Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence

Dynamic Bayesian Probabilistic Matrix Factorization

Sotirios P. Chatzis
Department of Electrical Engineering, Computer Engineering, and Informatics
Cyprus University of Technology
Limassol 3603, Cyprus
soteri0s@mac.com

Abstract

Collaborative filtering algorithms generally rely on the assumption that user preference patterns remain stationary. However, real-world relational data are seldom stationary. User preference patterns may change over time, giving rise to the requirement of designing collaborative filtering systems capable of detecting and adapting to preference pattern shifts. Motivated by this observation, in this paper we propose a dynamic Bayesian probabilistic matrix factorization model, designed for modeling time-varying distributions. Formulation of our model is based on imposition of a dynamic hierarchical Dirichlet process (dHDP) prior over the space of probabilistic matrix factorization models to capture the time-evolving statistical properties of modeled sequential relational datasets. We develop a simple Markov Chain Monte Carlo sampler to perform inference. We present experimental results to demonstrate the superiority of our temporal model.

Introduction

Ratings-based collaborative filtering (CF) systems have served as an effective approach to address the problem of discovering items of interest (Chen et al. 2009). They are based on the intuitive idea that the preferences of a user can be inferred by exploiting past ratings of that user as well as users with related behavior patterns. This thriving subfield of machine learning has started becoming popular since the late 1990s with the spread of online services that use recommender systems, such as Amazon, Yahoo! Music, MovieLens, Netflix, and citeULike.

The majority of existing CF algorithms employ static models in which relations are assumed to be fixed over time. However, this convenient assumption is rather implausible: user preferences in real-world systems usually evolve over time, exhibiting strong temporal patterns, since they are affected by moods, contexts, and pop culture trends. Therefore, it is crucial that we design CF systems capable of learning to detect preference pattern shifts in the context of real-world applications.

To address this challenge, in this paper we introduce a nonparametric Bayesian dynamic relational data modeling approach based on the Bayesian probabilistic matrix factorization (BPMF) model, a recently proposed model-based collaborative filtering method (Salakhutdinov and Mnih 2008). Our method utilizes a nonparametric Bayesian dynamic model for learning to detect density changes in a dynamic fashion, namely the dynamic hierarchical Dirichlet process (dHDP) (Ren, Carin, and Dunson 2008). The dHDP is developed to model the time-evolving statistical properties of sequential datasets, by linking the statistical properties of data collected at consecutive time points via a random parameter that controls their probabilistic similarity.

The remainder of this paper is organized as follows: At first, we briefly provide the background of our approach: we review existing algorithms for modeling dynamic relational data, and we briefly present the nonparametric Bayesian prior used to formulate our method, namely the dynamic hierarchical Dirichlet process. Subsequently, we introduce our proposed model, and derive its inference and prediction algorithms. Further, we experimentally evaluate our method using benchmark datasets. Finally, we summarize our results, and conclude this paper.

Methodological Background

Existing dynamic modeling approaches

Previous work on modeling dynamic relational data is rather limited. The first ever attempt to model time-evolving preference patterns in the context of CF systems can be traced back to the timeSVD++ model (Koren 2009). The timeSVD++ method assumes that the latent features of singular value decomposition (SVD) (Paterek 2007) consist of some components that are evolving over time and some others that constitute dedicated bias for each user at each specific time point. This model can effectively capture local changes of user preference, thus improving the performance over static algorithms.

Another class of dynamic CF methods are based on a Bayesian probabilistic tensor factorization (BPTF) approach, first proposed in (Xiong et al. 2010). The basic concept of the BPTF approach toward CF in (Xiong et al. 2010), and its recent variants (e.g., (Li et al. 2011b; Pan et al. 2013)), consists in using probabilistic latent factor models to perform learning (similar to, e.g., (Salakhutdinov and Mnih 2007; 2008)), which, in addition to the factors that are used to characterize entities, introduce another set

of latent features for each different time period. Intuitively, these additional factors represent the population-level preference of latent features at each particular time, capturing preference changes due to, e.g., mood variation, change of context, or different pop culture trends. Model formulation is based on CANDECOMP/PARAFAC (CP) decomposition (Kolda and Bader 2009) or Tucker decomposition (Tucker 1966), which generalize matrix SVD to tensors.

Finally, another interesting approach is the recently proposed RMGM-OT method of (Li et al. 2011a). This method uses Bi-LDA, a Bayesian latent factor model for matrix trifactorization (Porteous, Bart, and Welling 2008), to model user-interest drift in the context of CF systems. This method outperformed timeSVD++ in the Netflix dataset.

The dynamic hierarchical Dirichlet process

Dirichlet process (DP) models were first introduced by Ferguson (Ferguson 1973). A DP is characterized by a base distribution G_0 and a positive scalar α, usually referred to as the innovation parameter, and is denoted as $\mathrm{DP}(\alpha, G_0)$. Let us suppose we randomly draw a sample distribution G from a DP, and, subsequently, we independently draw M random variables $\{\boldsymbol{\theta}^*_m\}_{m=1}^M$ from G:

$$G|\alpha, G_0 \sim \mathrm{DP}(\alpha, G_0) \tag{1}$$

$$\boldsymbol{\theta}^*_m|G \sim G, \quad m = 1, \dots M \tag{2}$$

Integrating out G, the joint distribution of the variables $\{\boldsymbol{\theta}^*_m\}_{m=1}^M$ can be shown to exhibit a clustering effect. Specifically, given the first $M-1$ samples of G, $\{\boldsymbol{\theta}^*_m\}_{m=1}^{M-1}$, it can be shown that a new sample $\boldsymbol{\theta}^*_M$ is either (a) drawn from the base distribution G_0 with probability $\frac{\alpha}{\alpha+M-1}$, or (b) is selected from the existing draws, according to a multinomial allocation, with probabilities proportional to the number of the previous draws with the same allocation (Blackwell and MacQueen 1973).

Let $\{\boldsymbol{\theta}_c\}_{c=1}^C$ be the set of distinct values taken by the variables $\{\boldsymbol{\theta}^*_m\}_{m=1}^{M-1}$. A characterization of the (unconditional) distribution of the random variable G drawn from a DP, $\mathrm{DP}(\alpha, G_0)$, is provided by the stick-breaking construction of Sethuraman (Sethuraman 1994). Consider two infinite collections of independent random variables $\boldsymbol{v} = [v_c]_{c=1}^\infty$, $\{\boldsymbol{\theta}_c\}_{c=1}^\infty$, where the v_c are drawn from a Beta distribution, and the $\boldsymbol{\theta}_c$ are independently drawn from the base distribution G_0. The stick-breaking representation of G is then given by

$$G = \sum_{c=1}^{\infty} \varpi_c(\boldsymbol{v}) \delta_{\boldsymbol{\theta}_c} \tag{3}$$

where $\delta_{\boldsymbol{\theta}_c}$ denotes the distribution concentrated at a single point $\boldsymbol{\theta}_c$,

$$p(v_c) = \mathrm{Beta}(1, \alpha) \tag{4}$$

$$\varpi_c(\boldsymbol{v}) = v_c \prod_{j=1}^{c-1}(1 - v_j) \quad \in [0, 1] \tag{5}$$

and $\sum_{c=1}^\infty \varpi_c(\boldsymbol{v}) = 1$.

(Teh et al. 2005) proposed a hierarchical Dirichlet process (HDP) model that allows for linking a set of group-specific Dirichlet processes, learning the model components jointly across multiple groups of data. Specifically, let us assume J groups of data; let the dataset from the jth group be denoted as $\{\boldsymbol{x}_{ji}\}_{i=1}^{N_j}$. An HDP-based model considers that the data from each group are drawn from a distribution with different parameters $\{\boldsymbol{\theta}^*_{ji}\}_{i=1}^{N_j}$, which are in turn drawn from group-specific DPs. In addition, HDP assumes that the base distribution of the group-specific DPs is a common underlying DP. Under this construction, the generative model for the HDP yields

$$\boldsymbol{x}_{ji} \sim F(\boldsymbol{\theta}^*_{ji}) \tag{6}$$

$$\boldsymbol{\theta}^*_{ji} \sim G_j \tag{7}$$

$$G_j \sim \mathrm{DP}(\alpha_j, G_0) \tag{8}$$

$$G_0 \sim \mathrm{DP}(\gamma, H) \tag{9}$$

where $j = 1, \dots, J$, and $i = 1, \dots, N_j$.

In the context of the HDP, different observations that belong to the same group share the same parameters (atoms) that comprise G_j. In addition, observations across different groups might also share parameters (atoms), probably with different mixing probabilities for each DP G_j; this is a consequence of the fact that the DPs G_j pertaining to all the modeled groups share a common base measure G_0, which is also a discrete distribution.

Although the HDP introduces a dependency structure over the modeled groups, it does not account for the fact that, when it comes to modeling of sequential data, especially data the distribution of which changes over time, sharing of underlying atoms from the DPs is more probable among datasets collected closely in time. To allow for such a modeling capacity, a dynamic variant of the hierarchical Dirichlet process has been proposed in (Ren, Carin, and Dunson 2008).

Let us assume J datasets collected sequentially in time, i.e., $\{\boldsymbol{x}_{1i}\}_{i=1}^{N_1}$ is collected first, $\{\boldsymbol{x}_{2i}\}_{i=1}^{N_2}$ is collected second, and so on. To introduce the assumption that datasets collected at adjacent time points are more likely to share underlying patterns, and, hence, the same atoms, (Ren, Carin, and Dunson 2008) proposed the dHDP model, which postulates

$$G_j \sim (1 - \tilde{w}_{j-1}) G_{j-1} + \tilde{w}_{j-1} H_{j-1}, \; j > 1 \tag{10}$$

and $G_1 \sim \mathrm{DP}(\alpha_1, G_0)$, where G_0 is a discrete distribution drawn as given by (9), similar to the HDP model, H_{j-1} is called the *innovation distribution*, and is drawn as

$$H_{j-1} \sim \mathrm{DP}(\alpha_j, G_0) \tag{11}$$

and the weights $\tilde{w}_{j-1}$ are Beta-distributed variables:

$$\tilde{w}_{j-1} \sim \mathrm{Beta}(a_{j-1}, b_{j-1}) \tag{12}$$

Under this construction, the distribution pertaining to the jth dataset, G_j, deviates from the distribution pertaining to the previous dataset, G_{j-1}, by introducing a new innovation distribution H_{j-1}, and the Beta-distributed random variable $\tilde{w}_{j-1}$ that *controls the probability of innovation*. As a result

of this construction, the model encourages sharing between temporally proximate data.

Denoting as $\boldsymbol{\pi}_j$ the weights of the set of the atoms drawn from the innovation distribution H_{j-1}, and introducing the atom indicator variables $\{z_{ji}\}_{j,i}$, and the auxiliary *innovation distribution selection* latent variables $\{\phi_{ji}\}_{j,i}$, dHDP can be also expressed under the following alternative construction:

$$\boldsymbol{x}_{ji}|z_{ji}=k;\{\boldsymbol{\theta}_k\}_{k=1}^{\infty}\sim F(\boldsymbol{\theta}_k) \tag{13}$$

$$z_{ji}|\phi_{ji};\{\boldsymbol{\pi}_h\}_{h=1}^{j}\sim \mathrm{Mult}(\boldsymbol{\pi}_{\phi_{ji}}) \tag{14}$$

$$\boldsymbol{\pi}_j\sim \mathrm{DP}(\alpha_j,G_0) \tag{15}$$

$$G_0\sim \mathrm{DP}(\gamma,H) \tag{16}$$

$$\phi_{ji}|\tilde{\boldsymbol{w}}\sim \mathrm{Mult}(\boldsymbol{w}_j) \tag{17}$$

with $\boldsymbol{w}_j=(w_{jl})_{l=1}^{j}$, and

$$w_{jl}=\tilde{w}_{l-1}\prod_{m=l}^{j-1}(1-\tilde{w}_m),\ l=1,\ldots,j \tag{18}$$

while $\tilde{w}_0=1$, and

$$\tilde{w}_j|a_j,b_j\sim \mathrm{Beta}(\tilde{w}_j|a_j,b_j),\ j=1\ldots.,J-1 \tag{19}$$

Proposed Approach

Let us consider we are given a set of rankings $R=\{r_{ij}^t\}_{i,j,t}$ assigned by a set of users with indices $i\in\{1,\ldots,N\}$ to a set of items with indices $j\in\{1,\ldots,M\}$ at *different time slices,* $t\in\{1,\ldots,T\}$. Let us also consider that the preference patterns of the users may change over time (at different time slices), due to different moods, contexts, or pop culture trends. In addition, we assume that user preferences actually exhibit strong temporal patterns, thus tending to evolve gradually over time. To obtain a model-based CF method capable of incorporating these assumptions into its inference and prediction mechanisms, we proceed as follows.

Let us denote as $R^t=\{r_{ij}^t\}_{i,j=1}^{N,M}$, the set of rankings obtained at time slice $t\in\{1,\ldots,T\}$. We consider

$$p(R^t|U^t,V;\sigma^2)=\prod_{i=1}^{N}\prod_{j=1}^{M}\left[p(r_{ij}^t|U^t,V;\sigma^2)\right]^{I_{ij}^t} \tag{20}$$

where I_{ij}^t is an indicator variable equal to 1 if the ith user rated the jth item at time slice t, 0 otherwise, and U^t is the set of user latent feature vectors at the time slice t. To introduce temporal dynamics into the model, and capture the evolution of the latent feature vectors of the users over time, we impose a dHDP prior over the user latent feature vectors, yielding

$$r_{ij}^t|\ z_i^t=k\sim\mathcal{N}(\boldsymbol{u}_i^k\cdot\boldsymbol{v}_j,\sigma^2) \tag{21}$$

where $\boldsymbol{u}_i^k$ is the latent feature vector of the ith user, if we consider that user i belongs at time t to the kth *group of users,* i.e. $z_i^t=k$. In other words, to model the transitive dynamics of user preference patterns, we consider that each user belongs at each time slice to some *user group*, users change groups over time, and the latent feature vectors of the users belonging to the same group are generated from the same underlying distribution.

Following the assumptions of the imposed dHDP, we subsequently have

$$z_i^t|\phi_i^t;\{\boldsymbol{\pi}_\tau\}_{\tau=1}^{t}\sim\mathrm{Mult}(\boldsymbol{\pi}_{\phi_i^t}) \tag{22}$$

$\boldsymbol{\pi}_\tau=(\pi_{\tau l})_{l=1}^{\infty}$, where, following (Teh et al. 2005), we have

$$\pi_{tl}=\tilde{\pi}_{tl}\prod_{h=1}^{l-1}(1-\tilde{\pi}_{th}) \tag{23}$$

$$\tilde{\pi}_{tl}\sim\mathrm{Beta}(\alpha_t\beta_l,\alpha_t(1-\sum_{m=1}^{l}\beta_m)) \tag{24}$$

$$\beta_k=\varpi_k\prod_{q=1}^{k-1}(1-\varpi_q) \tag{25}$$

$$\varpi_k\sim\mathrm{Beta}(1,\gamma) \tag{26}$$

and the latent (innovation distribution selection) variables ϕ_i^t of the dHDP yield

$$\phi_i^t|\tilde{\boldsymbol{w}}\sim\mathrm{Mult}(\boldsymbol{w}_t) \tag{27}$$

with $\boldsymbol{w}_t=(w_{tl})_{l=1}^{t}$,

$$w_{tl}=\tilde{w}_{l-1}\prod_{m=l}^{t-1}(1-\tilde{w}_m),\ l=1,\ldots,t \tag{28}$$

$$\tilde{w}_t|a_t,b_t\sim\mathrm{Beta}(\tilde{w}_t|a,b),\ t=1\ldots.,T-1 \tag{29}$$

and $\tilde{w}_0=1$. Finally, we also consider

$$p(\boldsymbol{u}_i^k|\boldsymbol{\mu}_U^k,\boldsymbol{\Lambda}_U^k)=\mathcal{N}(\boldsymbol{u}_i^k|\boldsymbol{\mu}_U^k,[\boldsymbol{\Lambda}_U^k]^{-1}) \tag{30}$$

$$p(\boldsymbol{v}_i|\boldsymbol{\mu}_V,\boldsymbol{\Lambda}_V)=\mathcal{N}(\boldsymbol{v}_i|\boldsymbol{\mu}_V,\boldsymbol{\Lambda}_V^{-1}) \tag{31}$$

with

$$p(\boldsymbol{\mu}_U^k,\boldsymbol{\Lambda}_U^k)=\mathcal{NW}(\boldsymbol{\mu}_U^k,\boldsymbol{\Lambda}_U^k|\lambda_U,\boldsymbol{m}_U,\eta_U,\boldsymbol{S}_U) \tag{32}$$

$$p(\boldsymbol{\mu}_V,\boldsymbol{\Lambda}_V)=\mathcal{NW}(\boldsymbol{\mu}_V,\boldsymbol{\Lambda}_V|\lambda_V,\boldsymbol{m}_V,\eta_V,\boldsymbol{S}_V) \tag{33}$$

similar to the standard BPMF model.

This concludes the formulation of our model. We dub our approach the dynamic BPMF (dBPMF) model. Inference for our model can be efficiently performed by means of MCMC.

Inference algorithm

To efficiently perform inference for the dBPMF model, we devise a variant of the block Gibbs sampler in (Ishwaran and James 2001). To make inference tractable, we use a truncated expression of the stick-breaking representation of the underlying shared DP of our model, G_0. In other words, we set a truncation threshold C, and consider $\boldsymbol{\pi}_t=(\pi_{tl})_{l=1}^{C},\ \forall t$ (Ishwaran and James 2001). A large value of C allows for obtaining a good approximation of the infinite stick-breaking process, since in practice the π_{tl} are expected to diminish quickly with increasing $l,\ \forall t$ (Ishwaran and James 2001).

Let us begin with the expression of the conditional distribution of $\tilde{w}_t$, for $t = 1, \ldots, T-1$; we have

$$p(\tilde{w}_t|\ldots) = \text{Beta}(\tilde{w}_t|a + \sum_{j=t+1}^{T} n_{j,t+1}, b + \sum_{j=t+1}^{T}\sum_{h=1}^{t} n_{jh}) \tag{34}$$

where $n_{th} = \sum_{i=1}^{N} \delta(\phi_i^t = h)$, and $\delta(\cdot)$ is the Kronecker delta function. Similar, the conditional posterior of $\tilde{\pi}_{tl}$, $l = 1, \ldots, C$, yields

$$p(\tilde{\pi}_{tl}|\ldots) = \text{Beta}\big(\tilde{\pi}_{tl}|\alpha_t\beta_l + \sum_{j=1}^{T}\sum_{i=1}^{N}\delta(\phi_i^j = t)\delta(z_i^j = l), \alpha_t(1 - \sum_{m=1}^{l}\beta_m) + \sum_{j=1}^{T}\sum_{i=1}^{N}\sum_{k=l+1}^{C}\delta(\phi_i^j = t)\delta(z_i^j = k)\big) \tag{35}$$

The updates of the set of indicator variables $\{\phi_i^t\}_{i,t=1}^{N,T}$ can be obtained by generating samples from multinomial distributions with entries of the form

$$p(\phi_i^t = \tau|\ldots) \propto \tilde{w}_{\tau-1}\prod_{m=\tau}^{t-1}(1-\tilde{w}_m)\tilde{\pi}_{\tau z_i^t}\prod_{q=1}^{z_i^t-1}(1-\tilde{\pi}_{\tau q}) \times \prod_{j=1}^{M}\Big(\mathcal{N}(r_{ij}^t|\boldsymbol{u}_i^{z_i^t}\cdot\boldsymbol{v}_j, \sigma^2)\Big)^{I_{ij}^t}, \tau = 1, \ldots, t \tag{36}$$

Similar, the set of indicator variables $\{z_i^t\}_{i,t=1}^{N,T}$ are obtained by generating samples from multinomial distributions with entries of the form

$$p(z_i^t = k|\ldots) \propto \tilde{\pi}_{\phi_i^t k}\prod_{q=1}^{k-1}(1-\tilde{\pi}_{\phi_i^t q}) \times \prod_{j=1}^{M}\big(\mathcal{N}(r_{ij}^t|\boldsymbol{u}_i^k\cdot\boldsymbol{v}_j, \sigma^2)\big)^{I_{ij}^t}, k = 1, \ldots, C \tag{37}$$

The time-evolving latent factor vectors of the users, $\boldsymbol{u}_i^k$, read

$$p(\boldsymbol{u}_i^k|\ldots) = \mathcal{N}(\boldsymbol{u}_i^k|\boldsymbol{\mu}_i^k, [\boldsymbol{\Lambda}_i^k]^{-1}) \tag{38}$$

where

$$\boldsymbol{\Lambda}_i^k = \boldsymbol{\Lambda}_U^k + \frac{1}{\sigma^2}\sum_{t=1}^{T}\delta(z_i^t = k)\sum_{j=1}^{M}[\boldsymbol{v}_j\boldsymbol{v}_j^T]^{I_{ij}^t} \tag{39}$$

and

$$\boldsymbol{\mu}_i^k = [\boldsymbol{\Lambda}_i^k]^{-1}\left(\boldsymbol{\Lambda}_U^k\boldsymbol{\mu}_U^k + \frac{1}{\sigma^2}\sum_{t=1}^{T}\delta(z_i^t = k)\sum_{j=1}^{M}[\boldsymbol{v}_j r_{ij}^t]^{I_{ij}^t}\right) \tag{40}$$

with the conditional posteriors of the corresponding model hyperparameters yielding

$$p(\boldsymbol{\mu}_U^k, \boldsymbol{\Lambda}_U^k|\ldots) = \mathcal{NW}(\boldsymbol{\mu}_U^k, \boldsymbol{\Lambda}_U^k|\tilde{\lambda}_U^k, \tilde{\boldsymbol{m}}_U^k, \tilde{\eta}_U^k, \tilde{\boldsymbol{S}}_U^k) \tag{41}$$

where

$$\tilde{\boldsymbol{m}}_U^k = \frac{\lambda_U\boldsymbol{m}_U + \sum_{i=1}^{N}\boldsymbol{u}_i^k}{\lambda_U + N} \tag{42}$$

$$\tilde{\lambda}_U^k = \lambda_U + N \tag{43}$$

$$\tilde{\eta}_U^k = \eta_U + N \tag{44}$$

and, denoting $\bar{\boldsymbol{u}}^k = \frac{1}{N}\sum_{i=1}^{N}\boldsymbol{u}_i^k$ we have

$$\big[\tilde{\boldsymbol{S}}_U^k\big]^{-1} = \boldsymbol{S}_U^{-1} + \sum_{i=1}^{N}\left(\boldsymbol{u}_i^k - \bar{\boldsymbol{u}}^k\right)\left(\boldsymbol{u}_i^k - \bar{\boldsymbol{u}}^k\right)^T + \frac{\lambda_U N}{\lambda_U + N}\left(\boldsymbol{m}_U - \bar{\boldsymbol{u}}^k\right)\left(\boldsymbol{m}_U - \bar{\boldsymbol{u}}^k\right)^T \tag{45}$$

Finally, the latent factor vectors of the items, $\boldsymbol{v}_j$, yield

$$p(\boldsymbol{v}_j|\ldots) = \mathcal{N}(\boldsymbol{v}_j|\boldsymbol{\mu}_j^*, [\boldsymbol{\Lambda}_j^*]^{-1}) \tag{46}$$

where

$$\boldsymbol{\Lambda}_j^* = \boldsymbol{\Lambda}_V + \frac{1}{\sigma^2}\sum_{t=1}^{T}\sum_{k=1}^{C}\sum_{i=1}^{N}\delta(z_i^t = k)[\boldsymbol{u}_i^k(\boldsymbol{u}_i^k)^T]^{I_{ij}^t} \tag{47}$$

and

$$\boldsymbol{\mu}_j^* = [\boldsymbol{\Lambda}_j^*]^{-1} \times \left(\boldsymbol{\Lambda}_V\boldsymbol{\mu}_V + \frac{1}{\sigma^2}\sum_{t=1}^{T}\sum_{k=1}^{C}\sum_{i=1}^{N}\delta(z_i^t = k)[\boldsymbol{u}_i^k r_{ij}^t]^{I_{ij}^t}\right) \tag{48}$$

with the conditional posteriors of the corresponding model hyperparameters yielding

$$p(\boldsymbol{\mu}_V, \boldsymbol{\Lambda}_V|\ldots) = \mathcal{NW}(\boldsymbol{\mu}_V, \boldsymbol{\Lambda}_V|\tilde{\lambda}_V, \tilde{\boldsymbol{m}}_V, \tilde{\eta}_V, \tilde{\boldsymbol{S}}_V) \tag{49}$$

where

$$\tilde{\boldsymbol{m}}_V = \frac{\lambda_V\boldsymbol{m}_V + \sum_{j=1}^{M}\boldsymbol{v}_j}{\lambda_V + M} \tag{50}$$

$$\tilde{\lambda}_V = \lambda_V + M \tag{51}$$

$$\tilde{\eta}_V = \eta_V + M \tag{52}$$

and, denoting $\bar{\boldsymbol{v}} = \frac{1}{M}\sum_{j=1}^{M}\boldsymbol{v}_j$, we have

$$\Big[\tilde{\boldsymbol{S}}_V\Big]^{-1} = \boldsymbol{S}_V^{-1} + \sum_{j=1}^{M}\left(\boldsymbol{v}_j - \bar{\boldsymbol{v}}\right)\left(\boldsymbol{v}_j - \bar{\boldsymbol{v}}\right)^T + \frac{\lambda_V M}{\lambda_V + M}\left(\boldsymbol{m}_V - \bar{\boldsymbol{v}}\right)\left(\boldsymbol{m}_V - \bar{\boldsymbol{v}}\right)^T \tag{53}$$

Prediction

Having derived the inference algorithm of our model, we now proceed to derivation of its prediction algorithm. This consists in using the trained model to estimate the unknown rating a user with index $i \in \{1, \ldots, N\}$ would have assigned to an item with index $j \in \{1, \ldots, M\}$ at some time slice $t \in \{1, \ldots, T\}$, if they had rated it at that time slice. For this purpose, we resort to an MCMC-based approximation: We use block Gibbs sampling to generate multiple samples from the posterior distributions over the model parameters and hyperparameters. Using these samples, the generated prediction is approximated as

$$\hat{r}_{ij}^t \approx \frac{1}{\Xi}\sum_{\xi=1}^{\Xi}\boldsymbol{u}_i^{(\xi)}\cdot\boldsymbol{v}_j^{(\xi)} \tag{54}$$

where Ξ is the number of drawn samples, and $[\cdot]^{(\xi)}$ is the ξth drawn sample.

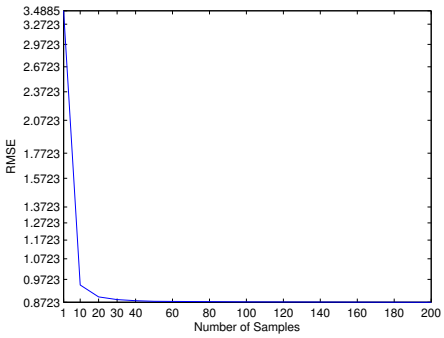


Figure 2: dBPMF convergence.

Experimental Evaluation

To assess the efficacy of our approach, we experiment with the Netflix dataset[1]. This dataset comprises users ratings to various movies, given on a 5-star scale. To obtain some comparative results, we also evaluate recently proposed dynamic relational data modeling approaches, namely the tensor factorization-based BPTF method (Xiong et al. 2010), and the Bi-LDA-based RMGM-OT method (Li et al. 2011a). We also compare to a related state-of-the-art static modeling method, namely BPMF (Salakhutdinov and Mnih 2008). We run our experiments as a single-threaded MATLAB process on a 2.53 GHz Intel Core 2 Duo CPU. In our experiments, we use source code provided by the authors of the BPTF (Xiong et al. 2010) and RMGM-OT (Li et al. 2011a) papers.

For all models, prediction results are clipped to fit in the interval $[1, 5]$. BPMF is initialized with $\lambda_U = \lambda_V = 1$, $\boldsymbol{m}_U = \boldsymbol{m}_V = 0$, $\eta_U = \eta_V = 30$, $\boldsymbol{S}_U = \boldsymbol{S}_V = \boldsymbol{I}$, similar to (Salakhutdinov and Mnih 2008). Similar hyperparameter values are selected for our model, to ensure fairness in our comparisons. For the RMGM-OT method, we heuristically determined that using 20 user and item groups, as also suggested in (Li et al. 2011a), and preset in the source code provided by the authors, yields optimal results. Finally, for the BPTF method, hyperparameter selection is based on the suggestions of (Xiong et al. 2010). Convergence of the Gibbs sampler (wherever applicable) is diagnosed by monitoring the behavior of the Frobenius norms of the sampled model parameters and hyperparameters. Performance is evaluated on the grounds of the root mean square error (RMSE) metric, a standard evaluation metric in CF literature.

Experimental Setup

Netflix dataset contains $100,480,507$ ratings from $N = 480,189$ users to $M = 17,770$ movies rated between 1999 and 2005. Apart from the training set, the Netflix dataset also provides a probe set which comprises $1,408,395$ uniformly selected users. Time information is provided in days. The timestamps we use in our experiments correspond to calendar months. However, the ratings in the early months are much more scarce than the later months. For this reason, similar to (Xiong et al. 2010), we aggregate several earlier months together so that every time slice contains an approximately equal number of ratings. This way, we eventually yield 27 time slices for the entire dataset.

We perform experiments on a subset of the Netflix data constructed by randomly selecting 20% of the users and 20% of the movies, similar to (Xiong et al. 2010). From the resulting dataset, we randomly select 30% of the ratings, and no more than 10 ratings in any case, from each user as the test set, and use the rest for training. This procedure is similar to the way Netflix Prize created their test set. We perform 10-fold cross-validation to alleviate the effects of random selection of training and test samples on the obtained performance measurements.

Results

In Fig. 1a, we illustrate how performance changes for the BPMF, BPTF, and dBPMF models by varying the number of latent features. These results are averages (mean RMSE) over the conducted 10 folds of our experiment. As we observe, in general, all models yield an RMSE decrease as the number of factors increases. We also note that increasing the latent features to more than 50 does not result in substantial further performance improvements for the dynamic data modeling methods; in contrast, BPMF performance continues to yield some improvement. This fact indicates that the lack of temporal structure extraction mechanisms in the context of static models results in them relying on the derived latent subspace representations to make up for this inadequacy.

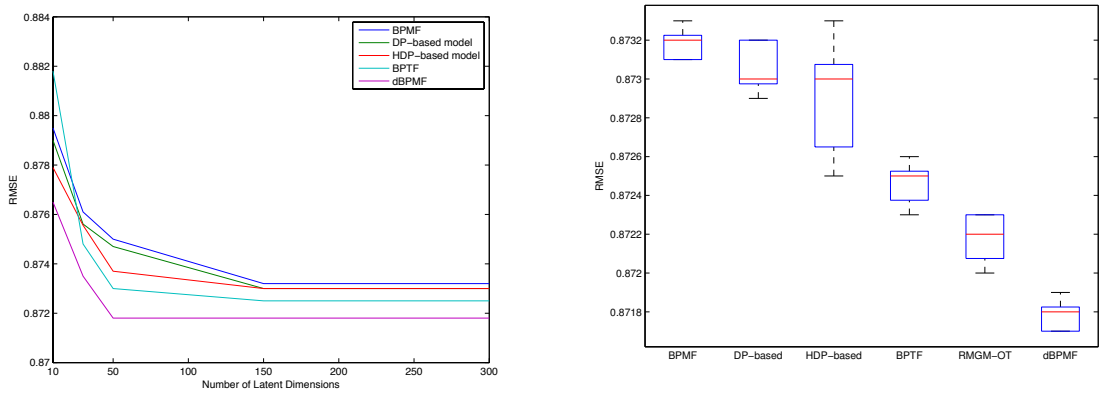
In Fig. 1b, we also illustrate in detail the performances of all the evaluated algorithms for *optimal latent space dimensionality* (wherever applicable), as determined from the previous illustration, in the form of box plots. As we observe, our algorithm works better than the competition, including the related BPTF method. Specifically, our method outperforms BPTF both in terms of median performance, and the 25th and 75th percentiles of the set of the obtained performances. Further, we observe that RMGM-OT obtains a statistically significant improvement over the other methods, being the second best performing method. However, it remains inferior to dBPMF.

Finally, to confirm the statistical significance of our findings, we run the Student's-t test on the pairs of performances (for optimal numbers of latent features) of our method and all its considered competitors. As we observe, the Student's-t test yields p-values ranging from $0.47 * 10^{-16}$, obtained when we compare our method to RMGM-OT, to $0.1 * 10^{-16}$, obtained when we compare to BPMF. Given that p-values below 10^{-2} indicate statistically significant differences, we deduce that our results are strongly statistically significant in all cases.

Further investigation

Following the definition of our model and the provided encouraging experimental results, a question that naturally arises is whether the observed advantages of our approach do actually stem from the dynamic nature of the clustering mechanism of the dHDP prior, or application of a simpler clustering mechanism could also yield similar benefits in

[1] http://archive.ics.uci.edu/ml/datasets/Netflix+Prize



(a) (b)

Figure 1: Netflix dataset: (a) Performance fluctuation as a function of latent dimensionality. (b) Box plot of the obtained RMSEs for optimal number of latent features.

terms of the obtained predictive performance. Indeed, based on the properties of the dHDP (Ren, Carin, and Dunson 2008), if we set $\tilde{w}_j = 0, \forall j$, we straightforwardly obtain that our dHDP-based model reduces to a simple Dirichlet process mixture model, while if we set $\tilde{w}_j = 1, \forall j$, our model reduces to a simpler HDP-based model. Such model variants do also entail parameter sharing, but not in a temporally coherent manner. Therefore, a question that naturally arises is: How would a DP- or HDP-based model perform compared to the full dHDP-based model proposed in this work?

To answer this question, we repeat our previous experiments, setting $\tilde{w}_j = 0, \forall j$, and $\tilde{w}_j = 1, \forall j$, respectively, for varying numbers of latent features. The obtained results are also provided in Figs. 1a and 1b (lines "DP-based model," and "HDP-based model," respectively). As we observe, the performance of these reduced versions of our model (lacking temporal dynamics in their parameter sharing mechanisms), is much inferior to the performance of our full-fledged method. Indeed, it seems that for $\tilde{w}_j = 0, \forall j$, i.e. considering a DP-based reduced version of our model, we yield performance almost identical to BPMF, while for $\tilde{w}_j = 1, \forall j$, i.e. considering an HDP-based variant of our model, performance is better than BPMF only in cases of low latent space dimensionality, and always inferior to dynamic methods. Therefore, retaining the full functionality of the dHDP, and, hence, extracting the temporal dynamics in the modeled data, is crucial for model performance.

Finally, in Fig. 2 we illustrate the convergence of our algorithm as a function of the number of drawn samples. We observe that our algorithm starts from a bad model estimate, due to its random initialization, but converges in a fast pace.

Computational complexity

To conclude, we provide an indicative comparison of computational costs. As we observed, generation of one sample from the BPMF model required less than 6 minutes time and 4GB RAM, while BPTF and RMGM-OT took about 8 minutes and 5GB RAM. In comparison, in our unoptimized MATLAB implementation, our method required about 9 minutes time and 5GB RAM. Turning to prediction generation, which is what actually matters in a real-world system, once the required number of model samples are obtained and stored, our model can generate predictions instantaneously, with no time overhead compared to the competition.

Conclusions

In this work, we presented a nonparametric Bayesian approach for modeling relational data the distributions of which evolve in a dynamic fashion. Formulation of our method is based on an extension of the probabilistic matrix factorization framework under a Bayesian model treatment. To allow for effectively capturing preference pattern shifts in a dynamic fashion, we imposed a dynamic hierarchical Dirichlet process prior over the latent feature vectors of the users. We evaluated our method using the Netflix datasets. We observed that our method both outperforms existing dynamic relational data modeling methods, as well as state-of-the-art static modeling approaches, without scalability sacrifices.

As we explained, our method reduces to simpler DP- or HDP-based models, i.e., models that do not entail temporal dynamics, by setting $\tilde{w}_j = 0, \forall j$, and $\tilde{w}_j = 1, \forall j$, respectively. As we showed, under such a setup, the obtained performance is significantly inferior to the full dBPMF model. Therefore, we deduce that the advantages of our model are a result of its dynamic nature, and could not be obtained by alternative clustering-based models that do not entail temporal dynamics in their clustering mechanisms.

Our future goal is to explore more efficient inference algorithms for our model under a deterministic framework, e.g., variational Bayes (Blei and Jordan 2006; Wang and Blei 2012) or expectation-propagation (Minka 2001).

References

Blackwell, D., and MacQueen, J. 1973. Ferguson distributions via Pólya urn schemes. *The Annals of Statistics* 1(2):353–355.

Blei, D. M., and Jordan, M. I. 2006. Variational inference for Dirichlet process mixtures. *Bayesian Analysis* 1(1):121–144.

Chen, W. Y.; Chu, J. C.; Luan, J.; Bai, H.; Wang, Y.; and Chang, E. Y. 2009. Collaborative filtering for Orkut communities: discovery of user latent behavior. In *Proc. WWW'09*, 681–690.

Ferguson, T. 1973. A Bayesian analysis of some nonparametric problems. *The Annals of Statistics* 1:209–230.

Ishwaran, H., and James, L. F. 2001. Gibbs sampling methods for stick-breaking priors. *Journal of the American Statistical Association* 96:161–173.

Kolda, T. G., and Bader, B. W. 2009. Tensor decompositions and applications. *SIAM Review* 51(3):455–500.

Koren, Y. 2009. Collaborative filtering with temporal dynamics. In *Proc. KDD-09*.

Li, B.; Zhu, X.; Li, R.; Zhang, C.; Xue, X.; and Wu, X. 2011a. Cross-domain collaborative filtering over time. In *Proc. 22nd AAAI*, 2293–2298.

Li, R.; Li, B.; Jin, C.; Xue, X.; and Zhu, X. 2011b. Tracking user-preference varying speed in collaborative filtering. In *Proc. 25th AAAI*.

Minka, T. 2001. Expectation Propagation for Approximate Bayesian Inference. In *Proc. UAI*.

Pan, J.; Ma, Z.; Pang, Y.; and Yuan, Y. 2013. Robust probabilistic tensor analysis for time-variant collaborative filtering. *Neurocomputing* 119(7):139 – 143.

Paterek, A. 2007. Improving regularized singular value decomposition for collaborative filtering. In *In KDDCup '07*.

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

Ren, L.; Carin, L.; and Dunson, D. B. 2008. The dynamic hierarchical Dirichlet process. In *Proc. International Conference on Machine Learning (ICML)*.

Salakhutdinov, R., and Mnih, A. 2007. Probabilistic matrix factorization. In *Proc. NIPS*.

Salakhutdinov, R., and Mnih, A. 2008. Bayesian probabilistic matrix factorization using Markov chain monte carlo. In *Proc. ICML'08*.

Sethuraman, J. 1994. A constructive definition of the Dirichlet prior. *Statistica Sinica* 2:639–650.

Teh, Y. W.; Jordan, M. I.; Beal, M. J.; and Blei, D. M. 2005. Hierarchical Dirichlet processes. Technical report, Dept. of Computer Science, National University of Singapore.

Tucker, L. R. 1966. Some mathematical notes on three-mode factor analysis. *Psychometrika* 31(3):279–311.

Wang, C., and Blei, D. M. 2012. Truncation-free Stochastic Variational Inference for Bayesian Nonparametric Models. In *Pro. NIPS*.

Xiong, L.; Chen, X.; Huang, T.-K.; Schneider, J.; and Carbonell, J. G. 2010. Temporal collaborative filtering with Bayesian probabilistic tensor factorization. In *Proc. SDM*.