Tracking User-Preference Varying Speed in Collaborative Filtering

Ruijiang Li
School of Computer Science, Fudan University
Shanghai 200433, China
rjli@fudan.edu.cn

Bin Li
QCIS Centre, FEIT, University of Technology, Sydney
NSW 2007, Australia
bin.li-1@uts.edu.au

Cheng Jin and **Xiangyang Xue**
School of Computer Science, Fudan University
Shanghai 200433, China
{jc,xyxue}@fudan.edu.cn

Xingquan Zhu
QCIS Centre, FEIT, University of Technology, Sydney
NSW 2007, Australia
xqzhu@it.uts.edu.au

Abstract

In real-world recommender systems, some users are easily influenced by new products and whereas others are unwilling to change their minds. So the preference varying speeds for users are different. Based on this observation, we propose a dynamic nonlinear matrix factorization model for collaborative filtering, aimed to improve the rating prediction performance as well as track the preference varying speeds for different users. We assume that user-preference changes smoothly over time, and the preference varying speeds for users are different. These two assumptions are incorporated into the proposed model as prior knowledge on user feature vectors, which can be learned efficiently by MAP estimation. The experimental results show that our method not only achieves state-of-the-art performance in the rating prediction task, but also provides an effective way to track user-preference varying speed.

1 Introduction

Recent years have witnessed an increasing number of research studies on collaborative filtering due to the rapid growth of recommendation sites on the Internet. A typical task of collaborative filtering is to fill in an incomplete sparse matrix with the elements of which containing the ratings made by a collection of users on a collection of items. Roughly speaking, existing methods for collaborative filtering can be divided into two categories: memory based (Resnick et al. 1994; Sarwar et al. 2001) and model based (Salakhutdinov and Mnih 2008a; Porteous, Bart, and Welling 2008; Lawrence and Urtasun 2009; Li, Yang, and Xue 2009).

Most existing methods model a static rating process, in which all the variables, such as the user-preferences, do not change over time. However, in practice, the rating process may last for a long time, during which a user's properties, such as interest, may change due to various reasons. For example, a boy who used to love classical music might become interested in pop music after he got a fashion girlfriend, and an IT nerd may get rid of his favors for traditional laptop after trying the new easy-to-use iPad. These observations suggest that the context for the rating process keeps changing over time, for which the static modeling is not enough.

There are some recent works considering the collaborative filtering problem in a dynamic context. Like the static methods, these approaches can also be classified as memory based (Ding and Li 2005) and model based (Koren 2009; Lu, Agarwal, and Dhillon 2009; Hayashi, Hirayama, and Ishii 2009; Xiong et al. 2010; Khoshneshin and Street 2010). One of the most well-known works is the Netflix price winner TimeSVD++ (Koren 2009), in which user/item feature vectors are allowed to change over time. Since online learning provides an elegant way to adapt the model to the changing context, some online algorithms were recently proposed to tackle the collaborative filtering problem (Das et al. 2007; Liu et al. 2010).

Although existing dynamic collaborative filtering methods can model users' dynamics, they have not taken into account the fact that users' preferences may change at different speeds. In reality, some users, such as young people, are easily influenced by new products or ideas, for which the preference varying speed should be high; while some other users, such as aged people or conservatives, are unwilling to change their minds, for which the preference varying speed should be low, or even unobservable. The users in a recommender system differs in preference varying speeds, which should be incorporated into the model for personalized recommendation.

In this paper, we propose a new dynamic collaborative filtering model called dynamic nonlinear matrix factorization (DNMF) based on the following two assumptions: 1) *the user preference changes smoothly over time,* 2) *the preference varying speeds for different users are different*. The first assumption is widely adopted, and proven to be valid in many dynamic models (Blei and Lafferty 2006; Wang, Fleet, and Hertzmann 2008; Hayashi, Hirayama, and Ishii 2009; Xiong et al. 2010). The second assumption motivates the proposed work. We take into account these two assumptions as temporal prior knowledge, and train the model by maximum *a posteriori* (MAP) estimation of the user feature vectors. We choose non-linear matrix factorization (NMF) (Lawrence and Urtasun 2009) as our base framework, in which the item feature vectors are marginalized out, which makes it possible to estimate the user-preference varying speed by considering the user feature vectors only. Experimental results show that our method not only achieves state-of-the-art performance in the rating prediction task, but

also offers an effective way to track the preference varying speeds for different users.

The remainder of the paper is organized as follows. In Section 2, we introduce our model, and describe the procedures for training and prediction. In Section 3, we discuss the relationships between our model and related existing methods. In Section 4, we evaluate our method on both synthetic and Netflix dataset. In Section 5, we conclude the paper and discuss the future work.

2 Dynamic Nonlinear Matrix Factorization

Problem Formulation and Notations

The task of dynamic collaborative filtering is to fill in an incomplete rating matrix whose ratings are associated with time-stamps. Specifically, we are given N ratings made by U users on M items during T time slices, and our task is to predict the rating made by a certain user u^* on a certain item m^* at a certain time slice t^*. We denote the user set by $\mathcal{U}$ and item set by $\mathcal{M}$, and let $r_{um}^{(t)}$ be the rating made by user u on item m at time slice t ($u \in \mathcal{U}, m \in \mathcal{M}, t = 1, 2, \cdots, T$).

Our model is based on the matrix factorization method. For item m, we let a d dimensional vector $\mathbf{x}_m$ be its feature vector, which does not change over time. And for user u, we let a d dimensional vector $\mathbf{w}_u^{(t)}$ be his/her feature vector at time slice t. Totally we have $U \times T$ user feature vectors and M item feature vectors.

For convenience, we let $\mathbf{W}_u^\top = [\mathbf{w}_u^{(1)}, \mathbf{w}_u^{(2)}, \cdots, \mathbf{w}_u^{(T)}]$, $\mathbf{X}^\top = [\mathbf{x}_m]_{m \in \mathcal{M}}$, $\mathbf{W}^\top = [\mathbf{W}_u^\top]_{u \in \mathcal{U}}$. All the $U \times M \times T$ ratings is denoted by $\mathbf{R}$, and N observed ratings is denoted by $\mathbf{R}^{obs}$. $\mathbf{r}_m^{obs}$ denotes the observed ratings on item m, and N_m denotes the number of ratings on item m.

Nonlinear Matrix Factorization

In the matrix factorization framework, rating $r_{um}^{(t)}$ is approximated with the inner product of two corresponding feature vectors,

$$p(r_{um}^{(t)} | \mathbf{x}_m, \mathbf{w}_u^{(t)}) = \mathcal{N}(r_{um}^{(t)} | \mathbf{x}_m^\top \mathbf{w}_u^{(t)}, \alpha_r^{-1} I)$$

where α_r is the parameter for noise. In recent probabilistic matrix factorization models (Salakhutdinov and Mnih 2008b; 2008a; Lawrence and Urtasun 2009; Hayashi, Hirayama, and Ishii 2009), a Gaussian prior is also given to the item feature vectors to control the capacity of the model:

$$p(\mathbf{x}_m) = \mathcal{N}(\mathbf{x}_m | 0, \alpha_x^{-1} I)$$

where α_x is the parameter for prior of item feature. Like NMF (Lawrence and Urtasun 2009), we marginalize out the item feature vectors $\mathbf{X}$ and obtain

$$\begin{aligned} p(\mathbf{R}|\mathbf{W}) &= \prod_{m \in \mathcal{M}} \int \prod_{u=1}^{U} \prod_{t=1}^{T} p(r_{um}^{(t)} | \mathbf{x}_m, \mathbf{w}_u^{(t)}) p(\mathbf{x}_m) \mathrm{d}\mathbf{x}_m \\ &= \prod_{m \in \mathcal{M}} \mathcal{N}(\mathbf{r}_m | 0, \alpha_x^{-1} \mathbf{W}\mathbf{W}^\top + \alpha_r^{-1} I) \end{aligned} \quad (1)$$

The $\mathbf{W}\mathbf{W}^\top$ in (1) could be treated as a kernel matrix computed over all the $U \times T$ user feature vectors. Using the marginalization property of multivariate normal distribution, we can easily marginalize out the unobserved ratings and get

$$p(\mathbf{R}^{obs}|\mathbf{W}) = \prod_{m \in \mathcal{M}} \mathcal{N}(\mathbf{r}_m^{obs} | 0, \alpha_x^{-1} \mathbf{W}_m^{obs} \mathbf{W}_m^{obs\top} + \alpha_r^{-1} \mathbf{I}) \quad (2)$$

where $\mathbf{W}_m^{obs}$ is a submatrix of $\mathbf{W}$ corresponding to $\mathbf{r}_m^{obs}$.

If we view $\mathbf{C} = \alpha_x^{-1} \mathbf{W}\mathbf{W}^T + \alpha_r^{-1} \mathbf{I}$ as a covariance matrix, the model becomes a special case of Gaussian Process Latent Variable Models (GPLVM) (Lawrence and Hyvärinen 2005) with a linear covariance function. We can replace $\mathbf{W}\mathbf{W}^\top$ in $\mathbf{C}$ with other Mercer kernels to get a nonlinear factorization model. The kernelization also provides a natural way to utilize the user side information such as age, sex, etc. For example, we can use a combined kernel function $k(\mathbf{w}_{um}^{(t)}, \mathbf{w}_{u'm'}^{(t')}) = k_0(\mathbf{w}_{um}^{(t)}, \mathbf{w}_{u'm'}^{(t')}) + sim(u, u')$, where k_0 is a kernel function like RBF, and $sim(u, u')$[1]indicates the similarity computed from the side information of user u and u'. Furthermore, if the user feature vectors $\mathbf{W}$ are known, we can treat $\mathbf{R}$ in (1) as the multiple outputs in a Gaussian Process Regression model with inputs $\mathbf{W}$, in which way the prediction can be made as the Gaussian Process Regression Model (Rasmussen and Williams 2004).

If all the ratings are made in the same time slice ($T = 1$ and there are only U user feature vectors), our model reduces to the NMF, in which $\mathbf{W}$ is found by maximizing $p(\mathbf{R}^{obs}|\mathbf{W})$ in (2) with stochastic gradient descent method. For $T > 1$, it is still possible to find $\mathbf{W}$ with the same method, but in this way we have to make the assumption that the user feature vectors at different time slices are independent, which is not reasonable. Another issue of this approach is that optimization w.r.t. $U \times d \times T$ unconstrained variables will easily overfit the model due to the limited training samples (the rating matrix is usually sparse in collaborative filtering).

Temporal Model

In a probabilistic framework, a straightforward way to constrain a variable is to introduce a prior to it. By assuming that for each user, the user feature vector is changing smoothly, we can give the following priors to the user feature vectors for each user u at time slice $t = 2, 3, \cdots, T$,

$$p(\mathbf{w}_u^{(t)} | \mathbf{w}_u^{(t-1)}, \sigma_u^2) = \mathcal{N}(\mathbf{w}_u^{(t)} | \mathbf{w}_u^{(t-1)}, \sigma_u^2 I) \quad (3)$$

where σ_u^2 is a parameter for user u. Further we can easily write down the prior for $\mathbf{W}_u$ by multiplying (3) over t,

$$\begin{aligned} p(\mathbf{W}_u | \mathbf{w}_u^{(1)}, \sigma_u^2) &= \prod_{t=2}^{T} p(\mathbf{w}_u^{(t)} | \mathbf{w}_u^{(t-1)}, \sigma_u^2) \\ &= \prod_{t=2}^{T} \mathcal{N}(\mathbf{w}_u^{(t)} | \mathbf{w}_u^{(t-1)}, \sigma_u^2 I) \end{aligned} \quad (4)$$

The temporal modeling in (3), i.e., the user feature vector at certain time slice only depends on its predecessor, is widely used in many applications such as topic extraction

[1]The similarity should be preprocessed to ensure that $k(\cdot, \cdot)$ is positive definite.

(Blei and Lafferty 2006), motion analysis (Wang, Fleet, and Hertzmann 2008) and collaborative filtering (Xiong et al. 2010). The σ_u^2 in (3) reflects the preference varying speed for user u. More specifically, larger σ_u^2 allows $\mathbf{w}_u^{(t)}(t = 1, 2, \cdots T)$ to be spread widely, indicating that the user preference is changing fast, and smaller σ_u^2 discourages the moving of user feature vector, indicating that the user preference is changing slowly.

When $\sigma_u^2 \to 0$, the user preference vectors at different time slices are restricted to be same. When $\sigma_u^2 \to +\infty$, the temporal dependence of user preference vectors vanishes. Thus, the value of σ_u^2 controls the capacity of DNMF. Instead of computing individual σ_u^2's for U users, which is difficult and inaccurate, we take a Bayesian treatment by introducing an inverse gamma distribution as a conjugate prior to each σ_u^2,

$$\begin{aligned} p(\sigma_u^2|\alpha, \beta) &= \text{invgam}(\sigma_u^2|\alpha, \beta) \\ &= \frac{\beta^\alpha}{\Gamma(\alpha)}(\sigma_u^2)^{(-\alpha-1)} \exp(\frac{-\beta}{\sigma_u^2}) \end{aligned} \tag{5}$$

where $\Gamma(\cdot)$ is the gamma function. Multiplying (2), (4) and (5) we write down the full probability of our model,

$$\begin{aligned} & p(\mathbf{R}^{obs}, \mathbf{W}, \Sigma|\alpha, \beta, \alpha_x, \alpha_r) \\ = & \prod_{m \in \mathcal{M}} \mathcal{N}(\mathbf{r}_m^{obs}|0, \mathbf{C}_m^{obs}) \\ & \times \prod_{u \in \mathcal{U}} \text{invgam}(\sigma_u^2|\alpha, \beta) \prod_{t=2}^{T} \mathcal{N}(\mathbf{w}_u^{(t)}|\mathbf{w}_u^{(t-1)}, \sigma_u^2 I) \end{aligned} \tag{6}$$

where Σ denotes all the σ_u^2s, and $\mathbf{C}_m^{obs}$ is a submatrix of $\mathbf{C}$ corresponding to $\mathbf{r}_m^{obs}$.

Learning

Note that inverse gamma distribution is conjugate to the normal distribution, we first marginalize Σ out in (6), then find the maximum a *posteriori* (MAP) estimation of $\mathbf{W}$. The negative log likelihood we are going to minimize is written as follows:

$$\begin{aligned} NL(\mathbf{W}) &= -\log p(\mathbf{R}^{obs}, \mathbf{W}|\alpha, \beta, \alpha_x, \alpha_r) \\ &= \sum_{m \in \mathcal{M}} E(\mathbf{W}_m^{obs}) + \sum_{u \in \mathcal{U}} F(\mathbf{W}_u) \end{aligned} \tag{7}$$

where

$$E(\mathbf{W}_m^{obs}) = \frac{1}{2}\log \mathbf{C}_m^{obs} + \frac{1}{2}\mathbf{r}_m^{obs\top}\mathbf{C}_m^{obs-1}\mathbf{r}_m^{obs}$$

$$F(\mathbf{W}_u) = \frac{2\alpha + d(T-1)}{2}\log(\beta + \frac{1}{2}\sum_{t=2}^{T}||\mathbf{w}_u^{(t)} - \mathbf{w}_u^{(t-1)}||_2^2)$$

It is clear that $E(\mathbf{W}_m^{obs})$ comes from the likelihood for item m in (2), which is similar to the likelihood in NMF, and $F(\mathbf{W}_u)$ comes from the prior of $\mathbf{W}$ in (4), which is introduced by temporal modeling. We don't use Newton based optimization method to minimize $NL(\mathbf{W})$ in (7), because it is not affordable to store or compute the Hessian matrix of $\mathbf{W}$ consisting of $U \times T \times d$ variables. Stochastic gradient descent method for NMF could not be used directly either, because $F(\mathbf{W}_u)$ can not be easily converted into the sum over m. We use a gradient based batch update algorithm for DNMF: In each epoch, we first randomly divide the m items into batches, each of which contains q items (the last batch may contain less than q items), then perform gradient descent on each batch. The gradient for each batch $\mathcal{B}$ is computed as $\sum_{m \in \mathcal{B}} \frac{\partial E(\mathbf{W}_m^{obs})}{\partial \mathbf{W}} + \sum_{u \in \mathcal{U}} \frac{\partial F(\mathbf{W}_u)}{\partial \mathbf{W}}$. Practically we find that the algorithm converges to a local minimum.

The time complexity for the gradient computation depends on the kernel type. For linear kernel, i.e. $\mathbf{C}_m^{obs} = \alpha_x^{-1}\mathbf{W}_m^{obs}\mathbf{W}_m^{obs\top} + \alpha_r^{-1}\mathbf{I}$, the time complexity for computing $\frac{\partial E(\mathbf{W}_m^{obs})}{\partial \mathbf{W}}$ is $\mathcal{O}(d^3 + dN_m^2)$ (see (Lawrence and Urtasun 2009) for a detailed form of the gradient). and the time cost for computing $\frac{\partial F(\mathbf{W}_u)}{\partial \mathbf{W}}$ is $\mathcal{O}(dT)$. Thus the average time cost for each epoch is $\mathcal{O}(Md^3 + Md\langle N_m^2 \rangle + \frac{d}{q}UTM)$. Compared with NMF, DNMF has to compute the extra gradient of prior for $\frac{M}{q}$ times.

Prediction

When we get $\mathbf{W}$ after training, the prediction procedure for DNMF is as same as that in Gaussian Process Regression (Rasmussen and Williams 2004). Specifically, we have the predicted distribution of the rating made by user u^*on item m^* at time slice t^* as follows,

$$\begin{aligned} p(r_{u^*m^*}^{(t^*)}|\mathbf{R}^{obs}, \mathbf{W}) &= \mathcal{N}(r_{u^*m^*}^{(t^*)}|u^*, v^*) \\ u^* &= \mathbf{k}^{*\top}(\mathbf{C}_{m^*}^{obs})^{-1}\mathbf{r}_{m^*}^{obs} \\ v^* &= k^{**} - \mathbf{k}^{*\top}(\mathbf{C}_{m^*}^{obs})^{-1}\mathbf{k}^* \end{aligned} \tag{8}$$

where $\mathbf{k}^*$ is the column vector in the kernel matrix $\mathbf{C}$ corresponding to the similarities between $\mathbf{W}_{m^*}$ and $\mathbf{w}_{u^*m^*}^{(t^*)}$, and k^{**} is the element in the kernel matrix $\mathbf{C}$ corresponding to the similarities between $\mathbf{w}_{u^*m^*}^{(t^*)}$ and $\mathbf{w}_{u^*m^*}^{(t^*)}$.

Tracking User-Preference Varying Speed

Tracking user-preference varying speed becomes straightforward after we get the optimal $\mathbf{W}$. For user u, we compute the preference varying speed l_u as follows:

$$l_u = \frac{\sum_{t=2}^{T}||\mathbf{w}_u^{(t)} - \mathbf{w}_u^{(t-1)}||_2^2}{T-1} \tag{9}$$

The l_u obtained from (9) is a *relative* value, which only makes sense when compared with $l_{u'}$ (for another user u') trained with the same kernel parameter settings. We can see the reason from (2): If we deem that there is a ground truth for the kernel matrix $\mathbf{C}$, α_X will influence the scale of $\mathbf{W}$, which will further influence the scale of l_u.

From (7), we can also see the necessity for choosing NMF as our base method. In NMF, the item feature vectors $\mathbf{X}$ are marginalized out, making the computation of $\mathbf{W}$ independent of $\mathbf{X}$. Without marginalization, the scale of $\mathbf{W}$ is also influenced by the scale of $\mathbf{X}$, in this case tracking the user preference varying speed becomes difficult.

3 Related Works

Our model has close relationships with the matrix factorization models, the core idea of which is that the rating matrix is a product of two low rank matrices. Earlier works such as SVD (Srebro and Jaakkola 2003) is not based on the probabilistic framework, for which the parameters penalizing the two matrices is tuned through validation. Recent works on matrix factorization (Salakhutdinov and Mnih 2008b; 2008a) take the probabilistic (Salakhutdinov and Mnih 2008b) and Bayesian treatments (Salakhutdinov and Mnih 2008a), which have the advantage in controlling the model capacity. NMF (Lawrence and Urtasun 2009) reveals the relationship between matrix factorization and the GPLVM (Lawrence and Hyvärinen 2005), which directly inspires our research.

There are also a number of works on dynamic matrix factorization. TimeSVD++ (Koren 2009) is a non-probabilistic approach to collaborative filtering, in which the total variance of user feature vectors at different time slices are used for regularization. (Hayashi, Hirayama, and Ishii 2009) considered the case in which the feature vectors for two dimensions (user and item) in matrix factorization are both varying over time, but their model might not get a good interpretation in our problem since the item preference is not changing over time in collaborative filtering. (Xiong et al. 2010) incorporated the temporal information with extra feature vectors in a third temporal dimension, and extend the matrix factorization to tensor factorization. Although their model enjoys good mathematical form and performance, the third dimension still lacks exact physical meaning. The temporal modeling in STKF (Lu, Agarwal, and Dhillon 2009) is similar to our model, but we differs in two aspects: Firstly, STKF focuses on incorporating spatial (side)-temporal information, while our model focuses on the preference varying speeds of different users. Secondly, different users share the same random walk speed in STKF, while in our model, each user has his/her own varying speed.

4 Experiments

Dataset

We prepare two datasets for evaluation. The synthetic dataset is used to show that DNMF is effective when our assumptions are true, i.e., the user feature vectors over time is generated according to (3), and the Netflix dataset[2] is used for comparison with the other four state-of-the-art methods.

Synthetic Dataset The synthetic dataset consists of the ratings made by 100 users on 100 items for 40 time slices. We generated the dataset according to the following steps:

1. Generate 100 item preference vectors with dimensionality 2 from the normal distribution: $\mathcal{N}(\cdot|0, \frac{\mathbf{I}}{10})$.
2. For the first time slice, generated 100 initial user feature vectors with dimensionality 2 from the normal distribution $\mathcal{N}(\cdot|0, \frac{\mathbf{I}}{10})$.
3. Generate a speed σ^2 for each user from the inverse gamma distribution $\mathrm{invgam}(\cdot|1, 1)$.
4. For each user, generate the feature vectors for the succeeding 39 time slices according to the following rule: $p(\mathbf{w}_t|\mathbf{w}_{t-1}) = \mathcal{N}(\mathbf{w}_t|\mathbf{w}_{t-1}, \sigma^2 I)$ for $t = 2, \cdots, 40$, where $\mathbf{w}_t$ is the feature vector for time slice t, and σ^2 is the speed generated form Step 3. Note that the scale of the user feature vector increases as t increases.
5. Compute the ratings using the inner products of the generated user feature vectors and item feature vectors.

The generated ratings for the first time slice range from -0.8 to 0.8, and for the last time slice range from -25 to 25. 5% of the ratings are used as training set, and the rest 95% are used as testing set.

Pruned Netflix Dataset The original Netflix dataset contains 100 million ratings made by $\sim 480,000$ users on $\sim 17,000$ movies from 1999 to 2005. Each rating comes with its own time stamp, and each movie comes with the year it came to Netflix. We perform pruning according to the following steps:

1. Keep the ratings made between Jan 2003 and Dec 2005 (36 months). The time stamp for each rating is then rounded to the corresponding month number indexed from 1 to 36.
2. Keep the users who has rated more than 100 movies. Also, keep the users who have more than 15 ratings for at least 4 months. After this step, we have 6784 users left.
3. Keep the movies which came to Netflix before 2003, and remove the movies which were rated less than 50 times. After this step, we have 3287 movies left.

After pruning, we have $1,005,019$ ratings left. We split these ratings into training set and test set for 10 times, at each time, the density of the training set is controlled around 0.9%. The first 5 splits are used for validation, and the rest 5 splits are used for evaluation.

Evaluation Protocol

Our evaluation metric is RMSE (Root Mean Square Error): $\sqrt{\sum_{i \in S}(r_i - \hat{r}_i)^2/|S|}$, where S is the set for testing ratings, $\hat{r}_i$ is the prediction, r_i is the ground truth. Smaller RMSE indicates better performance.

Results

For DNMF, we always use the linear kernel in (1). We use μ^* in (8) as the predicted rating. And the following parameters is set empirically: $\alpha_R = 1$, $\alpha_x = 10$, initial scale for gradient descent $s_0 = 10^{-1}$, number of epochs 10, $\alpha = 1$ and $\beta = 1$.

Synthetic Dataset For DNMF, we set the step size for the gradient descent algorithm to 0.01, and batch size q to 10 (empirically set). The experiment (from generation to prediction) is repeated 5 times, and the overall RMSE on the test sets is 0.2558 ± 0.0039. In the following we only show the detailed result of one time (randomly chosen).

The RMSEs of the training set and test set over time are show in Figure 1 panel (d), an interesting point of the result is that RMSE does not increase along with the rating scale.

[2] http://www.netflix.com

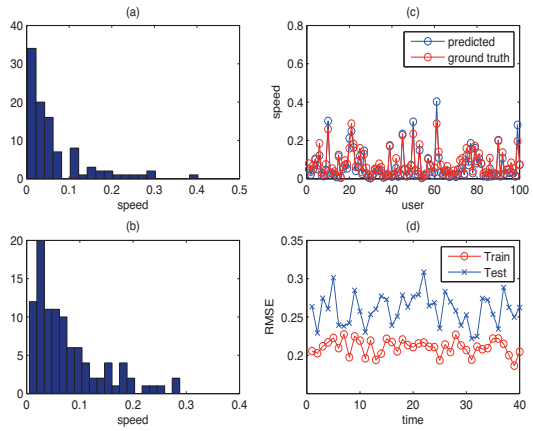

Figure 1: Results on the synthetic dataset (better viewed in color). (a) and (b) show the histograms of the predicted and ground truth vectors, respectively. (c) shows the two vectors directly. (d) shows the RMSEs on the training set and test set over time.

We also compute the varying speeds for all users according to (9), denoted as two 100 dimensional vectors, computed from the learnt $\mathbf{W}$ the ground truth, respectively. In Figure 1, panel (a) and (b) show the histogram of these two normalized vectors, and panel (c) plots these two normalized vectors directly. We can see from panel (c) that the normalized predicted preference varying speed is close to the ground truth. The inner product of these two normalized vectors are 0.9663 ± 0.0052 (computed over the 5 repeats).

Pruned Netflix Dataset We compare our method (DNMF) with four well-known methods: Nonlinear Matrix Factorization (NMF)[3](Lawrence and Urtasun 2009), Weight Low-Rank Approximation (SVD) (Srebro and Jaakkola 2003), Bayesian Probabilistic Tensor Factorization (BPTF)[4](Xiong et al. 2010) and TimeSVD++ (Koren 2009). Among the five methods, TimeSVD++, DNMF, BPTF are dynamic models, and SVD, NMF are static models. For the static models, the ratings from all the time slices in the training set are used for training.

Following the default settings in the BPTF program, we set the dimensionality of the feature factors (d) in all the five methods to 10. For SVD and TimeSVD++, the parameters are tuned on the 5 validation sets, i.e. choose the parameters minimizing the average test RMSE on validation sets. For NMF, we use the same parameter settings as suggested in their paper. For BPTF, we use the default parameters for Netflix dataset in their code. For DNMF, we only choose the step size from $\{10^{-3}, 10^{-2}, 10^{-1}\}$, which makes the average test RMSE on the validation set look acceptable. Finally we set the step size to be 0.01, and batch size q to be 200.

The overall RMSEs of the five methods are reported in Table 1. We can see that DNMF outperforms its base method

[3]http://www.cs.man.ac.uk/~neill/collab/

[4]http://www.cs.cmu.edu/~lxiong/bptf/bptf.html

Method Name	RMSE(mean±std)
SVD	0.9426 ± 0.0013
TimeSVD++	0.9282 ± 0.0018
NMF	0.8958 ± 0.0009
DNMF	0.8912 ± 0.0034
BPTF	0.8885 ± 0.0015

Table 1: Overall performance of five methods

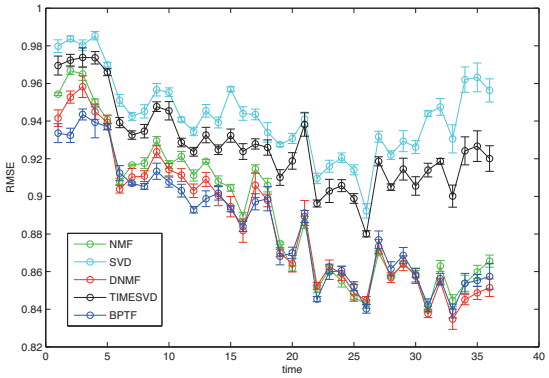

Figure 2: RMSE of the five methods at different time slices on Netflix dataset (better viewed in color). The horizontal axis represents the time slice indices, and the vertical axis represents the RMSE.

NMF, and is comparable with BPTF, which has the best overall performance in our experiment. We also plot the RMSEs at each time slice for each method, illustrated in Figure 2, from which we can see that BPTF performs better than DNMF in time slices $1-26$, and DNMF performs better than BPTF in time slice $27-36$.

TimeSVD++ performs better than its base method SVD, however is far worse than the result report in (Koren 2009). The major reason besides the dataset difference may be that our implementation of TimeSVD++ only constrains the total variance of the user feature vectors over time, which is only a subset of constraints used in the full TimeSVD++ methods.

It is difficult to evaluate our estimated user-preference varying speed since there is no ground truth. However, by visualizing the speed, we can still get some interesting points. We find that the distributions of the estimated user-preference varying speeds are consistent over the 5 splits. Figure 3 shows the histograms of user-preference varying speeds for all the users (computed according to (9) , not normalized), from which we can see that even if there are difference in the training sets as well as in the optimal $\mathbf{W}$, the estimated speed distributions for the population agree to some degree. Another interesting finding is that the histogram of varying speed is not heavily influenced by the inverse gamma prior in (5). The histogram in Figure 3 exhibits multiple modes, which may correspond to the conservatives and open-minded people in the population.

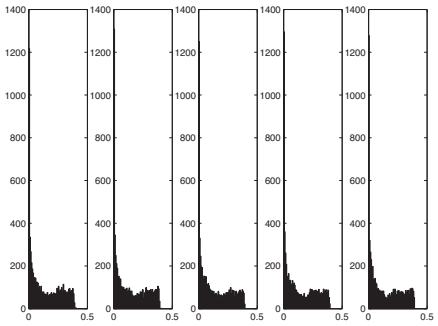

Figure 3: Histograms of varying speeds for different users. The 5 histograms correspond to 5 splits of the dataset. The horizontal axis represents the speed.

5 Conclusion

In this paper, we propose a dynamic non-linear matrix factorization model under the assumptions that the user-preference is varying smoothly over time, and the varying speeds for different users are different. We incorporate these two assumptions as prior knowledge into the NMF framework, and train the model by finding the MAP estimation of the user feature vectors, which are also used to estimate user-preference varying speed. Although more flexible than NMF, DNMF does not exhibit overfitting due to the effective temporal prior. The experimental results have shown that our model not only achieves state-of-the-art performance on Netflix dataset, but also provides a way to track the user-preference varying speed for population.

Acknowledgments

This work was supported by the National 973 Program of China (No. 2010CB327906), the NSF of China (No. 60873178), Australian Research Council's Discovery Project (No. DP1093762) and Future Fellowship (No. FT100100971).

References

Blei, D. M., and Lafferty, J. D. 2006. Dynamic topic models. In *Proc. of the 23rd Int'l Conf. on Machine Learning*, 113–120.

Das, A. S.; Datar, M.; Garg, A.; and Rajaram, S. 2007. Google news personalization: scalable online collaborative filtering. In *Proc. of the 16th Int'l Conf. on World Wide Web*, 271–280.

Ding, Y., and Li, X. 2005. Time weight collaborative filtering. In *Proc. of the 14th ACM Int'l Conf. on Information and Knowledge Management*, 485–492.

Hayashi, K.; Hirayama, J.; and Ishii, S. 2009. Dynamic Exponential Family Matrix Factorization. In *Proc. of the 13th Pacific-Asia Conf. on Knowledge Discovery and Data Mining*, 452–462.

Khoshneshin, M., and Street, W. N. 2010. Incremental collaborative filtering via evolutionary co-clustering. In *Proc. of the Fourth ACM Conf. on Recommender Systems*, 325–328.

Koren, Y. 2009. Collaborative filtering with temporal dynamics. In *Proc. of the 15th ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining*, 447–456.

Lawrence, N., and Hyvärinen, A. 2005. Probabilistic non-linear principal component analysis with Gaussian process latent variable models. *Journal of Machine Learning Research* 6:1783–1816.

Lawrence, N. D., and Urtasun, R. 2009. Non-linear matrix factorization with Gaussian processes. In *Proc. of the 26th Annual Int'l Conf. on Machine Learning*, 601–608.

Li, B.; Yang, Q.; and Xue, X. 2009. Can movies and books collaborate? cross-domain collaborative filtering for sparsity reduction. In *Proc of the 21st Int'l Joint Conf. on Artificial Intelligence*, 2052–2057.

Liu, N. N.; Zhao, M.; Xiang, E.; and Yang, Q. 2010. Online evolutionary collaborative filtering. In *Proc. of the Fourth ACM Conf. on Recommender Systems*, 95–102.

Lu, Z.; Agarwal, D.; and Dhillon, I. S. 2009. A spatio-temporal approach to collaborative filtering. In *Proc. of the third ACM conf. on Recommender systems*, 13–20.

Porteous, I.; Bart, E.; and Welling, M. 2008. Multi-HDP: A non parametric bayesian model for tensor factorization. In *Proc. of the 23rd National Conf. on Artificial Intelligence*, 1487–1490.

Rasmussen, C. E., and Williams, C. K. I. 2004. *Gaussian processes for machine learning*. Cambridge,MA: MIT Press.

Resnick, P.; Iacovou, N.; Suchak, M.; Bergstrom, P.; and Riedl, J. 1994. Grouplens: An open architecture for collaborative filtering of netnews. In *Proc. of the ACM Conf. on Computer Supported Cooperative Work*, 175–186.

Salakhutdinov, R., and Mnih, A. 2008a. Bayesian probabilistic matrix factorization using Markov Chain Monte Carlo. In *Proc. of the 25th Int'l Conf. on Machine Learning*, 880–887.

Salakhutdinov, R., and Mnih, A. 2008b. Probabilistic matrix factorization. In *Advances in Neural Information Processing Systems*, volume 20, 605–614.

Sarwar, B.; Karypis, G.; Konstan, J.; and Riedl, J. 2001. Item-based collaborative filtering recommendation algorithms. In *Proc. of the 10th Int'l World Wide Web Conf.*, 285–295.

Srebro, N., and Jaakkola, T. 2003. Weighted low-rank approximations. In *Proc. of the 20th Int'l Conf. on Machine Learning*, 720–727.

Wang, J. M.; Fleet, D. J.; and Hertzmann, A. 2008. Gaussian Process Dynamical Models for Human Motion. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 30(2):283–298.

Xiong, L.; Chen, X.; Huang, T.-K.; Schneider, J.; and Carbonell, J. G. 2010. Temporal collaborative filtering with Bayesian probabilistic tensor factorization. In *Proc. of the SIAM Int'l Conf. on Data Mining*, 211–222.