left(\alpha_i^k|\mu_\alpha,b_\alpha\right), \quad \Pr\left(\beta^k|\mu_\beta,b_\beta\right) = \prod_{j=1}^{n} \mathcal{L}\left(\beta_j^k|\mu_\beta,b_\beta\right)$$

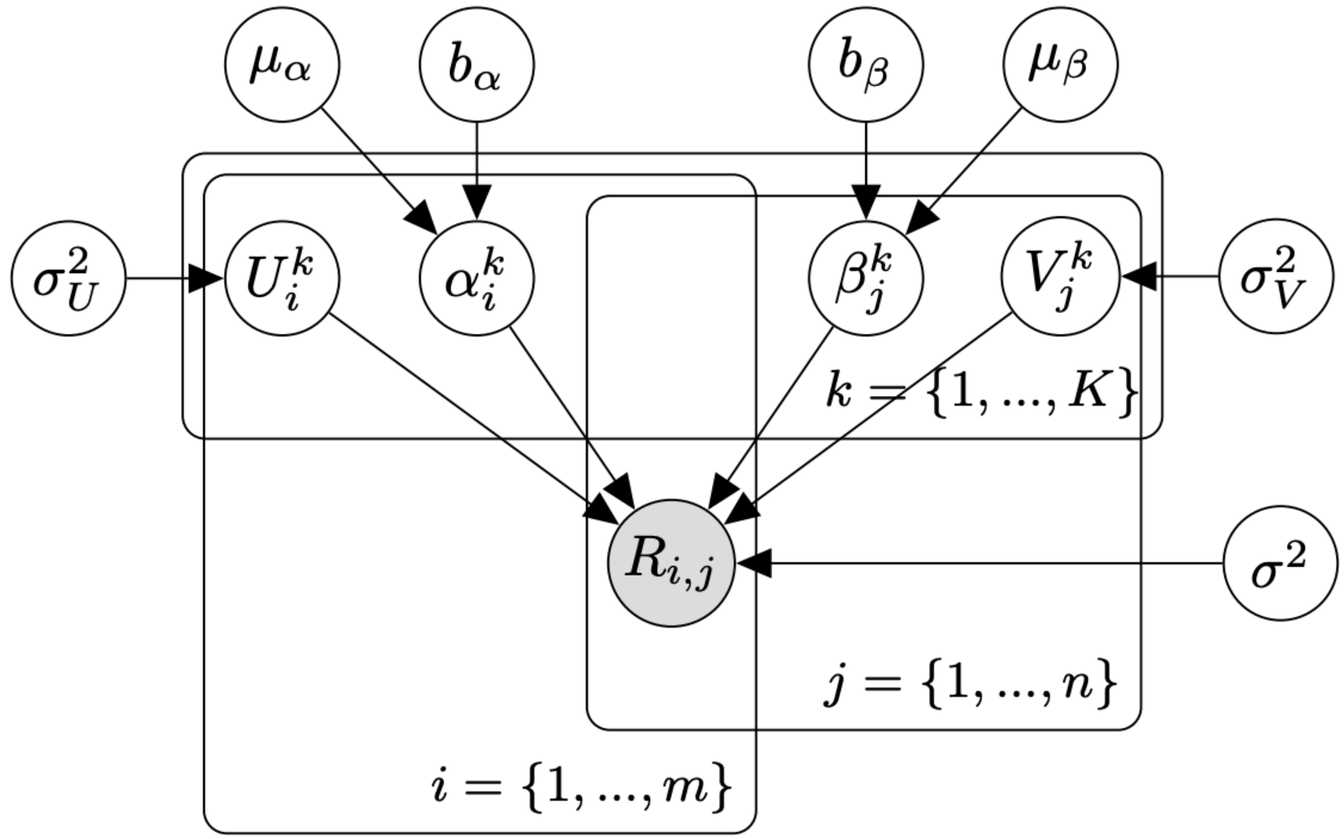
  - Because the models with most suitable ranks for each row/column should be with large weights, i.e., α^k and β^k ***should be sparse***.
  - μ_α and μ_β : location parameters of α and β.
  - b_α and b_β : scale parameters of α and β.

GRAPHICAL MODELS

- MRMA can be viewed as a generalized form of PMF.
  - Setting $K = 1$ in MRMA is the same problem as PMF.



[PMF]



[MRMA]

Posterior Distribution

- The log of the posterior distribution:

$$
\begin{aligned}
l &= \ln \Pr\left(U, V, \alpha, \beta | R, \sigma^2, \sigma_U^2, \sigma_V^2, \mu_\alpha, b_\alpha, \mu_\beta, b_\beta\right) \\
&\propto \ln\left[\Pr\left(R|U, V, \alpha, \beta, \sigma^2\right) \Pr\left(U|\sigma_U^2\right) \Pr\left(V|\sigma_V^2\right) \Pr\left(\alpha|\mu_\alpha, b_\alpha\right) \Pr\left(\beta|\mu_\beta, b_\beta\right)\right] \\
&= \sum_{i=1}^{m}\sum_{j=1}^{n} \mathbf{1}_{i,j} \left[\ln \sum_{k=1}^{K} \alpha_i^k \beta_j^k \mathcal{N}\left(R_{i,j}|U_i^k \left(V_j^k\right)^T, \sigma^2 I\right)\right] \\
&- \frac{1}{2\sigma_U^2} \sum_{k=1}^{K}\sum_{i=1}^{m} \left(U_i^k\right)^2 - \frac{1}{2\sigma_V^2} \sum_{k=1}^{K}\sum_{j=1}^{n} \left(V_i^k\right)^2 - \frac{1}{2} Km \ln \sigma_U^2 - \frac{1}{2} Kn \ln \sigma_V^2 \\
&- \frac{1}{b_\alpha} \sum_{k=1}^{K}\sum_{i=1}^{m} \left|\alpha_i^k - \mu_\alpha\right| - \frac{1}{b_\beta} \sum_{k=1}^{K}\sum_{j=1}^{n} \left|\beta_j^k - \mu_\beta\right| \\
&- \frac{1}{2} \sum_{k=1}^{K} m \ln b_\alpha^2 - \frac{1}{2} \sum_{k=1}^{K} n \ln b_\beta^2 + C
\end{aligned}
$$

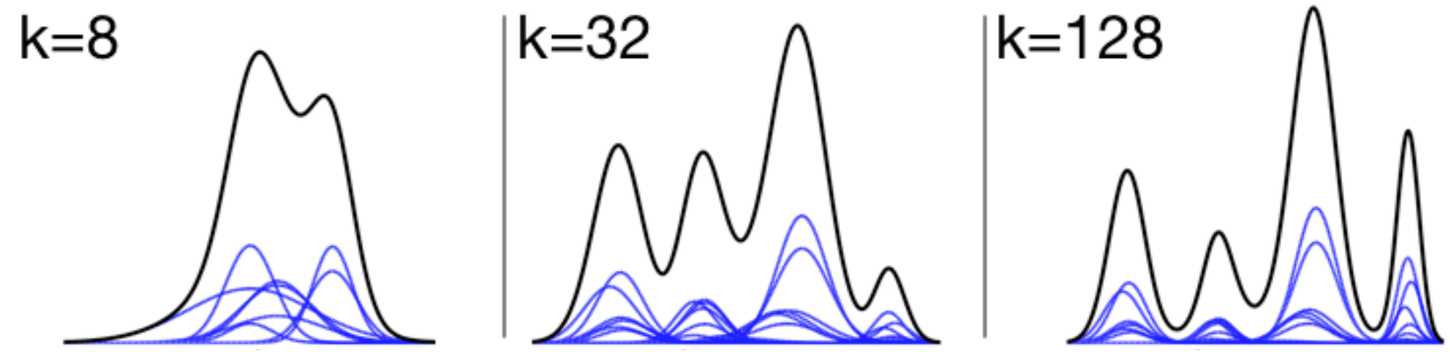
Difficult to solve directly!

Lower Bound of Log Posterior

- Given *i* and *j*,

$$f(U,V) = -\ln\left[\sum_{k=1}^{K}\alpha_i^k\beta_j^k\mathcal{N}\left(R_{i,j}|U_i^k\left(V_j^k\right)^T,\sigma^2 I\right)\right]$$

  - It is mixture of Gaussian distribution.



- If $K > I$, it is difficult to calculate $\frac{d}{dU}f(U,V)$ and $\frac{d}{dV}f(U,V)$

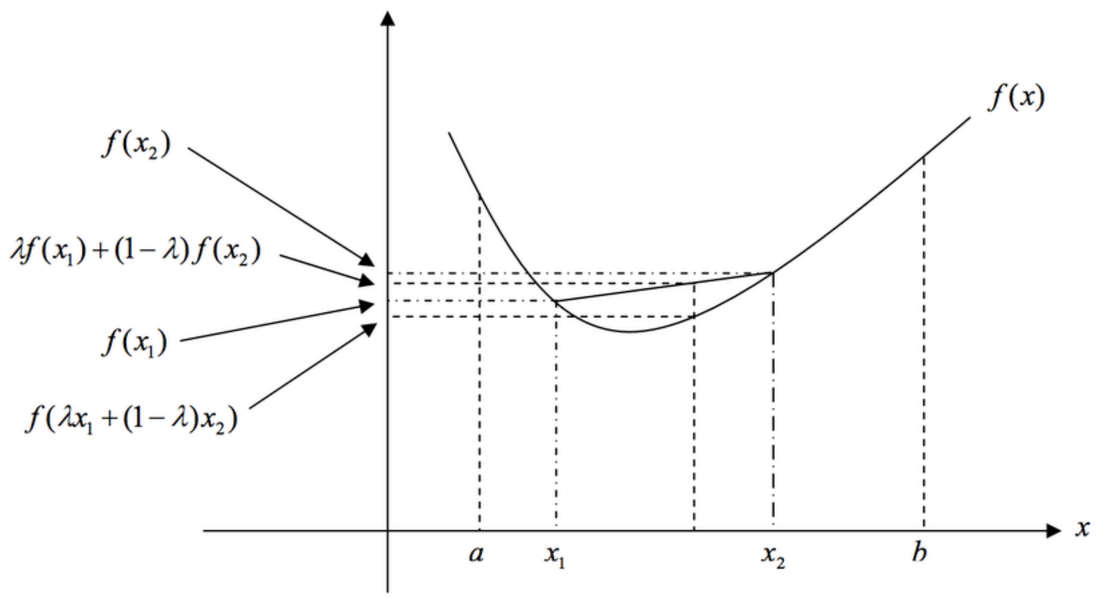
- We need another way to compute **the gradient of *I***.

Lower Bound of Log Posterior

- **Solution: Jensen's Inequality**
  - For a convex function *f*,

$$\forall 0 \leq \lambda \leq 1, f\left(\lambda x_1 + (1-\lambda)x_2\right) \leq \lambda f\left(x_1\right) + (1-\lambda) f\left(x_2\right)$$

- On the other hand, *f(U,V)* is convex. **Why?**
  - Gaussian function and -log(x) function is convex.
  - Sum of multiple convex function is convex.
  - The composite function of two convex function is convex.



Lower Bound of Log Posterior

- Applying Jensen's inequality to *f(U,V)*,

$$f(U,V) = -\ln\left[\sum_{k=1}^{K}\alpha_i^k\beta_j^k\mathcal{N}\left(R_{i,j}|U_i^k\left(V_j^k\right)^T,\sigma^2 I\right)\right]$$

$$\geq -\sum_{k=1}^{K}\alpha_i^k\beta_j^k\ln\mathcal{N}\left(R_{i,j}|U_i^k\left(V_j^k\right)^T,\sigma^2 I\right) = -\frac{1}{2\sigma^2}\sum_{k=1}^{K}\alpha_i^k\beta_j^k\left(R_{i,j}-U_i^k\left(V_j^k\right)^T\right)^2 - \frac{1}{2}\ln\sigma^2 + C'$$

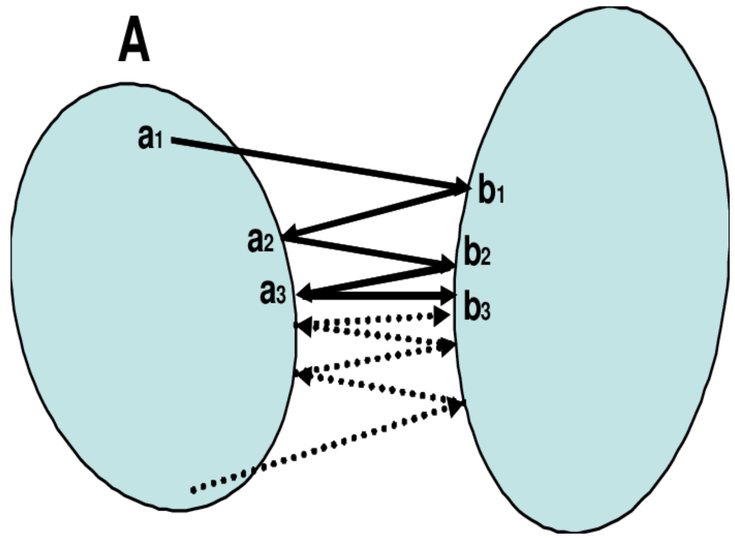
- The lower bound of the log posterior distribution:

$$\begin{aligned} l' = & -\frac{1}{2\sigma^2}\sum_{i=1}^{m}\sum_{j=1}^{n}\mathbb{1}_{i,j}\left[\sum_{k=1}^{K}\alpha_i^k\beta_j^k(R_{i,j}-U_i^k(V_j^k)^T)^2\right] - \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{n}\mathbb{1}_{i,j}\ln\sigma^2 \\ & -\frac{1}{2\sigma_U^2}\sum_{k=1}^{K}\sum_{i=1}^{m}(U_i^k)^2 - \frac{1}{2\sigma_V^2}\sum_{k=1}^{K}\sum_{j=1}^{n}(V_i^k)^2 - \frac{1}{2}Km\ln\sigma_U^2 - \frac{1}{2}Kn\ln\sigma_V^2 \\ & -\frac{1}{b_\alpha}\sum_{k=1}^{K}\sum_{i=1}^{m}|\alpha_i^k-\mu_\alpha| - \frac{1}{b_\beta}\sum_{k=1}^{K}\sum_{j=1}^{n}|\beta_j^k-\mu_\beta| - \frac{1}{2}Km\ln b_\alpha^2 - \frac{1}{2}Kn\ln b_\beta^2 + C. \end{aligned}$$

Iterated Conditional Modes (ICM)

- On the other hands, the loss function is difficult to compute.
  - Too many **parameters** and **hyper-parameters**.

- Iterated Conditional Modes (ICM) Steps [J. Besag, 1986]:
  - Conditioned on the rest variables, then iterate until convergence. (Alternating minimization)
  - To estimate optimal value of each variables by conditioning other variables.
  - Similar to works of Kittler and Foglein [1984], Kiiveri and Campbell [1986].



- In MRMA, the hyper-parameters are also trained by ICM.

ICM OF MRMA (PARAMETERS)

- Each parameters can be updated by solving the following minimization:

$$\forall k \in \{1, ..., K\}, \forall i \in \{1, ..., m\}:$$

$$U_i^k \leftarrow \arg\min_{U'} \Big\{ \frac{1}{2\sigma^2} \sum_{j=1}^{n} \mathbb{1}_{i,j} \Big[\sum_{k=1}^{K} \alpha_i^k \beta_j^k (R_{i,j} - U_i^k (V_j^k)^T)^2 \Big] + \frac{1}{2\sigma_U^2} \sum_{k=1}^{K} (U_i^k)^2 \Big\},$$

$$\alpha_i^k \leftarrow \arg\min_{\alpha'} \Big\{ \frac{1}{2\sigma^2} \sum_{j=1}^{n} \mathbb{1}_{i,j} \Big[\sum_{k=1}^{K} \alpha_i^k \beta_j^k (R_{i,j} - U_i^k (V_j^k)^T)^2 \Big] + \frac{1}{b_\alpha} \sum_{k=1}^{K} |\alpha_i^k - \mu_\alpha| \Big\}.$$

$$\forall k \in \{1, ..., K\}, \forall j \in \{1, ..., n\}:$$

$$V_j^k \leftarrow \arg\min_{V'} \Big\{ \frac{1}{2\sigma^2} \sum_{i=1}^{m} \mathbb{1}_{i,j} \Big[\sum_{k=1}^{K} \alpha_i^k \beta_j^k (R_{i,j} - U_i^k (V_j^k)^T)^2 \Big] + \frac{1}{2\sigma_V^2} \sum_{k=1}^{K} (V_j^k)^2 \Big\},$$

$$\beta_j^k \leftarrow \arg\min_{\beta'} \Big\{ \frac{1}{2\sigma^2} \sum_{i=1}^{m} \mathbb{1}_{i,j} \Big[\sum_{k=1}^{K} \alpha_i^k \beta_j^k (R_{i,j} - U_i^k (V_j^k)^T)^2 \Big] + \frac{1}{b_\beta} \sum_{k=1}^{K} |\beta_j^k - \mu_\beta| \Big\}.$$

- **These all convex optimization!**

ICM OF MRMA (HYPER-PARAMETERS)

- Each hyper-parameters can be updated by solving the following minimization:

$$\sigma^2 \leftarrow \sum_{i=1}^{m}\sum_{j=1}^{n} \mathbb{1}_{i,j}\big[\sum_{k=1}^{K} \alpha_i^k \beta_j^k (R_{i,j} - U_i^k (V_j^k)^T)^2\big] / \sum_{i=1}^{m}\sum_{j=1}^{n} \mathbb{1}_{i,j},$$

$$\sigma_U^2 \leftarrow \sum_{k=1}^{K}\sum_{i=1}^{m} (U_i^k)^2 / Km, \;\; \mu_\alpha \leftarrow \sum_{k=1}^{K}\sum_{i=1}^{m} \alpha_i^k / Km, \;\; b_\alpha = \sum_{k=1}^{K}\sum_{i=1}^{m} |\alpha_i^k - \mu_\alpha| / Km,$$

$$\sigma_V^2 \leftarrow \sum_{k=1}^{K}\sum_{j=1}^{n} (V_j^k)^2 / Kn, \;\; \mu_\beta \leftarrow \sum_{k=1}^{K}\sum_{j=1}^{n} \beta_j^k / Kn, \;\; b_\beta = \sum_{k=1}^{K}\sum_{j=1}^{n} |\beta_j^k - \mu_\beta| / Kn.$$

- The hyper-parameters can be learned as their maximum likelihood estimates by setting their partial derivatives on *l'* to 0. **Why?**
  - These all convex optimization!
  - Cons: the closed form solution exists.
- **Without ICM, we can not get good performance.**

WHY NOT UPDATING *U* AND *V* BY CLOSED FORM?

- *U,V* updating procedures can be seen as some form of ridge regression.
- Therefore, **the closed form solution exists!**
- Then, why not MRMA use closed form solution for *U* and *V*?

- Netflix Prize Winner said that
  "SGD shows better performance compared to closed form solution".
  - Many empirical studies have supported this claim. **Still open question!**

- (CF) Two main algorithms for matrix factorization:
  - **1. Stochastic Gradient Descent (SGD)**
  - 2. Alternating Minimization (closed form solution)
    - Sub-optimality has been proven [Jain et al 2013].
    - But performance is worse than SGD and slow on GPU (SVD calculation)

TRAINING OF MRMA

- Besag [1986] said that the performance of ICM is **highly sensitive** to **initial value** of the parameters and hyper-parameters.

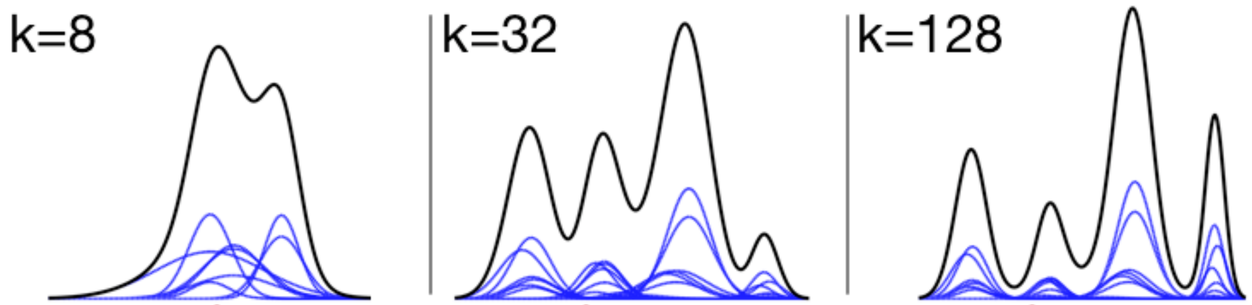
- Therefore, MRMA present the initial values.
  - *U,V* : the solution of classical PMF algorithm.
  - α^k and β^k: $1/\sqrt{K}$ (Bagging)

- I reproduced MRMA using tensorflow. **Difficult to reproduce!**
  - Even, the authors of RaFM (ICML 2019) said that they can not reproduce the performance of MRMA paper.
  - I can get better performance than MRMA's report on Movielen 1M (RMSE 0.833).
  - Really sensitive to initial value.
  - My implementation: https://github.com/JoonyoungYi/MRMA-tensorflow

How to Compute the Prediction of *R*?

- I guessed how to estimate *R* because it did not appear in the MRMA paper.

- Even if we have all the parameters, it is difficult to estimate *R* using *I*.



  - They have to estimate *R* using its lower bound *I'*.

- In *I'*, $\sum_{k=1}^{K} \alpha_i^k \beta_j^k (R_{i,j} - U_i^k (V_j^k)^T)^2$.

  weight *each sub-model*

  - This can be seen as weighted average of each sub-models.

- Hence, $\hat{R}_{i,j} = \sum_{k=1}^{K} \alpha_i^k \beta_j^k U_i^k \left(V_j^k\right)^T / \sum_{k=1}^{K} \alpha_i^k \beta_j^k$

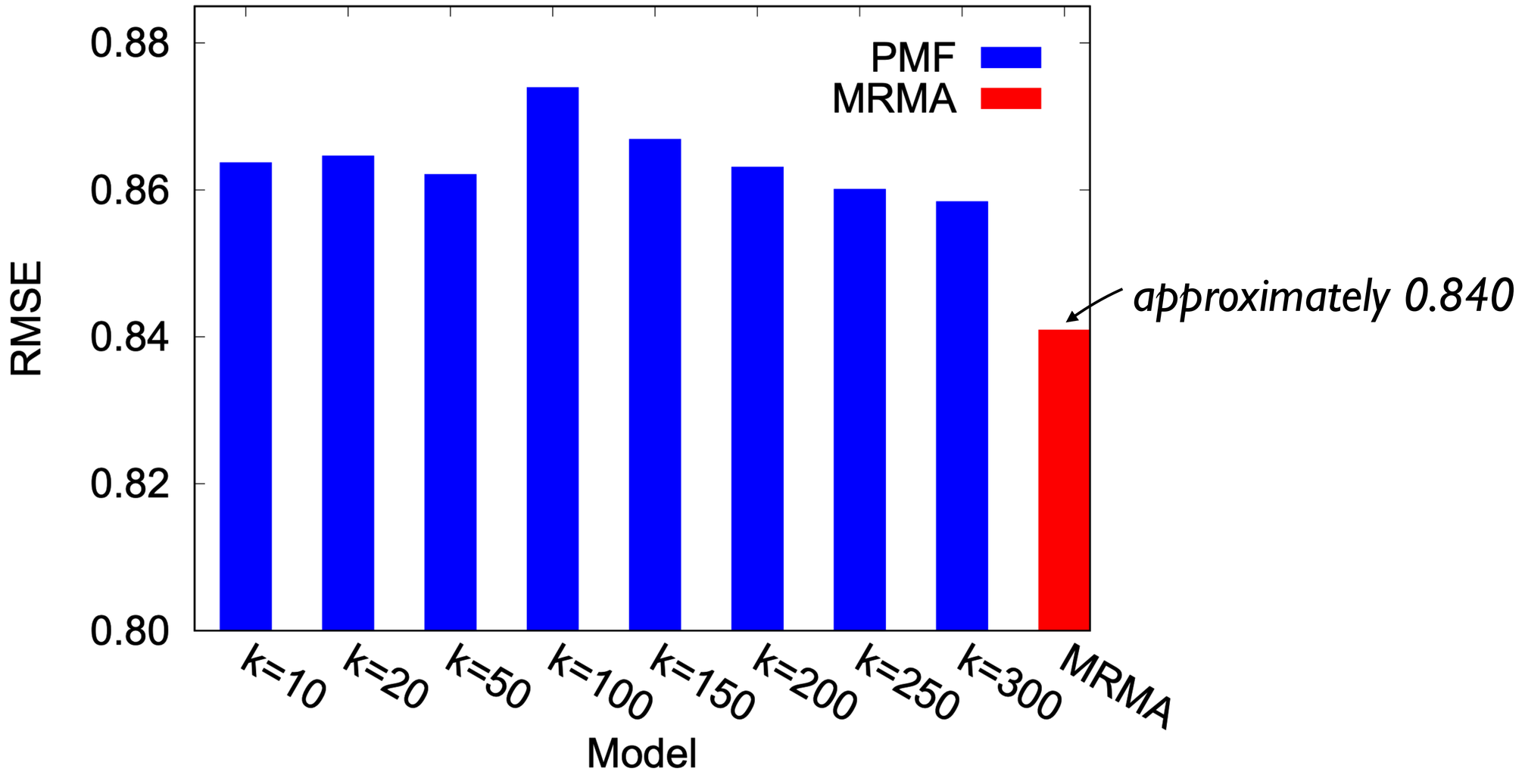
CONTENTS

KAIST

MLILAB
Machine Learning & Intelligence

VS. FIXED-RANK MATRIX APPROXIMATION

- Is MRMA effective compared to PMF? (Experiments on Movielens 1M)



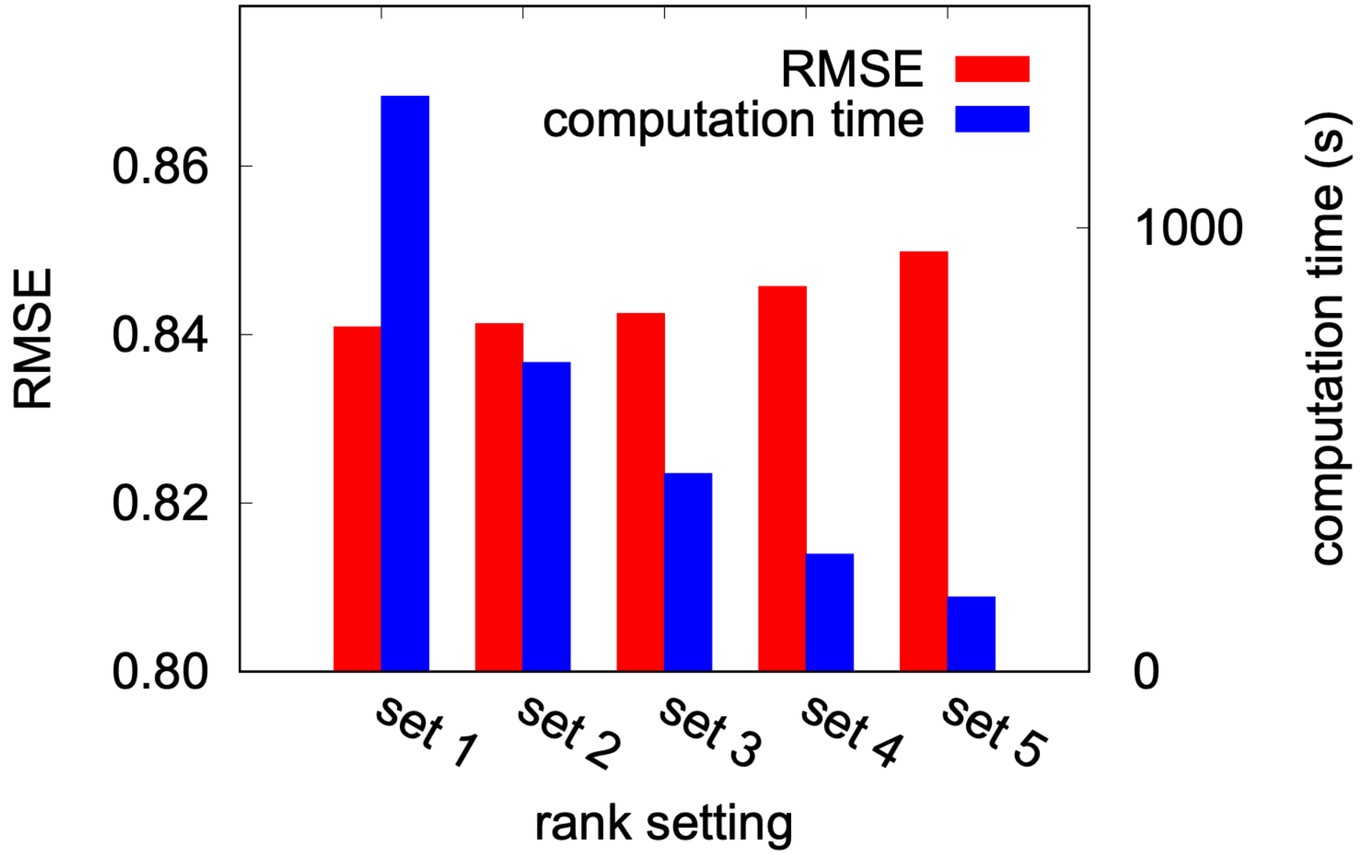
- I think experimental results of PMF are not natural.
  (The relationship between RMSE and *k* is inconsistent.)

SENSITIVITY OF RANK

- The set of ranks decide the performance of the final model.
- However, it is **not necessary** to choose all the ranks in [*1*, *2*, …, *K*].
  - The parameter may be too large and the calculation may not be efficient.

- **Assumption: Overlapping structure**
  - A rank-k approx. will be very similar to rank-(*k* − *1*) and rank-(*k* + *1*) approx..

- Experimental Settings:
  - set 1 = {*10, 20, 30, …, 300*}
  - set 2 = {*20, 40, 60, …, 300*}
  - set 3 = {*30, 60, 90, …, 300*}
  - set 4 = {*50, 100, 150, …, 300*}
  - set 5 = {*100, 200, 300*}



Accuracy Comparison (Rating Prediction)

- The set of ranks: {*10, 20, 50, 100, 150, 200, 250, 300*}
  - This set showed good performance emperically.

- The state-of-the-arts performance on Movielens 10M and Netflix datasets.
  - Table summarized RMSE with 95% confidence levels

		MovieLens (10M)	Netflix
single model	BPMF	0.8197 ± 0.0004	0.8421 ± 0.0002
	GSMF	0.8012 ± 0.0011	0.8420 ± 0.0006
	LLORMA	0.7855 ± 0.0002	0.8275 ± 0.0004
	WEMAREC	0.7775 ± 0.0007	0.8143 ± 0.0001
	MPMA	0.7712 ± 0.0002	0.8139 ± 0.0003
	SMA	0.7682 ± 0.0003	0.8036 ± 0.0004
	MRMA	$\mathbf{0.7634 \pm 0.0009}$	$\mathbf{0.7973 \pm 0.0002}$

- Better performance compared to other matrix approximation medthods.
  - Better than the studies after MRMA (ex. GLOMA, ABCF).

ACCURACY COMPARISON (ITEM RANKING)

- NDCG@N: Another metric that measures recommendation performance.
- Unlike other algorithms, this algorithm showed good performance in **both NDCG and RMSE**.

Metric		NDCG@N			
Data	Method	N=1	N=5	N=10	N=20
Movielens 1M	BPMF	0.6870 ± 0.0024	0.6981 ± 0.0029	0.7525 ± 0.0009	0.8754 ± 0.0008
	GSMF	0.6909 ± 0.0048	0.7031 ± 0.0023	0.7555 ± 0.0017	0.8769 ± 0.0011
	LLORMA	0.7025 ± 0.0027	0.7101 ± 0.0005	0.7626 ± 0.0023	0.8811 ± 0.0010
	WEMAREC	0.7048 ± 0.0015	0.7089 ± 0.0016	0.7617 ± 0.0041	0.8796 ± 0.0005
	MPMA	0.7020 ± 0.0005	0.7114 ± 0.0018	0.7606 ± 0.0006	0.8805 ± 0.0007
	SMA	0.7042 ± 0.0033	0.7109 ± 0.0011	0.7607 ± 0.0008	0.8801 ± 0.0004
	MRMA	**0.7153 ± 0.0027**	**0.7182 ± 0.0005**	**0.7672 ± 0.0013**	**0.8837 ± 0.0004**
Movielens 10M	BPMF	0.6563 ± 0.0005	0.6845 ± 0.0003	0.7467 ± 0.0007	0.8691 ± 0.0002
	GSMF	0.6708 ± 0.0012	0.6995 ± 0.0008	0.7566 ± 0.0017	0.8748 ± 0.0004
	LLORMA	0.6829 ± 0.0014	0.7066 ± 0.0005	0.7632 ± 0.0004	0.8782 ± 0.0012
	WEMAREC	0.7013 ± 0.0003	0.7176 ± 0.0006	0.7703 ± 0.0002	0.8824 ± 0.0006
	MPMA	0.6908 ± 0.0006	0.7133 ± 0.0002	0.7680 ± 0.0001	0.8808 ± 0.0004
	SMA	0.7002 ± 0.0006	0.7134 ± 0.0004	0.7679 ± 0.0003	0.8809 ± 0.0002
	MRMA	**0.7048 ± 0.0006**	**0.7219 ± 0.0001**	**0.7743 ± 0.0001**	**0.8846 ± 0.0001**

INTERPRETATION OF MRMA

- Does MRMA tackle the issue related to their observation?
- Top 10 movies with largest β values with rank *k=20* and *k=200:*

rank=20			rank=200		
movie name	β	#ratings	movie name	β	#ratings
Smashing Time	0.6114	2.4	American Beauty	0.9219	1781.4
Gate of Heavenly Peace	0.6101		Groundhog Day	0.9146	
Man of the Century	0.6079		Fargo	0.8779	
Mamma Roma	0.6071		Face/Off	0.8693	
Dry Cleaning	0.6071		2001: A Space Odyssey	0.8608	
Dear Jesse	0.6063		Shakespeare in Love	0.8553	
Skipped Parts	0.6057		Saving Private Ryan	0.8480	
The Hour of the Pig	0.6055		The Fugitive	0.8404	
Inheritors	0.6042		Braveheart	0.8247	
Dangerous Game	0.6034		Fight Club	0.8153	

- The movies appropriated to a smaller rank are less evaluated.
- The movies appropriated to a higher rank are more evaluated.

CONTENTS

KAIST

MLILAB
Machine Learning & Intelligence

Discussion Points

- Is there any idea for replacing ICM?
  - The ICM is too dependent on the initial value.
  - It takes too long to run the classical PMF algorithms for every rank *k*.

- Can not we estimate $\hat{R}$ in the original model rather than estimate $\hat{R}$ in the lower bound?
  - The estimation of MRMA comes from its lower bound.

- Is there any way to use the tight lower bound or expectation?

- The current rank setting is too heuristic. Is there any better way?

THE PERFORMANCE ON SMALL SIZE DATASET

- The current state-of-the-arts table [4] on Movielens 100k, 1M, 10M:

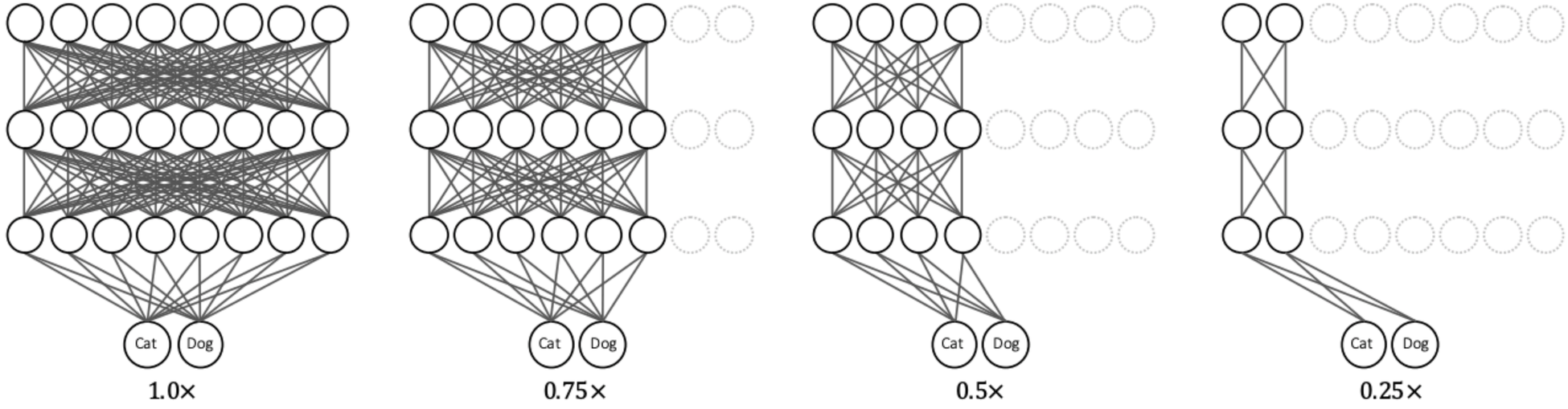
Models	Movielens 100K	Models	Movielens 1M	Models	Movielens 10M
ReDa [46]	0.911	NNMF [47]	0.843	GC-MC [48]	0.777
SVD++ [49]	0.913 †	ABCF [50]	0.836	MPMA [51]	0.771
Biased MF [52]	0.911 †	LLoRMA [53]	0.833	CF-NADE [17]	0.771
NNMF [47]	0.907	DMF+ [54]	0.832	SMA [55]	0.768
AutoSVD [56]	0.901	GC-MC [48]	0.832	GLOMA [57]	0.767
DMF+[54]	0.889	AutoRec [16]	0.831	ABCF [50]	0.766
LLoRMA [53]	0.888	CF-NADE [17]	0.829	MRMA [58]	**0.763**
AutoRec w/SN	**0.882**	**AutoRec w/SN**	**0.826**	**AutoRec w/SN**	0.769

- RMSE of MRMA on Movielens 100k: 0.893 (reproduction)
- RMSE of MRMA on Movielens 1M: 0.833 (reproduction)

- Why we can't get the best performance with relatively small dataset such as 100k and 1m?

WEIGHT SHARING

- Inspired by Slimmable Network [2].



- The first column of U^1 and U^2 may be similar.
- Sharing the first column of U^1 and U^2 would reduce the number of parameters drastically and enable efficient learning.

- If using weight sharing,
  - would not be necessary to worry about how to set a rank set.
  - would not be necessary to initialize using PMF?

Any Questions?

KAIST

REFERENCE

- [1] Li, Dongsheng, et al. "Mixture-rank matrix approximation for collaborative filtering." *Advances in Neural Information Processing Systems*. 2017.
- [2] Yu, Jiahui, and Thomas Huang. "Universally Slimmable Networks and Improved Training Techniques." *arXiv preprint arXiv:1903.05134* (2019).
- [3] Chen, Xiaoshuang, et al. "RaFM: Rank-Aware Factorization Machines." *arXiv preprint arXiv:1905.07570* (2019).
- [4] Yi, Joonyoung, et al. "Sparsity Normalization: Stabilizing the Expected Outputs of Deep Networks." *arXiv preprint arXiv:1906.00150* (2019).