Scalable Bayesian Matrix Factorization

Avijit Saha[1], Rishabh Misra[2], Balaraman Ravindran[1]

[1]Department of CSE, Indian Institute of Technology Madras, India

[2]Department of CSE, Thapar University, India

Presented by - Ayan Acharya

Outline

- Motivation
- Introduction
- Model
- Inference
- Experiments
  - Datasets
  - Baseline
  - Experimental Setup
  - Results
- Conclusion and Future Work
- References

Motivation

- In many scientific domains matrix factorization (MF) [1] is ubiquitous.
- MF is a dimensionality reduction technique and is used to fill missing entries of a matrix.

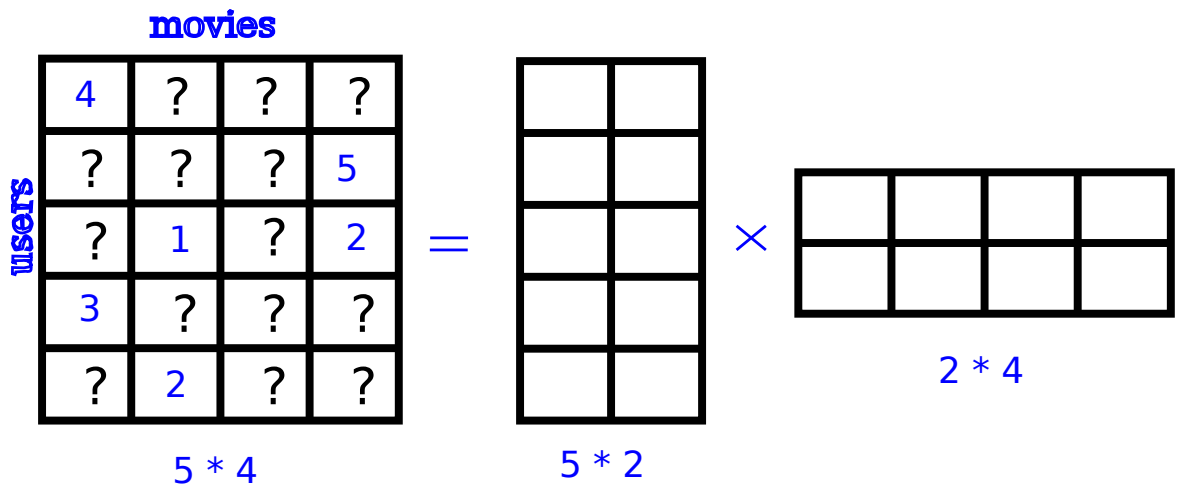

Figure 1: Example of MF.

Motivation

- One common approach to solve MF is using stochastic gradient descent (SGD) [1].
- SGD is scalable and enjoys local convergence guarantee [2].
- However, it often overfits the data and requires manual tuning of the learning rate and regularization parameters.
- Fully Bayesian methods for MF such as Bayesian Probabilistic Matrix Factorization (BPMF) [3] avoids these problems and provides state of the art performance.
- However, BPMF has cubic time complexity with respect to the target rank.
- To alleviate this problem of BPMF, we propose the Scalable Bayesian Matrix Factorization (SBMF).

Introduction

- MF is the simplest factor based model.
- Consider a user-movie matrix $\boldsymbol{R} \in \mathbb{R}^{I \times J}$ where the r_{ij} cell represents the rating provided to the j^{th} movie by the i^{th} user. MF decomposes the matrix $\boldsymbol{R}$ into two low-rank matrices $\boldsymbol{U} = [\boldsymbol{u}_1, \boldsymbol{u}_2, ..., \boldsymbol{u}_I]^T \in \mathbb{R}^{I \times K}$ and $\boldsymbol{V} = [\boldsymbol{v}_1, \boldsymbol{v}_2, ..., \boldsymbol{v}_J]^T \in \mathbb{R}^{J \times K}$:

$$\boldsymbol{R} \sim \boldsymbol{U}\boldsymbol{V}^T. \tag{1}$$

- Probabilistic Matrix Factorization (PMF) [4] provides a probabilistic interpretation for MF. PMF considers the likelihood term as:

$$p(\boldsymbol{R}|\boldsymbol{U}, \boldsymbol{V}, \tau^{-1}) = \prod_{(i,j)\in\Omega} \mathcal{N}(r_{ij}|\boldsymbol{u}_i^T \boldsymbol{v}_j, \tau^{-1}), \tag{2}$$

- In PMF, inferring the posterior distribution is intractable.
- PMF handles this intractability by providing a maximum a posteriori estimation, which is equivalent to minimizing the regularized square error:

$$\sum_{(i,j)\in\Omega} \left(r_{ij} - \boldsymbol{u}_i^T \boldsymbol{v}_j\right)^2 + \lambda \left(||\boldsymbol{U}||_F^2 + ||\boldsymbol{V}||_F^2\right) \tag{3}$$

Introduction

- The optimization problem in Eq. (3) is solved using SGD.
- Hence, maximum a posteriori estimation of PMF suffers from the same problems of SGD.
- On the other hand, fully Bayesain method BPMF directly approximates the posterior distribution using the Gibbs sampling technique and provides state of the art performance.

Introduction

- However, BPMF has cubic time complexity with respect to the target rank due to it's multivariate Gaussian assumption on priors.
- Hence, we develop the SBMF which uses Gibbs sampling as inference mechanism.
- SBMF has linear time complexity with respect to the target rank and linear space complexity with respect to the number of non-zero observations, as we assume univariate Gaussian prior.

Model

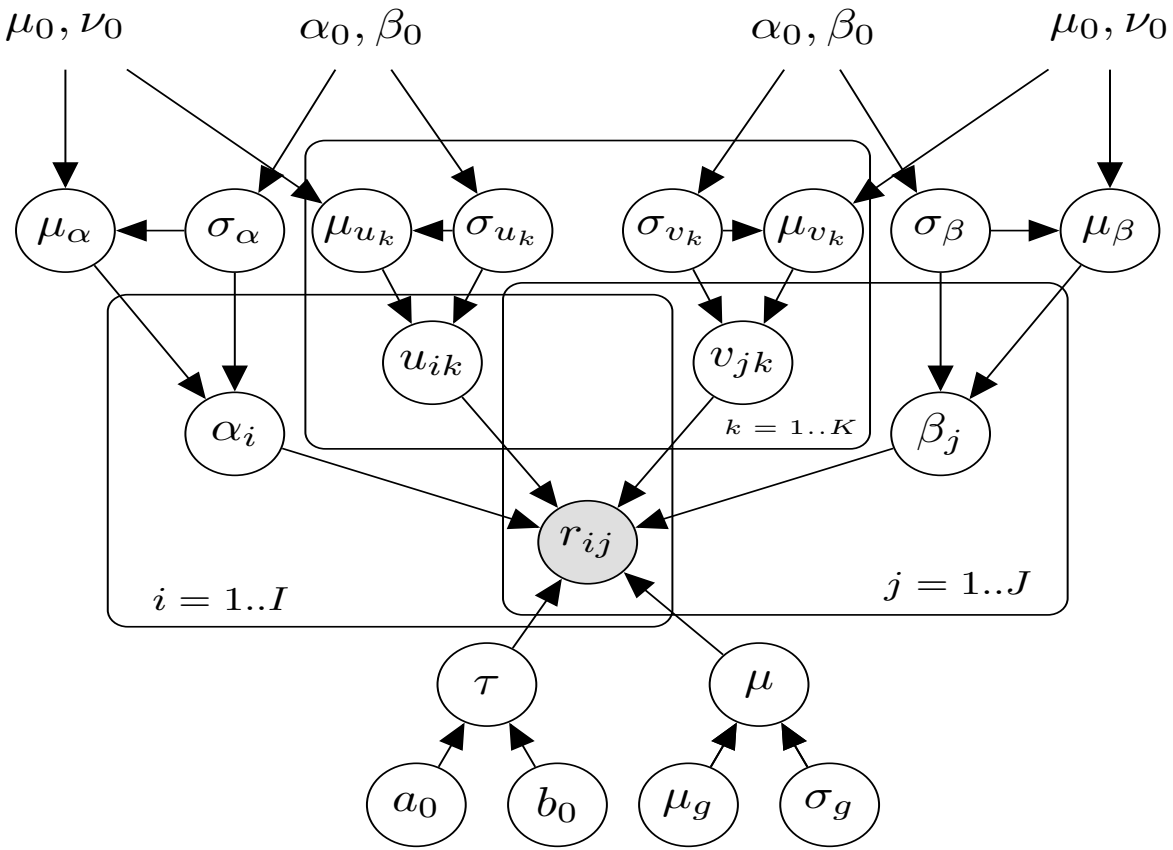

Figure 2: Graphical model for SBMF.

Model

The likelihood term of SBMF is as follows:

$$p(\boldsymbol{R}|\boldsymbol{\Theta}) = \prod_{(i,j)\in\Omega} \mathcal{N}(r_{ij}|\mu + \alpha_i + \beta_j + \boldsymbol{u}_i^T \boldsymbol{v}_j, \tau^{-1}), \qquad (4)$$

where μ is the global bias,
α_i is the bias for the i^{th} user,
β_j is the bias for the j^{th} item,
$\boldsymbol{u}_i$ is the latent factor vector for the i^{th} user,
$\boldsymbol{v}_j$ is the latent factor vector for the j^{th} item,
τ is the model precision,
Ω is the set of all observations, and
$\boldsymbol{\Theta}$ is the set of all the model parameters.

We place independent univariate priors on all the model parameters in $\boldsymbol{\Theta}$ as:

$$p(\mu) = \mathcal{N}(\mu|\mu_g, \sigma_g^{-1}), \tag{5}$$

$$p(\alpha_i) = \mathcal{N}(\alpha_i|\mu_\alpha, \sigma_\alpha^{-1}), \tag{6}$$

$$p(\beta_j) = \mathcal{N}(\beta_j|\mu_\beta, \sigma_\beta^{-1}), \tag{7}$$

$$p(\boldsymbol{U}) = \prod_{i=1}^{I}\prod_{k=1}^{K} \mathcal{N}(u_{ik}|\mu_{u_k}, \sigma_{u_k}^{-1}), \tag{8}$$

$$p(\boldsymbol{V}) = \prod_{j=1}^{J}\prod_{k=1}^{K} \mathcal{N}(v_{jk}|\mu_{v_k}, \sigma_{v_k}^{-1}), \tag{9}$$

$$p(\tau) = \mathcal{N}(\tau|a_0, b_0). \tag{10}$$

We further place Normal-Gamma priors on all the hyperparameters $\boldsymbol{\Theta}_H = \{\mu_\alpha, \sigma_\alpha, \mu_\beta, \sigma_\beta, \{\mu_{u_k}, \sigma_{u_k}\}, \{\mu_{v_k}, \sigma_{v_k}\}\}$ as:

$$
\begin{aligned}
p(\mu_\alpha, \sigma_\alpha) &= \mathcal{NG}\left(\mu_\alpha, \sigma_\alpha | \mu_0, \nu_0, \alpha_0, \beta_0\right), && (11)\\
p(\mu_\beta, \sigma_\beta) &= \mathcal{NG}\left(\mu_\beta, \sigma_\beta | \mu_0, \nu_0, \alpha_0, \beta_0\right), && (12)\\
p(\mu_{u_k}, \sigma_{u_k}) &= \mathcal{NG}\left(\mu_{u_k}, \sigma_{u_k} | \mu_0, \nu_0, \alpha_0, \beta_0\right), && (13)\\
p(\mu_{v_k}, \sigma_{v_k}) &= \mathcal{NG}\left(\mu_{v_k}, \sigma_{v_k} | \mu_0, \nu_0, \alpha_0, \beta_0\right). && (14)
\end{aligned}
$$

The joint distribution of the observations and the hidden variables can be written as:

$$p(\boldsymbol{R}, \boldsymbol{\Theta}, \boldsymbol{\Theta}_H | \boldsymbol{\Theta}_0) = p(\boldsymbol{R}|\boldsymbol{\Theta}) p(\mu) \prod_{i=1}^{I} p(\alpha_i) \prod_{j=1}^{J} p(\beta_j) p(\boldsymbol{U}) p(\boldsymbol{V}) \\ p(\mu_\alpha, \sigma_\alpha) p(\mu_\beta, \sigma_\beta) \prod_{k=1}^{K} p(\mu_{u_k}, \sigma_{u_k}) p(\mu_{v_k}, \sigma_{v_k}), \tag{15}$$

where $\boldsymbol{\Theta}_0 = \{a_0, b_0, \mu_g, \sigma_g, \mu_0, \nu_0, \alpha_0, \beta_0\}$

Inference

- Evaluation of the joint distribution in Eq. (15) is intractable.
- However, all the model parameters are conditionally conjugate [5].
- So we develop a Gibbs sampler with closed form updates.
- Replacing Eq. (4)-(14) in Eq. (15), the sampling distribution of u_{ik} can be written as follows:

$$p(u_{ik}|-) \sim \mathcal{N}\left(u_{ik}|\mu^*, \sigma^*\right), \tag{16}$$

$$\sigma^* = \left(\sigma_{u_k} + \tau \sum_{j \in \Omega_i} v_{jk}^2\right)^{-1} \tag{17}$$

$$\mu^* = \sigma^* \left(\sigma_{u_k}\mu_{u_k} + \tau \sum_{j \in \Omega_i} v_{jk}\left(r_{ij} - \left(\mu + \alpha_i + \beta_j + \sum_{l=1 \& l \neq k}^{K} u_{il}v_{jl}\right)\right)\right). \tag{18}$$

Inference

- Now, directly sampling u_{ik} from Eq. (16) requires $O(K|\Omega_i|)$ complexity.
- However, precomputing $e_{ij} = r_{ij} - (\mu + \alpha_i + \beta_j + u_i^T v_j)$ for all $(i, j) \in \Omega$ reduces the complexity to $O(|\Omega_i|)$.
- We sample model parameters in parallel using the Gibbs sampling equations.

Table 1: Complexity comparison.

Method	Time Complexity	Space Complexity
SBMF	$O(\|\Omega\|K)$	$O((I+J)K)$
BPMF	$O(\|\Omega\|K^2 + (I+J)K^3)$	$O((I+J)K)$

Datasets

Table 2: Dataset description.

Dataset	No. of users	No. of movies	No. of ratings
Movielens 10m	71567	10681	**10m**
Movielens 20m	138493	27278	**20m**
Netflix	480189	17770	**100m**

Baseline

- We compare SBMF with BPMF.
- Parallel implementation for BPMF is not available.
- Hence, we compare serial version of BPMF with both serial and parallel implementation of SBMF, and call them SBMF-S and SBMF-P, respectively.

Experimental Setup

- We use Intel core i5 machines with 16GB RAM.
- BPMF is initialized with standard parameter setting.
- We use 50 burn-in iterations followed by 100 collection iterations for all the experiments.

Results

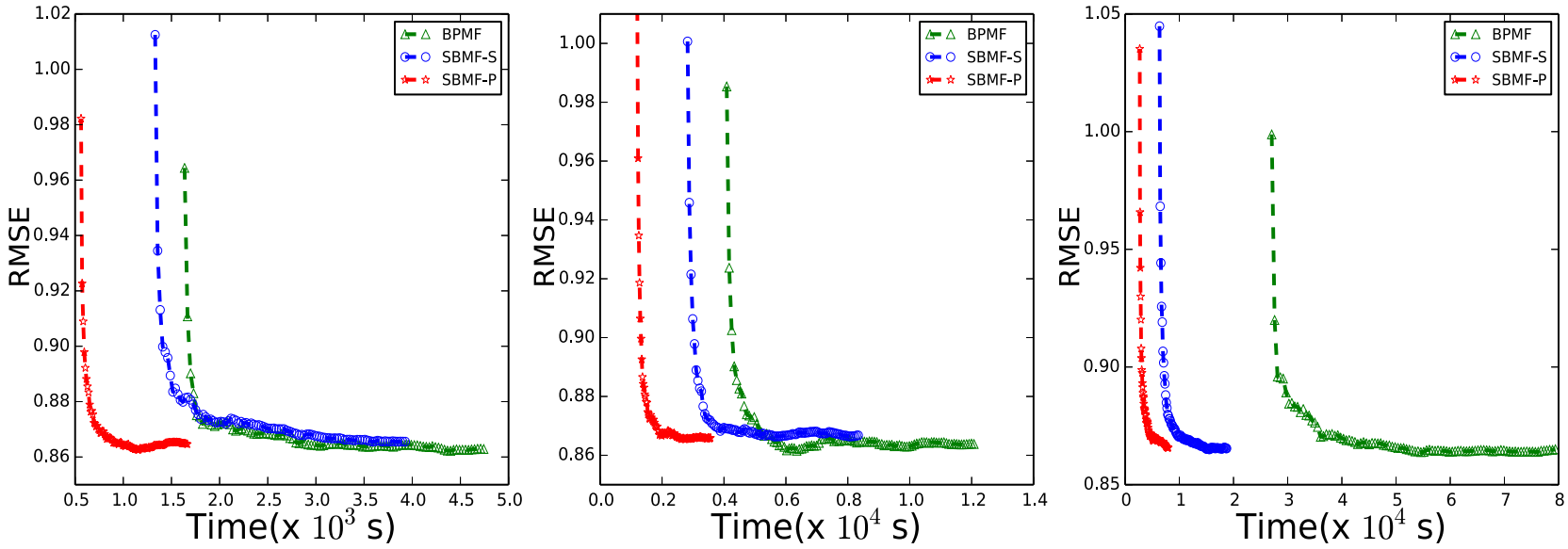

Figure 3: Graphs from left to right show results on Movielens 10M dataset for $K = 50, 100$, and 200.

Results

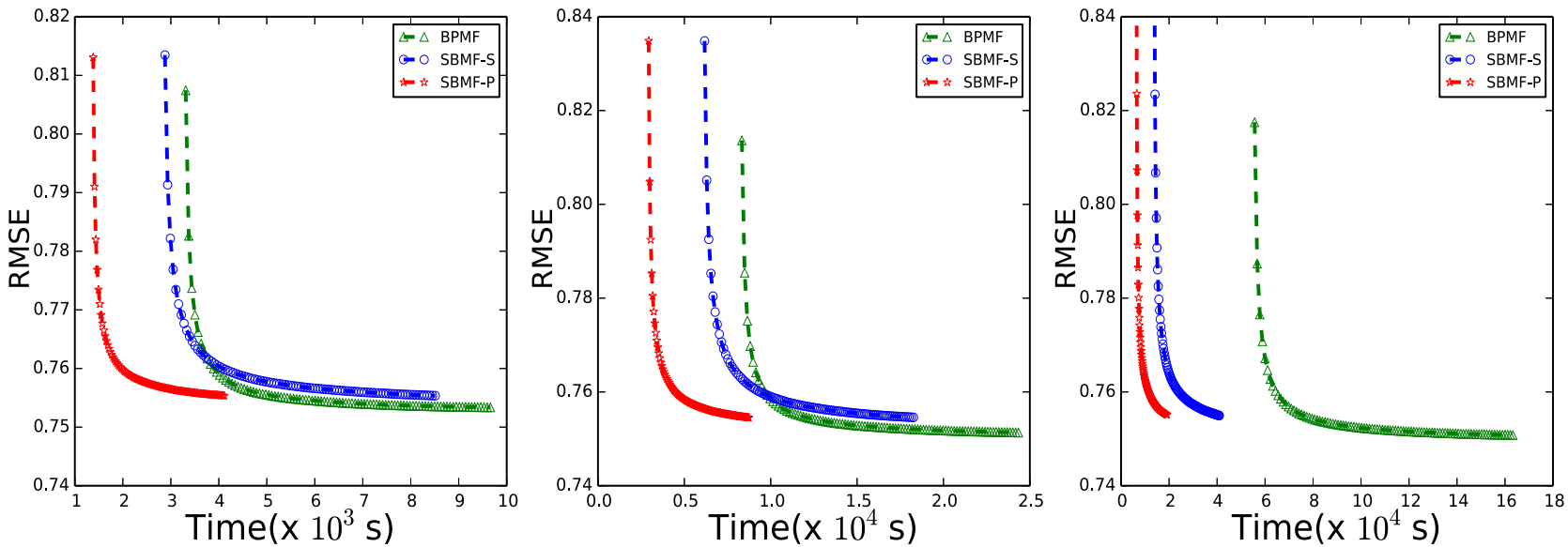

Figure 4: Graphs from left to right show results on Movielens 20M dataset for $K = 50, 100$, and 200.

Results

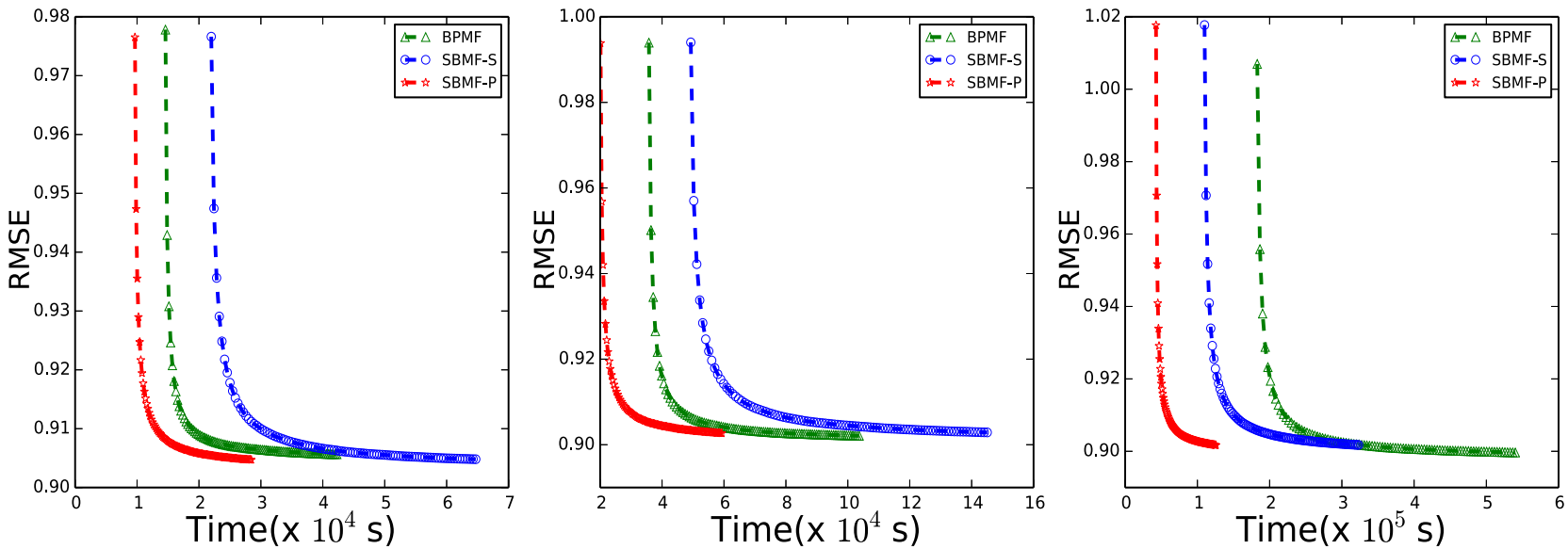

Figure 5: Graphs from left to right show results on Netflix dataset for $K = 50, 100,$ and 200.

Results

- SBMF-P takes much less time than BPMF.
- Similar trend exist for SBMF-S (except for $K = \{50, 100\}$ in Netflix dataset).
- We believe that in Netflix dataset for $K = \{50, 100\}$, BPMF takes less time than SBMF-S, because BPMF is implemented in Matlab where matrix operations are efficient, whereas SBMF uses unoptimized C++ code. The problem is compounded in Netflix dataset as large number of observations need more matrix operations.

Results

Table 3 shows detailed results comparison.

Table 3: Results Comparison.

		$K = 50$		$K = 100$		$K = 200$	
Dataset	Method	RMSE	Time(Hr)	RMSE	Time(Hr)	RMSE	Time(Hr)
Movielens 10m	BPMF	**0.8629**	1.317	**0.8638**	3.517	**0.8651**	22.058
	SBMF-S	0.8655	1.091	0.8667	2.316	0.8654	5.205
	SBMF-P	0.8646	**0.462**	0.8659	**0.990**	0.8657	**2.214**
Movielens 20m	BPMF	**0.7534**	2.683	**0.7513**	6.761	**0.7508**	45.355
	SBMF-S	0.7553	2.364	0.7545	5.073	0.7549	11.378
	SBMF-P	0.7553	**1.142**	0.7545	**2.427**	0.7551	**5.321**
Netflix	BPMF	0.9057	11.739	**0.9021**	28.797	**0.8997**	150.026
	SBMF-S	0.9048	17.973	0.9028	40.287	0.9017	89.809
	SBMF-P	**0.9047**	**7.902**	0.9026	**16.477**	0.9017	**34.934**

Results

- For $K = 200$, SBMF provides significant speed up.
- Total time difference between both of the variants of SBMF and BPMF increases with the dimension of latent factor vector.
- Both SBMF-P and SBMF-S suffer small loss in the performance compare to BPMF.
- Increasing the latent space dimension reduces the RMSE value in the Netflix dataset.

Conclusion and Future Work

- We have developed the SBMF.
- SBMF has linear time and space complexity.
- SBMF gives competitive performance in less time as compared to BPMF, as validated empirically.
- BPMF is preferred with lower latent space dimension, but for higher latent space dimension SBMF should be used.
- Future work is to optimize the code of SBMF and apply SBMF with side-information.

References


Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, vol. 42, pp. 30–37, Aug. 2009.


M.-A. Sato, "Online model selection based on the variational Bayes," *Neural Computation*, vol. 13, no. 7, pp. 1649–1681, 2001.


R. Salakhutdinov and A. Mnih, "Bayesian probabilistic matrix factorization using markov chain monte carlo," in *Proc. of ICML*, pp. 880–887, 2008.


R. Salakhutdinov and A. Mnih, "Probabilistic matrix factorization," in *Proc. of NIPS*, 2007.


M. Hoffman, D. Blei, C. Wang, and J. Paisley, "Stochastic variational inference," *JMLR*, vol. 14, pp. 1303–1347, may 2013.

Questions

Thanks!