Bayesian Probabilistic Matrix Factorization using Markov Chain Monte Carlo

Ruslan Salakhutdinov RSALAKHU@CS.TORONTO.EDU
Andriy Mnih AMNIH@CS.TORONTO.EDU
Department of Computer Science, University of Toronto, Toronto, Ontario M5S 3G4, Canada

Abstract

Low-rank matrix approximation methods provide one of the simplest and most effective approaches to collaborative filtering. Such models are usually fitted to data by finding a MAP estimate of the model parameters, a procedure that can be performed efficiently even on very large datasets. However, unless the regularization parameters are tuned carefully, this approach is prone to overfitting because it finds a single point estimate of the parameters. In this paper we present a fully Bayesian treatment of the Probabilistic Matrix Factorization (PMF) model in which model capacity is controlled automatically by integrating over all model parameters and hyperparameters. We show that Bayesian PMF models can be efficiently trained using Markov chain Monte Carlo methods by applying them to the Netflix dataset, which consists of over 100 million movie ratings. The resulting models achieve significantly higher prediction accuracy than PMF models trained using MAP estimation.

1. Introduction

Factor-based models have been used extensively in the domain of collaborative filtering for modelling user preferences. The idea behind such models is that preferences of a user are determined by a small number of unobserved factors. In a linear factor model, a user's rating of an item is modelled by the inner product of an item factor vector and a user factor vector. This means that the $N \times M$ preference matrix of ratings that N users assign to M movies is modeled by the product of an $D \times N$ user coefficient matrix U and a $D \times M$ factor matrix V (Rennie & Srebro, 2005; Srebro & Jaakkola, 2003). Training such a model amounts to finding the best rank-D approximation to the observed $N \times M$ target matrix R under the given loss function.

A variety of probabilistic factor-based models have been proposed (Hofmann, 1999; Marlin, 2004; Marlin & Zemel, 2004; Salakhutdinov & Mnih, 2008). In these models factor variables are assumed to be marginally independent while rating variables are assumed to be conditionally independent given the factor variables. The main drawback of such models is that inferring the posterior distribution over the factors given the ratings is intractable. Many of the existing methods resort to performing MAP estimation of the model parameters. Training such models amounts to maximizing the log-posterior over model parameters and can be done very efficiently even on very large datasets.

In practice, we are usually interested in predicting ratings for new user/movie pairs rather than in estimating model parameters. This view suggests taking a Bayesian approach to the problem which involves integrating out the model parameters. In this paper, we describe a fully Bayesian treatment of the Probabilistic Matrix Factorization (PMF) model which has been recently applied to collaborative filtering (Salakhutdinov & Mnih, 2008). The distinguishing feature of our work is the use of Markov chain Monte Carlo (MCMC) methods for approximate inference in this model. In practice, MCMC methods are rarely used on large-scale problems because they are perceived to be very slow by practitioners. In this paper we show that MCMC can be successfully applied to the large, sparse, and very imbalanced Netflix dataset, containing over 100 million user/movie ratings. We also show that it significantly increases the model's predictive accuracy, especially for the infrequent users, compared to the standard PMF models trained using MAP with regularization parameters that have been carefully tuned on the validation set.

Previous applications of Bayesian matrix factorization methods to collaborative filtering (Lim & Teh, 2007; Raiko et al., 2007) have used variational approxima-

Appearing in *Proceedings of the 25th International Conference on Machine Learning*, Helsinki, Finland, 2008. Copyright 2008 by the author(s)/owner(s).

tions for performing inference. These methods attempt to approximate the true posterior distribution by a simpler, factorized distribution under which the user factor vectors are independent of the movie factor vectors. The consequence of this assumption is that that the variational posterior distributions over the factor vectors is a product of two multivariate Gaussians: one for the viewer factor vectors and one for the movie factor vectors. This assumption of independence between the viewer and movie factors seems unreasonable, and, as our experiments demonstrate, the distributions over factors in such models turn out to be non-Gaussian. This conclusion is supported by the fact that the Bayesian PMF models outperform their MAP trained counterparts by a much larger margin than the variationally trained models do.

2. Probabilistic Matrix Factorization

Probabilistic Matrix Factorization (PMF) is a probabilistic linear model with Gaussian observation noise (see Fig. 1, left panel). Suppose we have N users and M movies. Let R_{ij} be the rating value of user i for movie j, U_i and V_j represent D-dimensional user-specific and movie-specific latent feature vectors respectively. The conditional distribution over the observed ratings $R \in \mathbb{R}^{N\times M}$ (the likelihood term) and the prior distributions over $U \in \mathbb{R}^{D\times N}$ and $V \in \mathbb{R}^{D\times M}$ are given by:

$$
\begin{aligned}
p(R|U,V,\alpha) &= \prod_{i=1}^{N}\prod_{j=1}^{M}\left[\mathcal{N}(R_{ij}|U_i^T V_j, \alpha^{-1})\right]^{I_{ij}} && (1)\\
p(U|\alpha_U) &= \prod_{i=1}^{N}\mathcal{N}(U_i|0, \alpha_U^{-1} I) && (2)\\
p(V|\alpha_V) &= \prod_{j=1}^{M}\mathcal{N}(V_j|0, \alpha_V^{-1} I), && (3)
\end{aligned}
$$

where $\mathcal{N}(x|\mu,\alpha^{-1})$ denotes the Gaussian distribution with mean μ and precision α, and I_{ij} is the indicator variable that is equal to 1 if user i rated movie j and equal to 0 otherwise.

Learning in this model is performed by maximizing the log-posterior over the movie and user features with fixed hyperparameters (i.e. the observation noise variance and prior variances):

$$
\begin{aligned}
\ln p(U,V|R,\alpha,\alpha_V,\alpha_U) &= \ln p(R|U,V,\alpha) + \\
&+ \ln p(U|\alpha_U) + \ln p(V|\alpha_V) + C,
\end{aligned}
$$

where C is a constant that does not depend on the parameters. Maximizing this posterior distribution with respect to U and V is equivalent to minimizing the sum-of-squares error function with quadratic regularization terms:

$$
\begin{aligned}
E &= \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{M} I_{ij}\big(R_{ij} - U_i^T V_j\big)^2 \\
&+ \frac{\lambda_U}{2}\sum_{i=1}^{N} \| U_i \|_{\text{Fro}}^2 + \frac{\lambda_V}{2}\sum_{j=1}^{M} \| V_j \|_{\text{Fro}}^2, \quad (4)
\end{aligned}
$$

where $\lambda_U = \alpha_U/\alpha$, $\lambda_V = \alpha_V/\alpha$, and $\| \cdot \|_{\text{Fro}}^2$ denotes the Frobenius norm. A local minimum of the objective function given by Eq. 4 can be found by performing gradient descent in U and V.

The main drawback of this training procedure is the need for manual complexity control that is essential to making the model generalize well, particularly on sparse and imbalanced datasets. One way to control the model complexity is to search for appropriate values of regularization parameters λ_U and λ_V defined above. We could, for example, consider a set of reasonable parameter values, train a model for each setting of the parameters, and choose the model that performs best on the validation set. This approach however is computationally very expensive, since it requires training a multitude of models instead of training a single one.

Alternatively, we could introduce priors for the hyperparameters and maximize the log-posterior of the model over *both parameters and hyperparameters*, which allows model complexity to be controlled automatically based on the training data (Nowlan & Hinton, 1992; Salakhutdinov & Mnih, 2008). Though this approach has been shown to work in practice it is not well-grounded theoretically, and it is not difficult to construct a simple example for which such joint optimization would not produce the desired results.

In the next section we describe a fully Bayesian treatment of the PMF model with model parameters and hyperparameters integrated out using MCMC methods, which provides fully automatic complexity control.

3. Bayesian Probabilistic Matrix Factorization

3.1. The Model

The graphical model representing Bayesian PMF is shown in Fig. 1 (right panel). As in PMF, the likelihood of the observed ratings is given by Eq. 1. The prior distributions over the user and movie feature vec-

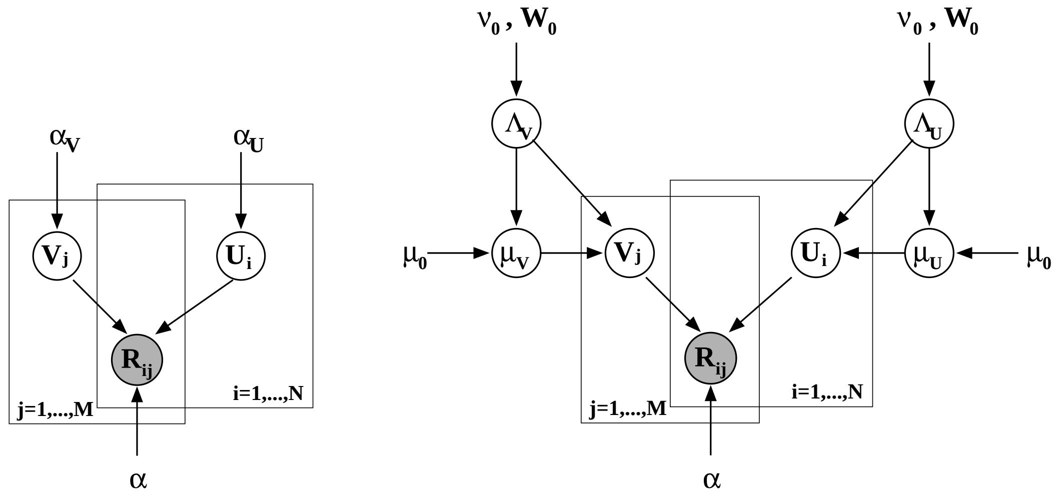

Figure 1. The left panel shows the graphical model for Probabilistic Matrix Factorization (PMF). The right panel shows the graphical model for Bayesian PMF.

tors are assumed to be Gaussian:

$$p(U|\mu_U, \Lambda_U) = \prod_{i=1}^{N} \mathcal{N}(U_i|\mu_U, \Lambda_U^{-1}), \qquad (5)$$

$$p(V|\mu_V, \Lambda_V) = \prod_{i=1}^{M} \mathcal{N}(V_i|\mu_V, \Lambda_V^{-1}). \qquad (6)$$

We further place Gaussian-Wishart priors on the user and movie hyperparameters $\Theta_U = \{\mu_U, \Lambda_U\}$ and $\Theta_V = \{\mu_V, \Lambda_V\}$:

$$\begin{aligned} p(\Theta_U|\Theta_0) &= p(\mu_U|\Lambda_U)p(\Lambda_U) \\ &= \mathcal{N}(\mu_U|\mu_0, (\beta_0\Lambda_U)^{-1})\mathcal{W}(\Lambda_U|W_0, \nu_0), \qquad (7) \end{aligned}$$

$$\begin{aligned} p(\Theta_V|\Theta_0) &= p(\mu_V|\Lambda_V)p(\Lambda_V) \\ &= \mathcal{N}(\mu_V|\mu_0, (\beta_0\Lambda_V)^{-1})\mathcal{W}(\Lambda_V|W_0, \nu_0). \qquad (8) \end{aligned}$$

Here $\mathcal{W}$ is the Wishart distribution with ν_0 degrees of freedom and a $D \times D$ scale matrix W_0:

$$\mathcal{W}(\Lambda|W_0, \nu_0) = \frac{1}{C}|\Lambda|^{(\nu_0-D-1)/2} \exp\left(-\frac{1}{2}\mathrm{Tr}(W_0^{-1}\Lambda)\right),$$

where C is the normalizing constant. For convenience we also define $\Theta_0 = \{\mu_0, \nu_0, W_0\}$. In our experiments we also set $\nu_0 = D$ and W_0 to the identity matrix for both user and movie hyperparameters and choose $\mu_0 = 0$ by symmetry.

3.2. Predictions

The predictive distribution of the rating value R_{ij}^* for user i and query movie j is obtained by marginalizing over model parameters and hyperparameters:

$$\begin{aligned} p(R_{ij}^*|R, \Theta_0) = \iint & p(R_{ij}^*|U_i, V_j)p(U, V|R, \Theta_U, \Theta_V) \\ & p(\Theta_U, \Theta_V|\Theta_0)\mathrm{d}\{U, V\}\mathrm{d}\{\Theta_U, \Theta_V\}. \qquad (9) \end{aligned}$$

Since exact evaluation of this predictive distribution is analytically intractable due to the complexity of the posterior we need to resort to approximate inference.

One choice would be to use variational methods (Hinton & van Camp, 1993; Jordan et al., 1999) that provide deterministic approximation schemes for posteriors. In particular, we could approximate the true posterior $p(U, V, \Theta_U, \Theta_V|R)$ by a distribution that factors, with each factor having a specific parametric form such as a Gaussian distribution. This approximate posterior would allow us to approximate the integrals in Eq. 9. Variational methods have become the methodology of choice, since they typically scale well to large applications. However, they can produce inaccurate results because they tend to involve overly simple approximations to the posterior.

MCMC-based methods (Neal, 1993), on the other hand, use the Monte Carlo approximation to the predictive distribution of Eq. 9 given by:

$$p(R_{ij}^*|R, \Theta_0) \approx \frac{1}{K}\sum_{k=1}^{K} p(R_{ij}^*|U_i^{(k)}, V_j^{(k)}). \qquad (10)$$

The samples $\{U_i^{(k)}, V_j^{(k)}\}$ are generated by running a Markov chain whose stationary distribution is the posterior distribution over the model parameters and hyperparameters $\{U, V, \Theta_U, \Theta_V\}$. The advantage of the Monte Carlo-based methods is that asymptotically they produce exact results. In practice, however, MCMC methods are usually perceived to be so computationally demanding that their use is limited to small-scale problems.

3.3. Inference

One of the simplest MCMC algorithms is the Gibbs sampling algorithm, which cycles through the latent variables, sampling each one from its distribution conditional on the current values of all other variables. Gibbs sampling is typically used when these conditional distributions can be sampled from easily.

Due to the use of conjugate priors for the parameters and hyperparameters in the Bayesian PMF model, the conditional distributions derived from the posterior distribution are easy to sample from. In particular, the conditional distribution over the user feature vector U_i, conditioned on the movie features, observed user rating matrix R, and the values of the hyperparameters is Gaussian:

$$p(U_i|R, V, \Theta_U, \alpha) = \mathcal{N}\big(U_i|\mu_i^*, [\Lambda_i^*]^{-1}\big) \qquad (11)$$
$$\sim \prod_{j=1}^{M} \Big[\mathcal{N}(R_{ij}|U_i^T V_j, \alpha^{-1})\Big]^{I_{ij}} p(U_i|\mu_U, \Lambda_U),$$

where

$$\Lambda_i^* = \Lambda_U + \alpha \sum_{j=1}^{M} \big[V_j V_j^T\big]^{I_{ij}} \qquad (12)$$
$$\mu_i^* = [\Lambda_i^*]^{-1}\bigg(\alpha \sum_{j=1}^{M} \big[V_j R_{ij}\big]^{I_{ij}} + \Lambda_U \mu_U\bigg). \qquad (13)$$

Note that the conditional distribution over the user latent feature matrix U factorizes into the product of conditional distributions over the individual user feature vectors:

$$p(U|R, V, \Theta_U) = \prod_{i=1}^{N} p(U_i|R, V, \Theta_U).$$

Therefore we can easily speed up the sampler by sampling from these conditional distributions in parallel. The speedup could be substantial, particularly when the number of users is large.

The conditional distribution over the user hyperparameters conditioned on the user feature matrix U is given by the Gaussian-Wishart distribution:

$$p(\mu_U, \Lambda_U|U, \Theta_0) = \mathcal{N}(\mu_U|\mu_0^*, (\beta_0^*\Lambda_U)^{-1})\mathcal{W}(\Lambda_U|W_0^*, \nu_0^*), \qquad (14)$$

where

$$\mu_0^* = \frac{\beta_0\mu_0 + N\bar{U}}{\beta_0 + N}, \quad \beta_0^* = \beta_0 + N, \quad \nu_0^* = \nu_0 + N,$$
$$\big[W_0^*\big]^{-1} = W_0^{-1} + N\bar{S} + \frac{\beta_0 N}{\beta_0 + N}(\mu_0 - \bar{U})(\mu_0 - \bar{U})^T$$
$$\bar{U} = \frac{1}{N}\sum_{i=1}^{N} U_i \qquad \bar{S} = \frac{1}{N}\sum_{i=1}^{N} U_i U_i^T.$$

The conditional distributions over the movie feature vectors and the movie hyperparameters have exactly the same form. The Gibbs sampling algorithm then takes the following form:

Gibbs sampling for Bayesian PMF

```
1. Initialize model parameters $\{U^1, V^1\}$
2. For t=1,...,T
    • Sample the hyperparameters (Eq. 14):
          $\Theta_U^t \sim p(\Theta_U|U^t, \Theta_0)$
          $\Theta_V^t \sim p(\Theta_V|V^t, \Theta_0)$
    • For each $i = 1, ..., N$ sample user features in parallel (Eq. 11):
          $U_i^{t+1} \sim p(U_i|R, V^t, \Theta_U^t)$
    • For each $i = 1, ..., M$ sample movie features in parallel:
          $V_i^{t+1} \sim p(V_i|R, U^{t+1}, \Theta_V^t)$
```

4. Experimental Results

4.1. Description of the dataset

The data, collected by Netflix, represent the distribution of all ratings Netflix obtained between October, 1998 and December, 2005. The training data set consists of 100,480,507 ratings from 480,189 randomly-chosen, anonymous users on 17,770 movie titles. As part of the training data, Netflix also provides validation data, containing 1,408,395 ratings. In addition, Netflix also provides a test set containing 2,817,131 user/movie pairs with the ratings withheld. The pairs were selected from the most recent ratings from a subset of the users in the training data set. Performance is assessed by submitting predicted ratings to Netflix which then posts the root mean squared error (RMSE) on an unknown half of the test set. As a baseline, Netflix provided the test score of its own system trained on the same data, which is 0.9514.

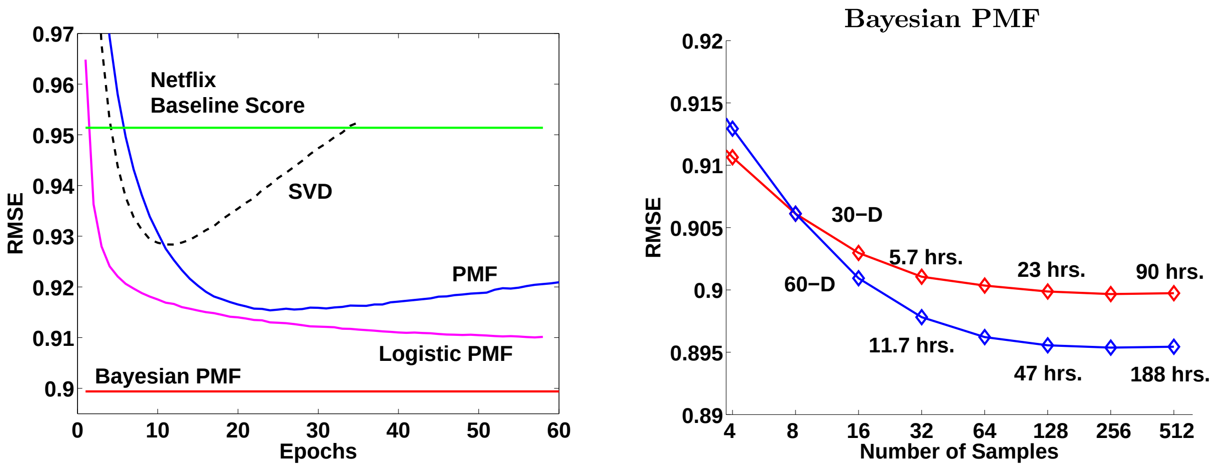

Figure 2. Left panel: Performance of SVD, PMF, logistic PMF, and Bayesian PMF using 30D feature vectors, on the Netflix validation data. The y-axis displays RMSE (root mean squared error), and the x-axis shows the number of epochs, or passes, through the entire training set. Right panel: RMSE for the Bayesian PMF models on the validation set as a function of the number of samples generated. The two curves are for the models with 30D and 60D feature vectors.

4.2. Training PMF models

For comparison, we have trained a variety of linear PMF models using MAP, choosing their regularization parameters using the validation set. In addition to linear PMF models, we also trained logistic PMF models, in which we pass the dot product between user- and movie-specific feature vectors through the logistic function $\sigma(x) = 1/(1 + \exp(-x))$ to bound the range of predictions:

$$p(R|U, V, \alpha) = \prod_{i=1}^{N}\prod_{j=1}^{M}\left[\mathcal{N}(R_{ij}|\sigma(U_i^T V_j), \alpha^{-1})\right]^{I_{ij}}. \quad (15)$$

The ratings $1, ..., 5$ are mapped to the interval $[0, 1]$ using the function $t(x) = (x - 1)/4$, so that the range of valid rating values matches the range of predictions our model can make. Logistic PMF models can sometimes provide slightly better results than their linear counterparts.

To speed up training, instead of performing full batch learning, we subdivided the Netflix data into mini-batches of size 100,000 (user/movie/rating triples) and updated the feature vectors after each mini-batch. We used a learning rate of 0.005 and a momentum of 0.9 for training the linear as well as logistic PMF models.

4.3. Training Bayesian PMF models

We initialized the Gibbs sampler by setting the model parameters U and V to their MAP estimates obtained by training a linear PMF model. We also set $\mu_0 = 0$, $\nu_0 = D$, and W_0 to the identity matrix, for both user and movie hyperpriors. The observation noise precision α was fixed at 2. The predictive distribution was computed using Eq. 10 by running the Gibbs sampler with samples $\{U_i^{(k)}, V_j^{(k)}\}$ collected after each full Gibbs step.

4.4. Results

In our first experiment, we compared a Bayesian PMF model to an SVD model, a linear PMF model, and a logistic PMF model, all using 30D feature vectors. The SVD model was trained to minimize the sum-squared distance to the observed entries of the target matrix, with no regularization applied to the feature vectors. Note that this model can be seen as a PMF model trained using maximum likelihood (ML). For the PMF models, the regularization parameters λ_U and λ_V were set to 0.002. Predictive performance of these models on the validation set is shown in Fig. 2 (left panel). The mean of the predictive distribution of the Bayesian PMF model achieves an RMSE of 0.8994, compared to an RMSE of 0.9174 of a moderately regularized linear PMF model, an improvement of over 1.7%.

The logistic PMF model does slightly outperform its linear counterpart, achieving an RMSE of 0.9097. However, its performance is still considerably worse than that of the Bayesian PMF model. A simple SVD achieves an RMSE of about 0.9280 and after about 10 epochs begins to overfit heavily. This experiment clearly demonstrates that SVD and MAP-trained PMF models can overfit and that the predictive accuracy can be improved by integrating out model parameters and hyperparameters.

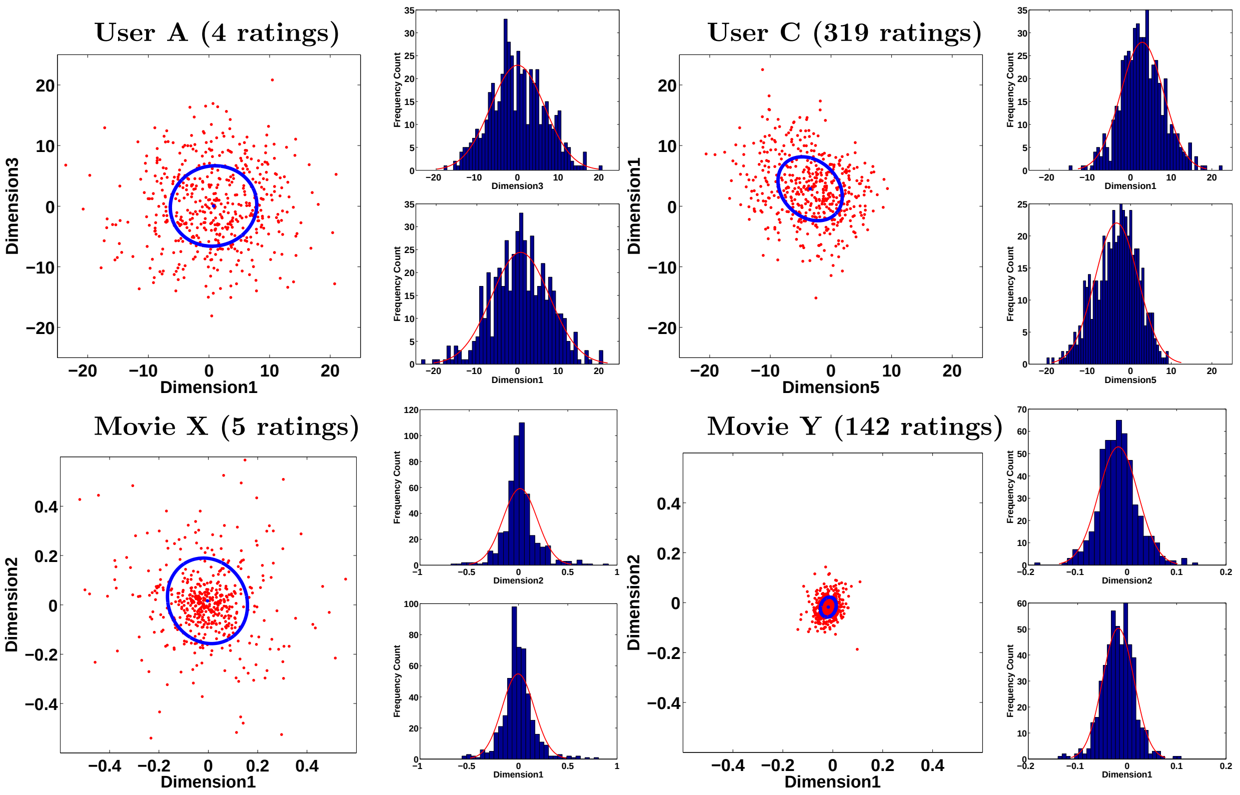

Figure 3. Samples from the posterior over the user and movie feature vectors generated by each step of the Gibbs sampler. The two dimensions with the highest variance are shown for two users and two movies. The first 800 samples were discarded as "burn-in".

D	Valid. RMSE		%	Test RMSE		%
	PMF	BPMF	Inc.	PMF	BPMF	Inc.
30	0.9154	0.8994	1.74	0.9188	0.9029	1.73
40	0.9135	0.8968	1.83	0.9170	0.9002	1.83
60	0.9150	0.8954	2.14	0.9185	0.8989	2.13
150	0.9178	0.8931	2.69	0.9211	0.8965	2.67
300	0.9231	0.8920	3.37	0.9265	0.8954	3.36

Table 1. Performance of Bayesian PMF (BPMF) and linear PMF on Netflix validation and test sets.

We than trained larger PMF models with $D = 40$ and $D = 60$. Capacity control for such models becomes a rather challenging task. For example, a PMF model with $D = 60$ has approximately 30 million parameters. Searching for appropriate values of the regularization coefficients becomes a very computationally expensive task. Table 1 further shows that for the 60-dimensional feature vectors, Bayesian PMF outperforms its MAP counterpart by over 2%. We should also point out that even the simplest possible Bayesian extension of the PMF model, where Gamma priors are placed over the precision hyperparameters α_U and α_V (see Fig. 1, left panel), significantly outperforms the MAP-trained PMF models, even though it does not perform as well as the Bayesian PMF models.

It is interesting to observe that as the feature dimensionality grows, the performance accuracy for the MAP-trained PMF models does not improve, and controlling overfitting becomes a critical issue. The predictive accuracy of the Bayesian PMF models, however, steadily improves as the model complexity grows. Inspired by this result, we experimented with Bayesian PMF models with $D = 150$ and $D = 300$ feature vectors. Note that these models have about 75 and 150 million parameters, and running the Gibbs sampler becomes computationally much more expensive. Nonetheless, the validation set RMSEs for the two models were 0.8931 and 0.8920. Table 1 shows that these models not only significantly outperform their MAP counterparts but also outperform Bayesian PMF models that have fewer parameters. These results clearly show that the Bayesian approach does not require limiting the complexity of the model based on the number of the training samples. In practice, however, we will be limited by the available computer resources.

For completeness, we also report the performance results on the Netflix *test set*. These numbers were ob-

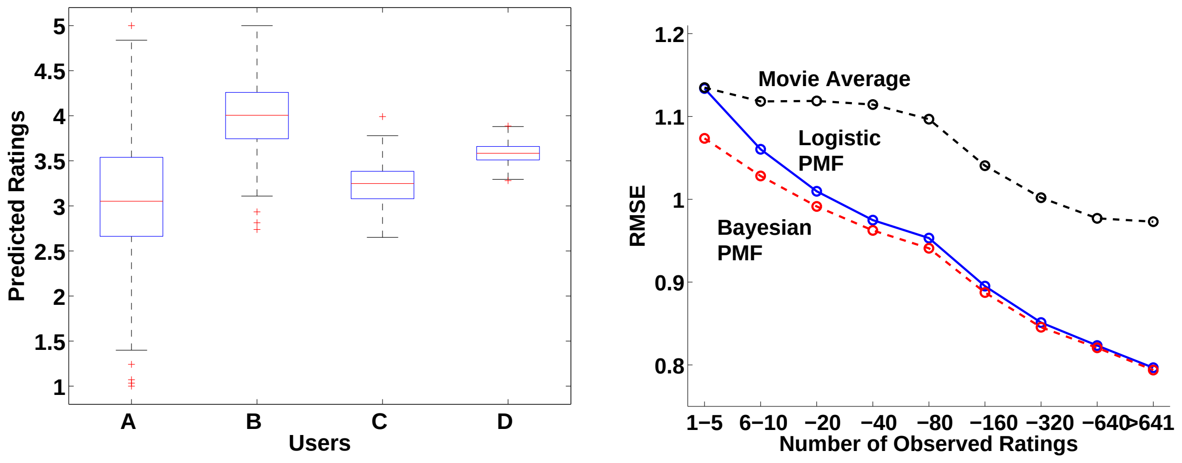

Figure 4. Left panel: Box plot of predictions, obtained after each full Gibbs step, for 4 users on a randomly chosen test movies. Users A,B,C, and D have 4, 23, 319 and 660 ratings respectively. Right panel: Performance of Bayesian PMF, logistic PMF, and the movie average algorithm that always predicts the average rating of each movie. The users were grouped by the number of observed ratings in the training data. The linear PMF model performed slightly worse than the logistic PMF model.

tained by submitting the predicted ratings to Netflix who then provided us with the test score on an unknown half of the test set. The test scores are slightly worse than the validation scores, but the relative behavior across all models remains the same.

To diagnose the convergence of the Gibbs sampler, we monitored the behaviour of the Frobenius norms of the model parameters and hyperparameters: U, V, Λ, and μ. Typically, after a few hundred samples these quantities stabilize. Fig. 2 (right panel) shows the RMSE error on the Netflix validation set for Bayesian PMF models as the number of samples increases. After obtaining a few hundred samples[1] the predictive accuracy does not significantly improve. Note that the initial predictive accuracy is already high because the Markov chain is initialized using the MAP values of the model parameters.

For the Bayesian PMF model with $D = 30$ we also collected samples over the user and movie feature vectors generated by each full step of the Gibbs sampler. The first 800 samples were discarded as "burn-in". Figure 3 shows these samples for two users and two movies projected onto the two dimensions of the highest variance. Users A and C were chosen by randomly picking among rare users who have fewer than 10 ratings and more frequent users who have more than 100 ratings in the training set. Movies X and Y were chosen in the same way. Note that the empirical distribution of the samples from the posterior appear to be non-Gaussian.

Using these samples from the posterior we also looked at the uncertainty of the predictions of four users on randomly chosen test movies. Figure 4 (left panel) shows results for users A,B,C, and D who have 4, 23, 319 and 660 ratings respectively. Note that there is much more uncertainty about the prediction of user A than about the prediction of user D, whose feature vector is well-determined. Figure 4 (right panel) shows that the Bayesian PMF model considerably outperforms the logistic PMF model on users with few ratings. As the number of ratings increases, both the logistic PMF and the Bayesian PMF exhibit similar performance.

The advantage of Bayesian PMF models is that by averaging over all settings of parameters that are compatible with the data as well as the prior they deal with uncertainty more effectively than the non-Bayesian PMF models, which commit to a single most probable setting.

Since the main concern when applying Bayesian methods to large datasets is their running time, we provide the times for our simple Matlab Bayesian PMF implementation. One full Gibbs step on a single core of a recent Pentium Xeon 3.00GHz machine for models with $D = 10, 30, 60, 300$ takes 6.6, 12.9 , 31.6, and 220 minutes respectively. Note that the most expensive aspect of training Bayesian PMF models is the inversion of a $D \times D$ matrix per feature vector[2] (see Eq. 13),

[1]We store the model parameters after each full Gibbs step as a sample. The fact that these samples are not independent does not matter for making predictions.

[2]In our implementation, we solve a system of D equa-

which is an $O(D^3)$ operation.

5. Conclusions

We have presented a fully Bayesian treatment of Probabilistic Matrix Factorization by placing hyperpriors over the hyperparameters and using MCMC methods to perform approximate inference. We have also demonstrated that Bayesian PMF models can be successfully applied to a large dataset containing over 100 million movie ratings, and achieve significantly higher predictive accuracy compared to the MAP-trained PMF models with carefully tuned regularization parameters. An additional advantage of using a Bayesian model is that it provides a predictive distribution instead of just a single number, allowing the confidence in the prediction to be quantified and taken into account when making recommendations using the model.

Using MCMC instead of variational methods for approximate inference in Bayesian matrix factorization models leads to much larger improvements over the MAP trained models, which suggests that the assumptions made by the variational methods about the structure of the posterior are not entirely reasonable. This conclusion is confirmed by inspecting the empirical distribution of the samples from the posterior, which appears to be significantly non-Gaussian.

A major problem of MCMC methods is that it is hard to determine when the Markov chain has converged to the desired distribution. In practice, we have to rely on rules of thumb to diagnose convergence, which means that there is a risk of using samples from a distribution that differs from the true posterior distribution, potentially leading to suboptimal predictions. Our results show that this problem is not a sufficient reason to reject MCMC methods.

For our models, the number of samples from the posterior that can be generated within a reasonable amount of time will typically be constrained by the available computer resources. However, as mentioned above, sampling the feature vectors for multiple users or movies in parallel provides an easy way to greatly speed up the process of generating samples using multiple cores.

Acknowledgments

We thank Geoffrey Hinton for many helpful discussions. This research was supported by NSERC.

tions instead of inverting a matrix. The computational cost of this operation is still $O(D^3)$.

References

Hinton, G. E., & van Camp, D. (1993). Keeping the neural networks simple by minimizing the description length of the weights. *COLT* (pp. 5–13).

Hofmann, T. (1999). Probabilistic latent semantic analysis. *Proceedings of the 15th Conference on Uncertainty in AI* (pp. 289–296). San Fransisco, California: Morgan Kaufmann.

Jordan, M. I., Ghahramani, Z., Jaakkola, T. S., & Saul, L. K. (1999). An introduction to variational methods for graphical models. *Machine Learning*, *37*, 183.

Lim, Y. J., & Teh, Y. W. (2007). Variational Bayesian approach to movie rating prediction. *Proceedings of KDD Cup and Workshop*.

Marlin, B. (2004). Modeling user rating profiles for collaborative filtering. In S. Thrun, L. Saul and B. Schölkopf (Eds.), *Advances in neural information processing systems 16*. Cambridge, MA: MIT Press.

Marlin, B., & Zemel, R. S. (2004). The multiple multiplicative factor model for collaborative filtering. *Machine Learning, Proceedings of the Twenty-first International Conference (ICML 2004), Banff, Alberta, Canada*. ACM.

Neal, R. M. (1993). *Probabilistic inference using Markov chain Monte Carlo methods* (Technical Report CRG-TR-93-1). Department of Computer Science, University of Toronto.

Nowlan, S. J., & Hinton, G. E. (1992). Simplifying neural networks by soft weight-sharing. *Neural Computation*, *4*, 473–493.

Raiko, T., Ilin, A., & Karhunen, J. (2007). Principal component analysis for large scale problems with lots of missing values. *ECML* (pp. 691–698).

Rennie, J. D. M., & Srebro, N. (2005). Fast maximum margin matrix factorization for collaborative prediction. *Machine Learning, Proceedings of the Twenty-Second International Conference (ICML 2005), Bonn, Germany* (pp. 713–719). ACM.

Salakhutdinov, R., & Mnih, A. (2008). Probabilistic matrix factorization. *Advances in Neural Information Processing Systems 20*. Cambridge, MA: MIT Press.

Srebro, N., & Jaakkola, T. (2003). Weighted low-rank approximations. *Machine Learning, Proceedings of the Twentieth International Conference (ICML 2003), Washington, DC, USA* (pp. 720–727). AAAI Press.