

# Scalable Variational Bayesian Matrix Factorization

**Yong-Deok Kim and Seungjin Choi**

Machine Learning Laboratory

POSTECH, Korea

MLG  
POSTECH



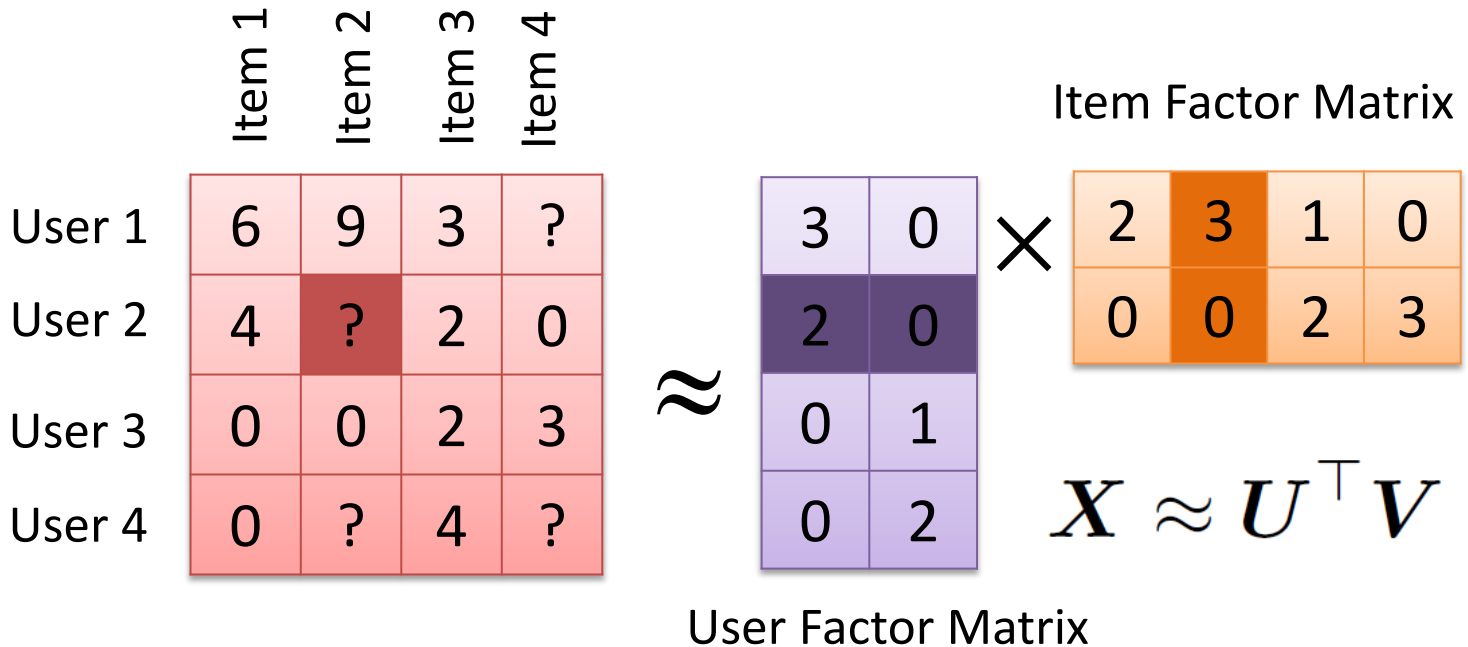
LSRS-13

**POSTECH**

# Outline

1. Matrix Factorization  
for Collaborative Prediction
2. Regularized Matrix Factorization  
vs. Bayesian Matrix Factorization
3. Scalable Variational Bayesian  
Matrix Factorization
4. Related Works
5. Numerical Experiments
6. Conclusion

# Matrix Factorization for Collaborative Prediction



- **Collaborative prediction**
  - ➔ Filling missing entries of the user-item rating matrix
- **Matrix factorization**
  - ➔ Predicting an unknown rating by product of user factor vector and item factor vector

# Regularized Matrix Factorization

- Minimize the regularized squared error loss

$$\sum_{(i,j) \in \Omega} [(X_{ij} - \mathbf{u}_i^\top \mathbf{v}_j)^2 + \lambda(\|\mathbf{u}_i\|^2 + \|\mathbf{v}_j\|^2)]$$

## Alternating Least Squares (ALS)

```
Initialize  $U, V$ 
for  $t = 1 \dots T$  do
  /***** Update  $U$  *****/
  parallel for  $i = 1, \dots, I$  do
     $\mathbf{u}_i \leftarrow \left( \lambda |\Omega_i| \mathbf{I} + \sum_{j \in \Omega_i} \mathbf{v}_j \mathbf{v}_j^\top \right)^{-1} \sum_{j \in \Omega_i} X_{ij} \mathbf{v}_j$ 
  end parallel for
  /***** Update  $V$  *****/
  for  $j = 1, \dots, J$  do
     $\mathbf{v}_j \leftarrow \left( \lambda |\Omega_j| \mathbf{I} + \sum_{i \in \Omega_j} \mathbf{u}_i \mathbf{u}_i^\top \right)^{-1} \sum_{i \in \Omega_j} X_{ij} \mathbf{u}_i$ 
  end for
end for
```

Time complexity

$$\rightarrow O(2|\Omega|K^2 + (I+J)K^3)$$

Parallelization

$\rightarrow$  Easy

Tuning parameter

$\rightarrow \lambda$  (regularization)

# Regularized Matrix Factorization

- Minimize the regularized squared error loss

$$\sum_{(i,j) \in \Omega} [(X_{ij} - \mathbf{u}_i^\top \mathbf{v}_j)^2 + \lambda(\|\mathbf{u}_i\|^2 + \|\mathbf{v}_j\|^2)]$$

## Stochastic Gradient Descent (SGD)

```
Initialize  $U, V$ 
for  $t = 1 \dots T$  do
  for  $(i, j) \in \Omega$  do
     $R_{ij} \leftarrow X_{ij} - \mathbf{u}_i^\top \mathbf{v}_j$ 
     $\mathbf{u}_i \leftarrow \mathbf{u}_i - \eta(-R_{ij}\mathbf{v}_j + \lambda\mathbf{u}_i)$ 
     $\mathbf{v}_j \leftarrow \mathbf{v}_j - \eta(-R_{ij}\mathbf{u}_i + \lambda\mathbf{v}_j)$ 
  end for
end for
```

Time complexity

$$\rightarrow O(2|\Omega|K)$$

Parallelization

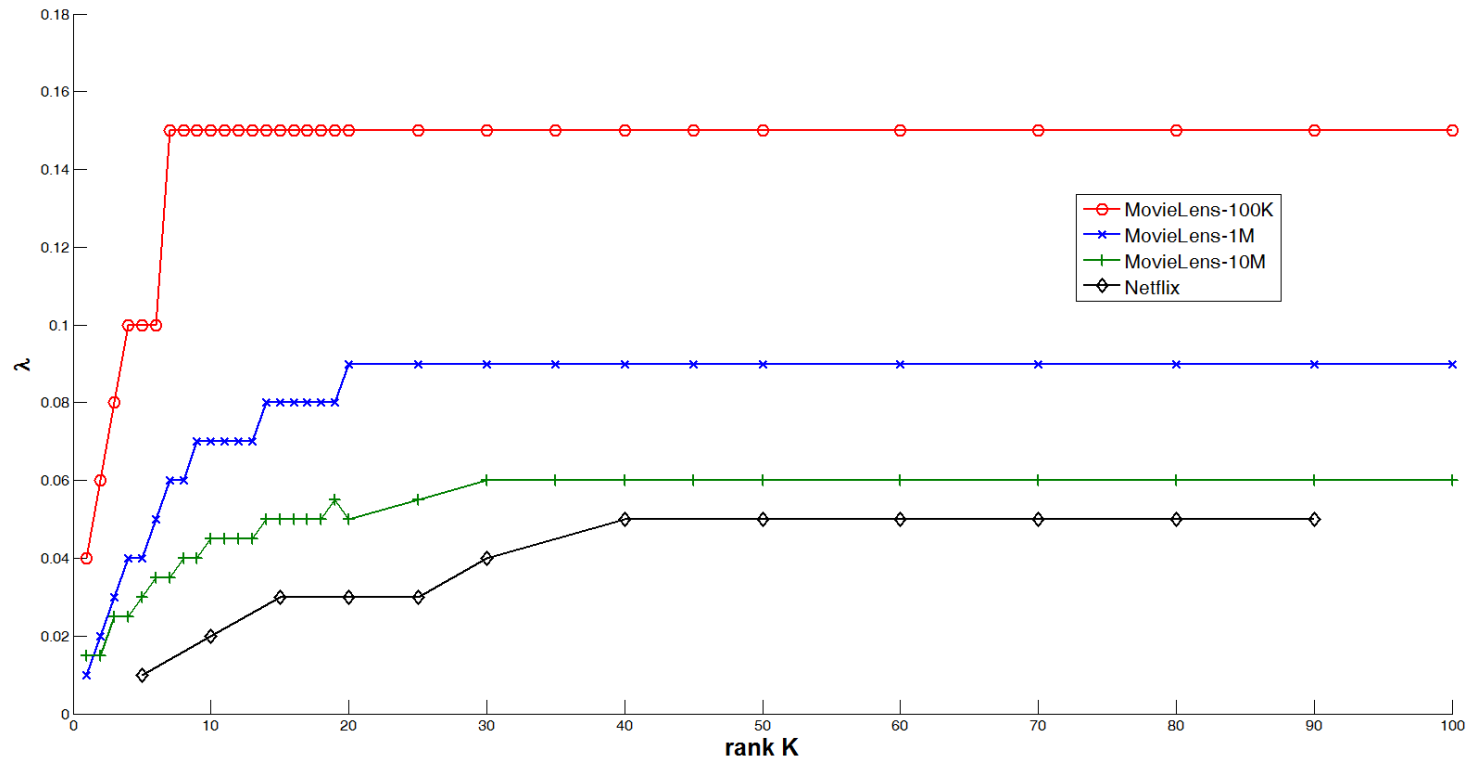
$\rightarrow$  Possible, but not easy

Tuning parameter

$\rightarrow \lambda$  (regularization)  
 $\rightarrow \eta$  (learning rate)

# Problem of parameter tuning

- The value of optimal regularization parameter depend on the dataset and rank  $K$ .



Regularization parameter chosen by cross-validation on various datasets and rank  $K$  (Kim & Choi, IEEE SPL 2013)

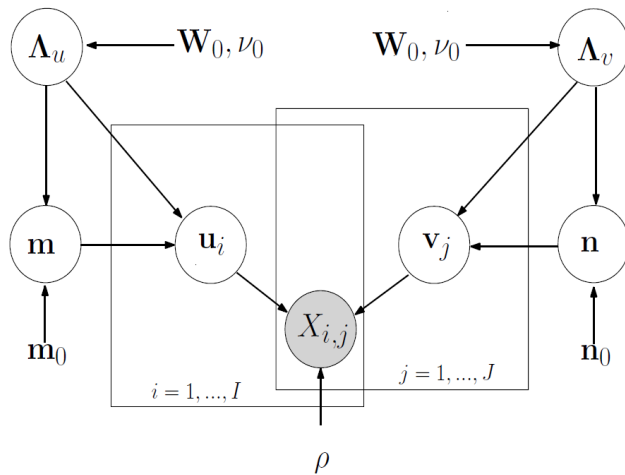
# Problem of parameter tuning

- SGD require tuning of regularization parameter, learning rate and even the number of epochs.

| $\lambda \backslash \eta$ | 0.005      | 0.007      | 0.010      | 0.015      | 0.020      |
|---------------------------|------------|------------|------------|------------|------------|
| 0.005                     | 0.9601/ 13 | 0.9079/ 15 | 0.9117/ 19 | 0.9168/ 28 | 0.9168/ 44 |
| 0.007                     | 0.9056/ 10 | 0.9074/ 11 | 0.9112/ 13 | 0.9168/ 19 | 0.9169/ 31 |
| 0.010                     | 0.9064/ 7  | 0.9077/ 8  | 0.9113/ 10 | 0.9174/ 13 | 0.9186/ 21 |
| 0.015                     | 0.9099/ 5  | 0.9011/ 6  | 0.9152/ 6  | 0.9257/ 7  | 0.9390/ 7  |
| 0.020                     | 0.9166/ 4  | 0.9175/ 4  | 0.9217/ 4  | 0.9314/ 4  | 0.9431/ 3  |

Netflix probe10 RMSE/optimal number of epochs of the BRSIMF for various  $\eta$  and  $\lambda$  values ( $K=40$ ). (Tákacs et al., JMLR 2009)

# Bayesian Matrix Factorization

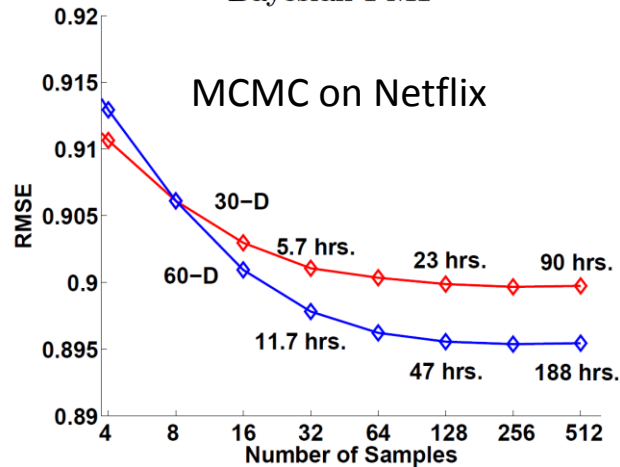


$$\text{Prior } P(U), P(V) \times \text{Likelihood } P(X | U, V) \propto \text{Posterior } P(U, V | X)$$

Approximate the posterior by

- MCMC (Salakhutdinov & Mnih, NIPS 2008)
- Variational method (Lim & Teh, KDDcup 2007)

Bayesian PMF



- ↑ No parameter tuning
- ↑ No overfitting
- ↑ High accuracy
- ↓ Huge computational cost  $O(2|\Omega|K^2 + (I+J)K^3)$

# Scalable Variational Bayesian Matrix Factorization

- No parameter tuning
- Linear space complexity:  $O(2(I+J)K)$
- Linear time complexity:  $O(6|\Omega|K)$
- Easily parallelized on multi-core systems
- Optimize  
element-wisely factorized variational distribution

$$q(\mathbf{U}, \mathbf{V}) = \prod_{k=1}^K \prod_{i=1}^I \mathcal{N}(u_{ki} | \bar{u}_{ki}, s_{ki}^u) \prod_{k=1}^K \prod_{j=1}^J \mathcal{N}(v_{kj} | \bar{v}_{kj}, s_{kj}^v)$$

with coordinate descent method.

# Variational Bayesian Matrix Factorization

- Likelihood is given by

$$p(\mathbf{X} | \mathbf{U}, \mathbf{V}, \tau) = \prod_{(i,j) \in \Omega} \mathcal{N}(X_{ij} | \mathbf{u}_i^\top \mathbf{v}_j, \tau^{-1})$$

- Gaussian priors on factor matrices  $\mathbf{U}$  and  $\mathbf{V}$ :

$$p(\mathbf{U} | \boldsymbol{\alpha}) = \prod_{k=1}^K \prod_{i=1}^I \mathcal{N}(u_{ki} | 0, \alpha_k^{-1}), \quad p(\mathbf{V} | \boldsymbol{\beta}) = \prod_{k=1}^K \prod_{j=1}^J \mathcal{N}(v_{kj} | 0, \beta_k^{-1})$$

- Approximate posterior by variational distribution by **maximizing the variational lower bound**, or equivalently minimizing the KL-divergence

$$\begin{aligned} \mathcal{F}(q) &= \int \int q(\mathbf{U}, \mathbf{V}) \log \frac{p(\mathbf{X} | \mathbf{U}, \mathbf{V}) p(\mathbf{U}) p(\mathbf{V})}{q(\mathbf{U}, \mathbf{V})} d\mathbf{U} d\mathbf{V} \\ &= \log p(\mathbf{X}) - \text{KL}(q(\mathbf{U}, \mathbf{V}) \| p(\mathbf{U}, \mathbf{V} | \mathbf{X})) \end{aligned}$$

# VBMF-BCD (Lim & The KDDcup 2007)

- Matrix-wisely factorized variational distribution

$$p(\mathbf{U}, \mathbf{V} | \mathbf{X}) \approx q(\mathbf{U}, \mathbf{V}) = q(\mathbf{U})q(\mathbf{V}) = \prod_{i=1}^I \mathcal{N}(\mathbf{u}_i | \bar{\mathbf{u}}_i, \mathbf{S}_i^u) \prod_{j=1}^J \mathcal{N}(\mathbf{v}_j | \bar{\mathbf{v}}_j, \mathbf{S}_j^v)$$

## VBMF-BCD

```
Initialize  $\bar{\mathbf{U}}, \{\mathbf{S}_i^u\}_{i=1}^I, \bar{\mathbf{V}}, \{\mathbf{S}_j^v\}_{j=1}^J$ 
for  $t = 1 \dots T$  do
  /***** Update  $\mathbf{U}$  *****/
  parallel for  $i = 1, \dots, I$  do
     $\mathbf{S}_i^u \leftarrow \left( \text{diag}(\alpha) + \tau \sum_{j \in \Omega_i} (\bar{\mathbf{v}}_j \bar{\mathbf{v}}_j^\top + \mathbf{S}_j^v) \right)^{-1}$ 
     $\bar{\mathbf{u}}_i \leftarrow \mathbf{S}_i^u (\tau \sum_{j \in \Omega_i} X_{ij} \bar{\mathbf{v}}_j)$ 
  end parallel for
  /***** Update  $\mathbf{V}$  *****/
  for  $j = 1, \dots, J$  do
     $\mathbf{S}_j^v \leftarrow \left( \text{diag}(\beta) + \tau \sum_{i \in \Omega_j} (\bar{\mathbf{u}}_i \bar{\mathbf{u}}_i^\top + \mathbf{S}_i^u) \right)^{-1}$ 
     $\bar{\mathbf{v}}_j \leftarrow \mathbf{S}_j^v (\tau \sum_{i \in \Omega_j} X_{ij} \bar{\mathbf{u}}_i)$ 
  end for
end for
```

Space complexity

$$\rightarrow O((I+J)(K+K^2))$$

Time complexity

$$\rightarrow O(2|\Omega|K^2 + (I+J)K^3)$$

Parallelization

$\rightarrow$  Easy

# Scalable VBMF: linear space complexity

$$q(\mathbf{U}, \mathbf{V}) = \prod_{i=1}^I \mathcal{N}(\mathbf{u}_i | \bar{\mathbf{u}}_i, \mathbf{S}_i^u) \prod_{j=1}^J \mathcal{N}(\mathbf{v}_j | \bar{\mathbf{v}}_j, \mathbf{S}_j^v)$$



**Element-wisely factorized variational distribution**

$$q(\mathbf{U}, \mathbf{V}) = \prod_{k=1}^K \prod_{i=1}^I \mathcal{N}(u_{ki} | \bar{u}_{ki}, s_{ki}^u) \prod_{k=1}^K \prod_{j=1}^J \mathcal{N}(v_{kj} | \bar{v}_{kj}, s_{kj}^v)$$

| $K=100$  | $O((I+J)(K+K^2))$ | $O(2(I+J)K)$ |
|--|-------------------|--------------|
| <b>Netflix</b><br>$I = 480,189$<br>$J = 17,770$        | 4.4 GB            | 0.8 GB       |
| <b>Yahoo-music</b><br>$I = 1,000,990$<br>$J = 624,961$ | 131 GB            | 2.6 GB       |

# Scalable VBMF: quadratic time complexity

Updating rules for  $q(u_{ki})$

$$\begin{aligned} s_{ki}^u &= \left( \alpha_k + \tau \sum_{j \in \Omega_i} (\bar{v}_{kj}^2 + s_{kj}^v) \right)^{-1}, \\ \bar{u}_{ki} &= s_{ki}^u \left( \tau \sum_{j \in \Omega_i} (x_{ij} - \sum_{k' \neq k} \bar{u}_{k'i} \bar{v}_{k'j}) \bar{v}_{kj} \right) \end{aligned} \quad O(|\Omega_i|K)$$

Updating all variational parameters  $\bar{U}$ ,  $S^u$ ,  $\bar{V}$ , and  $S^v$

$$O \left( \sum_{k=1}^K \sum_{i=1}^I |\Omega_i|K + \sum_{k=1}^K \sum_{j=1}^J |\Omega_j|K \right) = O(2|\Omega|K^2)$$

# Scalable VBMF: linear time complexity

Let  $R_{ij}$  denote the residual on  $(i, j)$  observation:

$$R_{ij} = X_{ij} - \bar{\mathbf{u}}_i^\top \bar{\mathbf{v}}_j = X_{ij} - \sum_{k=1}^K \bar{u}_{ki} \bar{v}_{kj}$$

With  $R_{ij}$  updating rule can be rewritten as

$$\bar{u}_{ki} = s_{ki}^u \left( \tau \sum_{j \in \Omega_i} (x_{ij} - \sum_{k' \neq k} \bar{u}_{k'i} \bar{v}_{k'j}) \bar{v}_{kj} \right) \quad O(|\Omega_i|K)$$



$$\bar{u}_{ki} = s_{ki}^u \left( \tau \sum_{j \in \Omega_i} (R_{ij} + \bar{u}_{ki} \bar{v}_{kj}) \bar{v}_{kj} \right) \quad O(|\Omega_i|)$$

# Scalable VBMF: linear time complexity

When  $\bar{u}_{ki}$  is changed to  $\bar{u}'_{ki}$ ,  $R_{ij}$  can be easily updated to  $R'_{ij} = R_{ij} - (\bar{u}'_{ki} - \bar{u}_{ki})\bar{v}_{kj}$

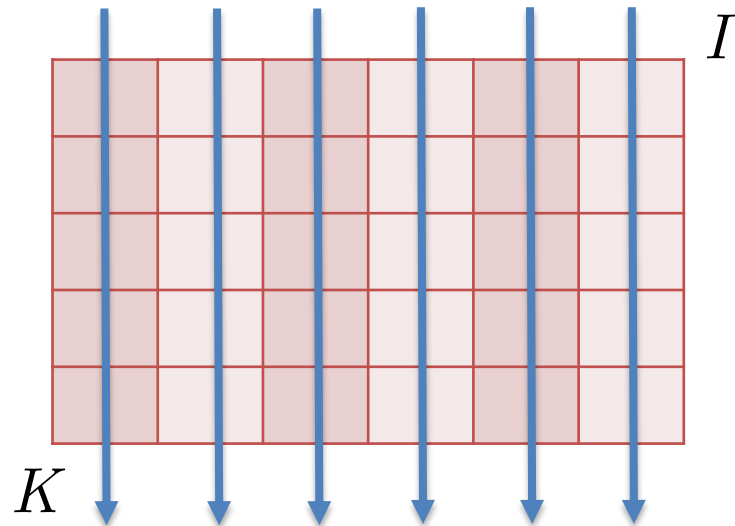
```
// Update  $q(U)$ 
parallel for  $i = 1, \dots, I$  do
  for  $k = 1, \dots, K$  do
     $\xi \leftarrow \bar{u}_{ki}$ 
     $s_{ki}^u \leftarrow \left( \alpha_k + \tau \sum_{j \in \Omega_i} (\bar{v}_{kj}^2 + s_{kj}^v) \right)^{-1}$ 
     $\bar{u}_{ki} \leftarrow s_{ki}^u \left( \tau \sum_{j \in \Omega_i} (R_{ij} + \bar{u}_{ki} \bar{v}_{kj}) \bar{v}_{kj} \right)$ 
    for  $j \in \Omega_i$  do
       $R_{ij} \leftarrow R_{ij} - (\bar{u}_{ki} - \xi) \bar{v}_{kj}$ 
    end for
  end for
end parallel for
```

```
// Update  $q(V)$ 
parallel for  $j = 1, \dots, J$  do
  for  $k = 1, \dots, K$  do
     $\xi \leftarrow \bar{v}_{kj}$ 
     $s_{kj}^v \leftarrow \left( \beta_k + \tau \sum_{i \in \Omega_j} (\bar{u}_{ki}^2 + s_{ki}^u) \right)^{-1}$ 
     $\bar{v}_{kj} \leftarrow s_{kj}^v \left( \tau \sum_{i \in \Omega_j} (R_{ij} + \bar{u}_{ki} \bar{v}_{kj}) \bar{u}_{ki} \right)$ 
    for  $i \in \Omega_j$  do
       $R_{ij} \leftarrow R_{ij} - \bar{u}_{ki} (\bar{v}_{kj} - \xi)$ 
    end for
  end for
end parallel for
```

$$O \left( \sum_{k=1}^K \sum_{i=1}^I 3|\Omega_i| + \sum_{k=1}^K \sum_{j=1}^J 3|\Omega_j| \right) = O(6|\Omega|K)$$

# Scalable VBMF: parallelization

```
// Update  $q(\mathbf{U})$ 
parallel for  $i = 1, \dots, I$  do
  for  $k = 1, \dots, K$  do
     $\xi \leftarrow \bar{u}_{ki}$ 
     $s_{ki}^u \leftarrow \left( \alpha_k + \tau \sum_{j \in \Omega_i} (\bar{v}_{kj}^2 + s_{kj}^v) \right)^{-1}$ 
     $\bar{u}_{ki} \leftarrow s_{ki}^u \left( \tau \sum_{j \in \Omega_i} (R_{ij} + \bar{u}_{ki} \bar{v}_{kj}) \bar{v}_{kj} \right)$ 
    for  $j \in \Omega_i$  do
       $R_{ij} \leftarrow R_{ij} - (\bar{u}_{ki} - \xi) \bar{v}_{kj}$ 
    end for
  end for
end parallel for
```



- Each column of variational parameters can be updated independently from the updates of other columns.
- Parallelization can be easily done in a column-by-column manner.
- Easy implementation with the OpenMP library on multi-core system.

## Related work (Pilásy et al., ReSys 2010)

- Similar idea is used to reduce the cubic time complexity of ALS to linear one.

$$\sum_{(i,j) \in \Omega} [(X_{ij} - \mathbf{u}_i^\top \mathbf{v}_j)^2 + \lambda(\|\mathbf{u}_i\|^2 + \|\mathbf{v}_j\|^2)]$$



$$\mathcal{F}(q) = \sum_{(i,j) \in \Omega} \mathcal{F}_{ij} + \sum_{k=1}^K \sum_{i=1}^I \mathcal{F}_{ki}^u + \sum_{k=1}^K \sum_{j=1}^J \mathcal{F}_{kj}^v$$

where

$$\mathcal{F}_{ij} = \mathbb{E}_q \{ \log p(X_{ij} | \mathbf{U}, \mathbf{V}) \} = -\frac{\tau}{2} \tilde{\mathcal{E}}_{ij} - \frac{1}{2} \log(2\pi\tau^{-1}),$$

$$\tilde{\mathcal{E}}_{ij} = (X_{ij} - \sum_{k=1}^K \bar{u}_{ki} \bar{v}_{kj})^2 + \sum_{k=1}^K (\bar{u}_{ki}^2 s_{kj}^v + \bar{v}_{kj}^2 s_{ki}^u + s_{ki}^u s_{kj}^v),$$

$$\begin{aligned} \mathcal{F}_{ki}^u &= \mathbb{E}_q \{ \log p(u_{ki}) - \log q(u_{ki}) \} \\ &= -\frac{\alpha_k}{2} (\bar{u}_{ki}^2 + s_{ki}^u) + \frac{1}{2} \log(s_{ki}^u \alpha_k) + \frac{1}{2}, \end{aligned}$$

$$\begin{aligned} \mathcal{F}_{kj}^v &= \mathbb{E}_q \{ \log p(v_{kj}) - \log q(v_{kj}) \} \\ &= -\frac{\beta_k}{2} (\bar{v}_{kj}^2 + s_{kj}^v) + \frac{1}{2} \log(s_{kj}^v \beta_k) + \frac{1}{2}. \end{aligned}$$

RMF

$$O(4|\Omega|K)$$

Scalable VBMMF

$$O(6|\Omega|K)$$

With small extra effort, more accurate model is obtainable without tuning of regularization parameter

## Related Work (Raiko et al., ECML 2007)

- Consider element-wisely factorized variational distribution

$$q(\mathbf{U}, \mathbf{V}) = \prod_{k=1}^K \prod_{i=1}^I \mathcal{N}(u_{ki} | \bar{u}_{ki}, s_{ki}^u) \prod_{k=1}^K \prod_{j=1}^J \mathcal{N}(v_{kj} | \bar{v}_{kj}, s_{kj}^v)$$

- Update  $\mathbf{U}$  and  $\mathbf{V}$  by scaled gradient descent method

$$\bar{u}_{ki} \leftarrow \bar{u}_{ki} - \eta \left( \frac{\partial^2 \mathcal{C}}{\partial^2 \bar{u}_{ki}} \right)^{-\gamma} \frac{\partial \mathcal{C}}{\partial \bar{u}_{ki}}$$

$$\bar{v}_{kj} \leftarrow \bar{v}_{kj} - \eta \left( \frac{\partial^2 \mathcal{C}}{\partial^2 \bar{v}_{kj}} \right)^{-\gamma} \frac{\partial \mathcal{C}}{\partial \bar{v}_{kj}}$$

- Require tuning of learning rate
- Learning speed is slower than our algorithm

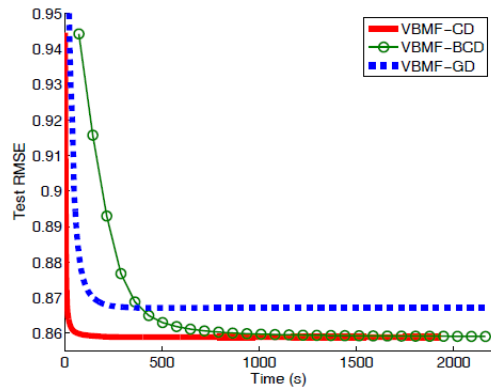
# Numerical Experiments

- Compare VBMF-CD, VBMF-BCD (Lim & The KDDcup 2007), VBMF-GD (Raiko et al., ECML 2007)
- Experimental environment
  - Quad-core Intel® core™ i7-3820 @ 3.6GHz
  - 64 GB memory
  - Implemented in Matlab 2011a, where main computational modules are implemented in C++ as mex files
  - Parallelized with the OpenMP library
- Datasets

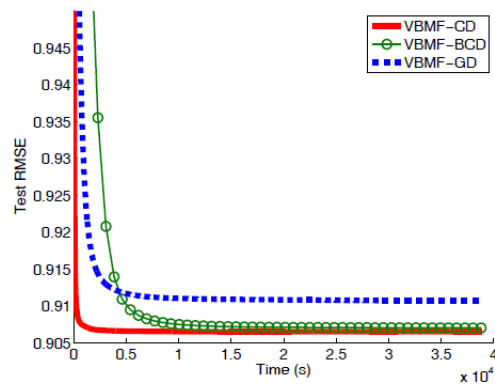
|             | MovieLens10M | Netflix     | Yahoo-music |
|-------------|--------------|-------------|-------------|
| # of user   | 69,878       | 480,189     | 1,000,990   |
| # of item   | 10,677       | 17,770      | 624,961     |
| # of rating | 10,000,054   | 100,480,507 | 262,810,275 |

# Numerical Experiments: $K = 20$

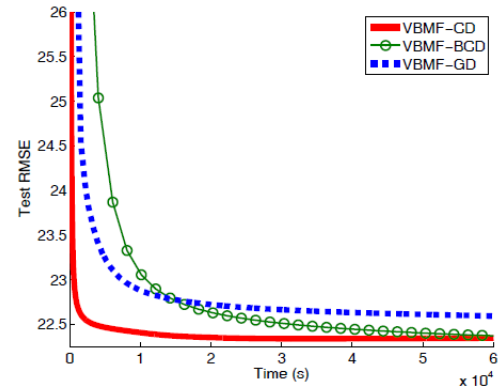
RMSE versus computation time on a quad-core system for each dataset:  
 (a) MovieLens10M, (b) Netflix, (c) Yahoo-music



(a)



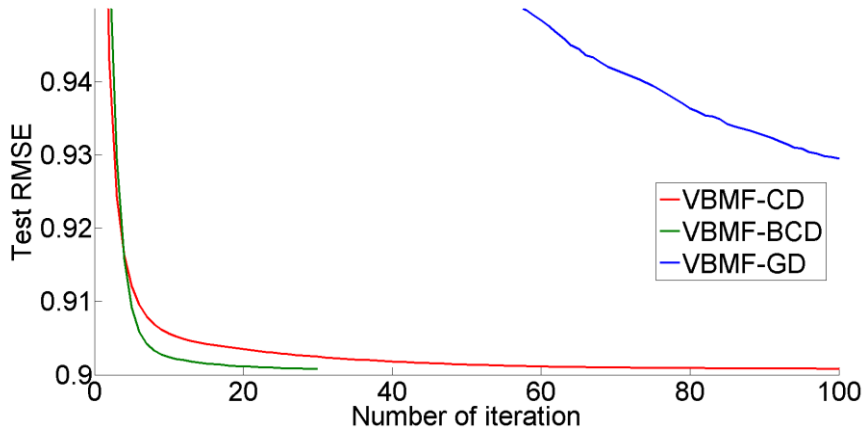
(b)



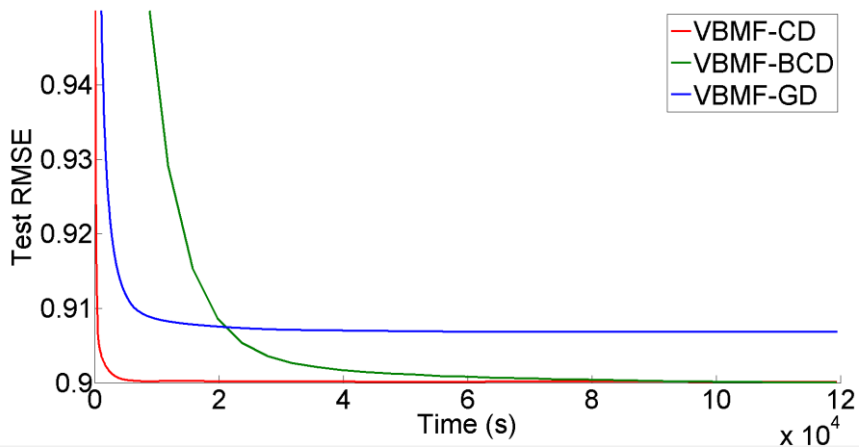
(c)

|          | MovieLens10M | Netflix | Yahoo-music |
|----------|--------------|---------|-------------|
| VBMF-CD  | 0.8589       | 0.9065  | 22.3425     |
| VBMF-BCD | 0.8671       | 0.9070  | 22.3671     |
| VBMF-GD  | 0.8591       | 0.9167  | 22.5883     |

# Numerical Experiments: Netflix, $K = 50$



|          | Time per iter. |
|----------|----------------|
| VBMF-BCD | 66 min.        |
| VBMF-CD  | 77 sec.        |
| VBMF-GD  | 29 sec.        |



| RMSE   | VBMF-BCD |      | VBMF-CD |      |
|--------|----------|------|---------|------|
|        | Iter.    | Time | Iter.   | Time |
| 0.9005 | 19       | 21 h | 63      | 74 m |
| 0.9004 | 21       | 23 h | 70      | 82 m |
| 0.9003 | 22       | 24 h | 84      | 98 m |
| 0.9002 | 25       | 28 h | 108     | 2 h  |
| 0.9001 | 27       | 31 h | 680     | 13 h |
| 0.9000 | 30       | 33 h |         |      |

# Conclusion

- We presented scalable learning algorithm for VBMF, VBMF-CD.
- VBMF-CD optimizes element-wisely factorized variational distributions with coordinate descent method.
- Space and time complexity of VBMF-CD are linear.
- VBMF-CD can be easily parallelized.
- Experimental results confirmed the user behavior of VBMF-CD such as scalability, fast learning, and prediction accuracy.