Modeling Temporal Adoptions Using Dynamic Matrix Factorization

Freddy Chong Tat Chua, Richard J. Oentaryo, Ee-Peng Lim
Living Analytics Research Centre
Singapore Management University
{freddy.chua.2009, roentaryo, eplim}@smu.edu.sg

***Abstract*—The problem of recommending items to users is relevant to many applications and the problem has often been solved using methods developed from Collaborative Filtering (CF). Collaborative Filtering model-based methods such as Matrix Factorization have been shown to produce good results for static rating-type data, but have not been applied to time-stamped item adoption data. In this paper, we adopted a Dynamic Matrix Factorization (DMF) technique to derive different temporal factorization models that can predict missing adoptions at different time steps in the users' adoption history. This DMF technique is an extension of the Non-negative Matrix Factorization (NMF) based on the well-known class of models called Linear Dynamical Systems (LDS). By evaluating our proposed models against NMF and TimeSVD++ on two real datasets extracted from ACM Digital Library and DBLP, we show empirically that DMF can predict adoptions more accurately than the NMF for several prediction tasks as well as outperforming TimeSVD++ in some of the prediction tasks. We further illustrate the ability of DMF to discover evolving research interests for a few author examples.**

***Keywords*-Kalman Filter, Linear Dynamical Systems, State Space Models, Dynamic Matrix Factorization**

I. Introduction

A. Motivation

Recommender systems have been widely used to suggest products, content and services to consumers. Recommender techniques have been largely related to rating prediction and evaluated on Netflix and Movielens datasets. The common assumptions underlying rating prediction are that: (a) each item can be rated or adopted only once by a user; (b) ratings assigned to items are restricted to a pre-defined rating options, say 1 to 5; and (c) the user-rate-item data is static. Although these assumptions are reasonable in many application settings, there are also many other settings that violate these assumptions.

For example, there are many application scenarios where a user can adopt the same item more than once, i.e. a user may buy the same product in different purchases. These include food, stationery, drug, and other items. A user may visit the same restaurant, bookstore, or cinema multiple times. In the context of social media, a user may adopt the same URL, tag or keyword multiple times as the user shares messages with her friends. When the same item is adopted at different time steps, the user may adopt it with different quantities as the user's preference or demand on the item changes over time. Assumption (a) hence does not hold in these scenarios and we need to consider recommending the same item even if it has been previously adopted.

The above scenarios also violate assumption (b) as they do not necessarily involve users giving ratings to items. The user's propensity to adopt an item can be measured by adoption quantity, which can be any non-negative integer value instead of a fixed range of rating values. A user may choose not to adopt an item at all if he dislikes the item, or adopt an item with a large quantity if he likes it. Adoption count also does not imply likeness. By adopting one instance of item does not mean the user does not like the item. Conversely, by adopting multiple instances of an item does not mean the user like the item.

The last assumption (c) is clearly not applicable to many recommender systems involving dynamic user adoption patterns. These recommender systems have to determine trends that affect user adoptions. Unfortunately, most existing recommendation algorithms only deal with static adoption data. When applied to dynamic adoption data, the data is usually first divided into time steps and the recommendation algorithm is applied to the adoption data in each time step independently of other time step. The result is that items recommended to a user in one time step may look entirely different from those recommended in the next time step, which is not ideal in many application settings.

Figure 1 shows an example between the differences of rating and adoption in two time steps $t = 1, 2$. In the case of temporal rating, the user can only rate an item once, any changes in the rating between the user and the item at a later time step is seen as an updated rating. When we collapse the data into its static equivalent (denoted by *), the value between user and item reflects the latest rating. However for temporal adoption, the edges in the collapsed static data (*) has weights that are aggregated through time.

B. Research Objectives

In this paper, we focus on addressing the problem of modeling users adopting items across different time steps to generate recommendations considering evolving user preferences. Unlike rating-based recommendation, we assume the same users can adopt items more than once with different quantity numbers.

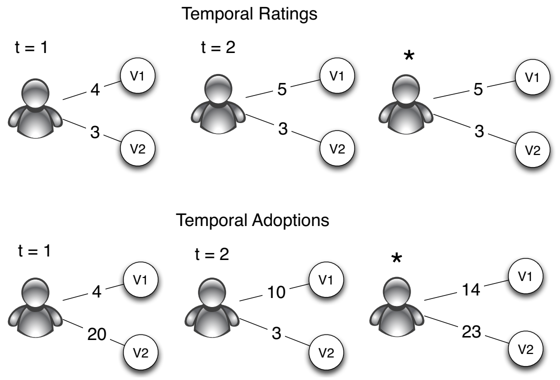


Figure 1. Temporal Rating vs Temporal Adoption

The main idea of our approach is to model dynamic adoption data using a combination of *Non-negative Matrix Factorization* (NMF) and *Linear Dynamical Systems* (LDS). We represent the adoption data of each time step as a state defined by the preferences of users and the characteristics of items in low rank factors as well as transitions of low rank factors so as to smoothen the evolution of user preferences.

Suppose we model users adopting items as a bipartite graph where users and items represent the two types of vertices, the weights on the user-adopt-item edges represent the number of times the users adopt the items. For the time steps of adoption data, we can define a bipartite graph Y_t for each time step t. When a user n adopts w instances of an item m in time t, an edge is created between n and m and an edge weight w is assigned. In the adjacency matrix representation, this translates to $y_{m,n,t} = w$.

We now define the *dynamic adoption prediction problem* as follows. Given the item adoption data for a set of N users and M items in different time steps for $t = 1 \cdots T$, we want to find the low rank factors of user preference for every time step and to use the low rank factors to predict for the possibility of missing adoptions in each time step t.

A direct and simple way of solving dynamic adoption prediction problem is to perform NMF independently for each time step. Suppose we have M items and N users, using MF for K latent factors,

$$Y_t = C_t \cdot X_t$$

where $Y_t \in \mathbf{R}^{M \times N}$, $C_t \in \mathbf{R}^{M \times K}$ and $X_t \in \mathbf{R}^{K \times N}$.

But there are drawbacks to such an approach. Given that solving for NMF is a non-convex optimization problem, this approach suffers from the identifiability problem where multiple solutions exist. This makes the interpretation of resultant predictions difficult, as the lower rank factors are not related across different time steps. That is, the user preference factors derived for one time step may be completely unrelated with those for another time step.

Linear Dynamical Systems (LDS) offer an elegant way of expressing the relationship between latent factors at different time steps. For each user n, LDS derives for each time step t a dynamics matrix $A_{n,t}$ that represents the mapping of latent factors from time step $t-1$ to t. When LDS is applied to a set of users, we obtain Dynamic Matrix Factorization (DMF). Different matrix factorization techniques can be utilized in DMF and this paper introduces DMF based on NMF, a MF technique very often used for rating prediction. To the best of our knowledge, using LDS and NMF for DMF to model dynamic adoption data is novel and has not been attempted before. In a previous work by Sun et al., a Dynamic Matrix Factorization approach based on LDS has been developed for rating prediction [1], but they do not include the use of NMF. The use of NMF is extremely important for obtaining insights into the "topics" that users follow. Without the non-negativity constraints, the latent factors obtained for users become uninterpretable. Previous works on rating prediction [2], [3], [4], [5], [6] which employ Probabilistic Matrix Factorization (PMF) [7], [8] are not able to show interpretable topics because of the unconstrained sign of their latent factors. Our approach of enforcing a non-negativity constraint in DMF has never been applied and evaluated in item adoption prediction.

C. On the Necessity of Non-negativity

We briefly argue that non-negativity is necessary for ranking items in each latent factor to obtain interpretable topics. For a given element $y_{m,n}$ of the item-user matrix Y, the MF approach is to approximate $y_{m,n}$ using the item latent factors c_m and user latent factors x_n.

$$y_{m,n} = \sum_{k=1}^{K} c_{m,k} \cdot x_{k,n}$$

In topic models based on NMF [9], [10], the important items for each latent factor is obtained by ranking the items' value in the respective latent factor. So if $c_{i,k} > c_{j,k}$, it implies that item i is more representative than item j for the latent factor k in NMF. But this is not true if the latent factors contain negative values. A negative $c_{m,k}$ can also be important for contributing to the value $y_{m,n}$ if the corresponding $x_{k,n}$ is also negative. Since the negative latent factors prevent one from interpreting their semantics, we propose to use NMF with LDS to obtain DMF with non-negative values.

D. Contributions

We now summarize the research contributions of this paper as follows:

- This paper makes a clear distinction between item adoption recommendation and rating recommendation. We point out that as user interests evolve, we need to model these changes and adapt the prediction of adoption data temporally.

- We propose Dynamic Matrix Factorization (DMF) based on Non-Negative Matrix Factorization (NMF) and Linear Dynamical Systems (LDS), and apply it to solving several prediction tasks involving adoptions at different time steps as well cumulated adoptions across multiple time steps. We also derive a few variants of DMF based on the choice of item factor scaling and dynamics matrix and show how they can be used in the different adoption prediction tasks.
- We propose three evaluation tasks for comparing the performance of our proposed models against other baselines in the temporal item adoption problem.
- We conduct a series of experiments to show that our proposed models outperform NMF in the different prediction tasks and TimeSVD++ for some prediction tasks involving dynamic adoptions. A few author case examples illustrating changes of research interests learnt in DMF have also been given to highlight the knowledge discovered by using DMF.

II. Related Works

Matrix Factorization (MF) has been successful in solving the (rating) recommendation problem by representing users and items using low rank vectors. One well-known MF approach is Non-negative Matrix Factorization (NMF) [11], [12] which have been proposed to model image pixels and encoding variables, as well as documents and words. Our survey showed that NMF has not been used for adoption recommendation where a user can adopt items with quantities at different points in time. In the item adoption scenario, it is important to address the abundance of temporal data. NMF alone is not adequate to address the relationship between the latent factors between different time points.

In the context of rating recommendation, various MF-based approaches have been proposed, which broadly fall into two categories: *static* and *dynamic* methods. As an example of *static* MF methods, Koren [3] developed a hybrid MF approach that smoothly integrates the latent factor and neighborhood models in order to effectively capture the global and local structure of user and/or item relationships, respectively. Meanwhile, Salakhutdinov and Mnih introduced the first probabilistic linear model of matrix factorization with Gaussian observation noise [7], and later extended it by providing a full Bayesian treatment through the Markov Chain Monte Carlo (MCMC) algorithm [8]. These methods produce good predictive accuracy and can scale up to large/sparse static data. However, they do not consider temporal dynamics, and thus lacks the ability to track the trending patterns that are more relevant to the current user preferences than those in the past.

In light of this limitation, several *dynamic* MF methods have been developed. In [13], an online NMF (ONMF) was proposed that could process the ratings one at a time and automatically update the latent factors by combining the old factors with the newly arrived rating. Koren developed TimeSVD++ to address temporal dynamics through a specific parameterization with factors drifting from a central time [5], [4]. Recently, [6] presented a probabilistic tensor factorization, which extends [7], [8] to model time-evolving relational data. In [14], an evolutionary nonnegative matrix factorization (eNMF) was devised, which assumes factorized matrices evolve smoothly over time, and uses an efficient projected gradient algorithm to minimize the difference between the matrices at consecutive time steps. But none of these proposed temporal MF models make use of the well-known Kalman Filtering and Rauch Tung Striebel (RTS) smoothing algorithm which gives globally optimal solution for the latent states.

Another model was developed in [15] that uses low-rank MF and Kalman filter to estimate user and item factors. This provides a single joint model to simultaneously incorporate both spatial and temporal structure in ratings. But [15] does not model the dynamics of transition between latent factors in consecutive time steps.

Our work is closely related to the recent dynamic MF approach advocated in [1]. The centerpiece of this work is a dynamic state-space model that builds upon probabilistic matrix factorization in [7], [8] and Kalman filter in order to provide recommendations in the presence of process and measurement noises. We combine the use of NMF for the non-negative parameter estimation and proposed different variants of DMF for different adoption scenarios.

In summary, we show that Linear Dynamical Systems (LDS) which was extended to Dynamic Matrix Factorization (DMF) by [1], has parameters that can first be solved by NMF to satisfy the non-negativity constraints. We use NMF for obtaining the item latent factor matrix and initial values of the user latent factors. Then we apply Kalman filtering and RTS smoothing for each individual user to obtain better estimates of their latent factors.

III. Dynamic Matrix Factorization

Given the relationship between static Matrix Factorization (**MF**) and Dynamic Matrix Factorization (**DMF**), we show how to use the parameters obtained from the learning of MF for learning DMF.

A. Problem Definition

Dynamic adoption prediction can be formally defined as a MF problem for an *adoption matrix* $Y \in \mathbf{R}^{M \times N \times T}$, where M denotes the number of items, N denotes the number of users and T denotes the number of time steps. Each element $y_{m,n,t}$ of Y denotes the adoption count (≥ 0) for item m by user n in time step t.

Not all temporal adoptions in Y are observed. We denote $y_{m,n,t}$ as the temporal adoption observed for user n, item m and time step t. When $y_{m,n,t} = 0$, it means that the temporal adoption is *missing* or we do not observe the item

n adopted by user m in the corresponding time step. Y is sparse as each user adopts usually only very few items.

The adoption matrix Y can be collapsed into a $M \times N$ *total adoption matrix* Y^* by aggregating the temporal adoptions of each user-item pair across all time steps. That is, each element of Y^* is obtained by $y^*_{m,n} = \sum_t y_{m,n,t}$.

Depending on what we want to predict for the adoption matrix Y, we can formulate three prediction tasks:

- Task 1, *Prediction of missing temporal adoptions*: The task of predicting missing adoptions at some time step t for some user n and item m and we represent the predicted adoptions by $\hat{y}_{m,n,t}$.
- Task 2, *Prediction of all total adoptions given missing temporal adoptions*: The task of predicting all total adoptions $y^*_{m,n}$ given some missing temporal adoptions, i.e., $y_{m,n,t} = 0$, **at some time step** t for some n and m for a set of (m, n, t) triplets. We represent the predicted total adoptions of user n and item m as $\hat{y}^*_{m,n}$
- Task 3, *Prediction of missing total adoptions*: The task of predicting missing total adoptions $\hat{y}^*_{m,n}$ for user n and item m with $y_{m,n,t} = 0$ **for all** $t \in T$.

We can solve the above prediction tasks in a naive approach using non-negative matrix factorization (NMF) in the next section before extending it to DMF.

B. Non-Negative Matrix Factorization

Given a static adoption matrix $Y^* \in \mathbf{R}^{M \times N}$, matrix factorization returns two lower ranked matrices item-factor matrix $C \in \mathbf{R}^{M \times K}$ and user-factor matrix $X^* \in \mathbf{R}^{K \times N}$, where K represents the number of factors. The *item-factor matrix* C represents the mappings from items to a set of factors, while the *factor-user matrix* X^* represents the mappings from factors to users. In adoption prediction tasks, we would like to regard the factor values as their weights and hence require them to be non-negative.

Non-negative matrix factorization (NMF) meets the requirement for non-negativity of both the item-factor matrix C and factor-user matrix X^*. NMF finds the lower rank matrices C and X^* such that their product recovers missing values in Y^*. As NMF is a well-defined and well-understood technique, we only briefly show how to solve for C and X^* using stochastic gradient descent (SGD) with the log-barrier approach for non-negativity constraints.

The parameters of NMF can be obtained using the following derivatives executed using multiple iterations,

$$\frac{\partial \log p(y^*_{m,n})}{\partial c_{m,k}} = \gamma \left(y^*_{m,n} - \sum_{k=1}^{K} c_{m,k} x^*_{k,n} \right) x^*_{k,n} + \frac{\xi}{c_{m,k}}$$

$$\frac{\partial \log p(y^*_{m,n})}{\partial x^*_{k,n}} = \gamma \left(y^*_{m,n} - \sum_{k=1}^{K} c_{m,k} x^*_{k,n} \right) c_{m,k} + \frac{\xi}{x^*_{k,n}}$$

$$new\ c_{m,k} = old\ c_{m,k} + \eta \cdot \frac{\partial \log p(y^*_{m,n})}{\partial c_{m,k}}$$

$$new\ x^*_{k,n} = old\ x^*_{k,n} + \eta \cdot \frac{\partial \log p(y^*_{m,n})}{\partial x^*_{k,n}}$$

where γ represents the precision of error, ξ represents the strictness of the log barrier constraint and η represents the rate of learning for SGD. In our experiments, we use the parameter settings $\gamma = 1$, $\xi = 0.01$, and $\eta = 0.0001$.

C. Dynamic Matrix Factorization

DMF can be seen as an extension of NMF by adding the time dimension based on *Linear Dynamical Systems* (LDS). LDS is originally designed to relate an output signal $y_t \in \mathbf{R}^M$ at time step t with some latent vector $x_t \in \mathbf{R}^K$ at time step t, and the latent vectors x at earlier time steps. Formally, we define LDS as follows,

$$\begin{aligned} y_t &= C \cdot x_t + v & x_t &= A_t \cdot x_{t-1} + w \\ v &\sim \mathcal{N}(0, R) & w &\sim \mathcal{N}(0, Q) \end{aligned}$$

where $C \in \mathbf{R}^{M \times K}$ is the item-factor matrix, and $A_t \in \mathbf{R}^{K \times K}$ is the factor to factor mapping between adjacent time steps. The covariance matrices $Q \in \mathbf{R}^{K \times K}$ and $R \in \mathbf{R}^{M \times M}$ are set to be $0.1 \cdot \mathbf{I}$ in our experiments.

The above LDS formulation models only a single user's data across time steps. It can be extended to model dynamic data of a set of users in a dynamic matrix factorization model. Sun et al. defined a version of DMF as follows [1]:

$$\begin{aligned} y_{n,t} &= C_n \cdot x_{n,t} + v & x_{n,t} &= A_{n,t} \cdot x_{n,t-1} + w \\ v &\sim \mathcal{N}(0, R) & w &\sim \mathcal{N}(0, Q) \end{aligned}$$

This version of DMF learns a fixed item-factor matrix C for all users. Instead of learning C, we propose to use an item-factor matrix derived from NMF. We also propose four other versions of DMF based on the options used for **Item Factor Matrix** and **Dynamics Matrix** as shown in Table I.

Table I
PROPOSED DMF MODELS

	Non-Scaled Item Factors	Scaled Item Factors
Variable Dynamics Matrix	DMF-B	DMF-I
Fixed Dynamics Matrix	DMF-A	DMF-IA

Basic DMF (DMF-B). In this Basic DMF model, we determine a static item-factor matrix C using NMF while allowing the factor-user matrix X_t to vary with time. That is, we define Basic DMF to be:

$$y_{n,t} = C \cdot x_{n,t} + v \quad Y^* = C \cdot X^* \quad \text{using NMF}$$

keeping the equations of $x_{n,t}$, w and v the same.

To use DMF for obtaining an estimate of $\hat{y}_{m,n,t}$, we calculate $y_{n,m,t|T}$, the frequency of adoptions by user n on item m at time t conditioned on all information up to the last time step T.

$$y_{n,t|T} = C \cdot x_{n,t|T}$$

DMFs with Scaled Item Factors (DMF-I and DMF-IA). The DMF-I and DMF-IA models consider that the item-factor matrix C learnt from NMF is determined for the observations for *all* time steps, i.e., Y^*. With large observed adoption counts in Y^*, we expect larger entries in C. The consequence of this is an over-estimation of item factors for each time step. Consider using the NMF model to recover adoptions, we have

$$y^*_{m,n} = \sum_{k=1}^{K} c_{m,k} \cdot x^*_{k,n}$$

However, in DMF, we have

$$y_{m,n,t} = \sum_{k=1}^{K} c_{m,k} \cdot x_{n,t,k}$$

In NMF, both C and X^* contribute to the observation of the magnitude in Y^* for the respective indices. However, in DMF-B, the adoption magnitude Y is spread out over multiple time periods. If C remains constant when inferring for the values of $x_{n,t}$, the value of $x_{n,t}$ will have to be adjusted downwards in order to compensate for the reduction of the observed value $y_{m,n,t}$. While it is convenient to allow $x_{n,t}$ to bear the burden of adjusting for $y_{m,n,t}$, we could also adjust C such that it is suitable for the number of observed time steps for each user n. For example, if a user is only active in one time step, then C_n should be no different with the C from NMF. However, if user is active in multiple time steps, then C_n for user n should be scaled such that $C_n < C$. In the DMF-I model, we therefore scale C by the number of time steps.

$$C_n = \frac{C}{\text{\# of observed time steps for user } n}$$

Alternatively, C can be estimated via a log likelihood maximization approach in the same way as how A is optimized. But the elegance of how LDS is being defined allows for the parallel estimation of the x_n's and A_n's parameters independently from each user n. Therefore if we learn the C that is coupled with all other users, it becomes computationally expensive with little room for parallelization and scalability.

DMFs with Fixed Dynamics (DMF-A and DMF-IA). In both DMF-B and DMF-I, the dynamics matrix A is different (or variable) for each user and each time step. In predicting missing total adoptions, we have user-item pairs that do not involve any adoption across all time steps. Using different dynamics matrices across time steps may cause over-fitting problem in DMF-B and DMF-I and prevent accurate prediction of missing total adoptions. We therefore propose to learn a fixed dynamics matrix A for each user across all time steps. DMF-A and DMF-IA thus have the following equation for $x_{n,t}$.

$$x_{n,t} = A \cdot x_{n,t-1} + w$$

Parameter Learning for DMF. The estimation of parameters in all the DMF models can be derived as laid out in Rauch, Tung and Striebel [16] and that of Ghahramani and Hinton [17]. In the following, we only show the learning of parameters for DMF-B and DMF-I.

Let $x_{n,t|T}$ be the *smoothed* latent state variable of user n at time t conditioned on T. Without showing the explicit derivations, we only state the equations here. Readers interested in the derivations can refer to Rauch, Tung and Striebel [16]. The steps listed here is known as *RTS smoothing*.

$$\begin{aligned} x_{n,t|T} &= x_{n,t|t} + J_{n,t}\left(x_{n,t+1|T} - x_{n,t+1|t}\right) \\ J_{n,t} &= P_{n,t|t} A'_{n,t} P^{-1}_{n,t+1|t} \\ P_{n,t|T} &= P_{n,t|t} + J_{n,t}\left(P_{n,t+1|T} - P_{n,t+1|t}\right) J'_{n,t} \end{aligned}$$

The smoothed latent states depends on the prior latent states $x_{n,t|t-1}$ and posterior latent states $x_{n,t|t}$. The posterior and prior latent states are obtained through a process known as *Kalman filtering* [18].

$$\begin{aligned} x_{n,t|t-1} &= A_{n,t} x_{n,t-1|t-1} \\ P_{n,t|t-1} &= A_{n,t} P_{n,t-1|t-1} A'_{n,t} + Q \\ K_{n,t} &= P_{n,t|t-1} C' \left(C P_{n,t|t-1} C' + R\right)^{-1} \\ x_{n,t|t} &= x_{n,t|t-1} + K_{n,t}\left(y_{n,t} - C x_{n,t|t-1}\right) \\ P_{n,t|t} &= \left(I - K_{n,t} C\right) P_{n,t|t-1} \end{aligned}$$

The dynamics matrix $A_{n,t}$ is given by

$$\left(x_{n,t|T} \cdot x'_{n,t-1|T} + P_{n,t,t-1|T}\right)\left(x_{t-1|T} \cdot x'_{t-1|T} + P_{n,t-1|T}\right)^{-1}$$

Although the matrix C remains the same as before, the latent space vectors $x_{n,t}$, now divided by different time steps no longer have their non-negativity constraints enforced by the Kalman filtering and smoothing steps. This is because when solving for the posterior and smooth distributions of x, it also involves a maximization step without additional constraints on the polarity of the vectors. Although Lagrange constraints can be added to enforce x to lie on the positive orthant, the algebraic manipulations becomes far too complex to solve analytically. Stochastic gradient descent can be used for solving the posterior and smoothed vectors numerically but given the multiple time steps involved for multiple users, the complexity of such an approach is not feasible for data on a larger scale.

IV. Experiments

We evaluate our models against the baseline NMF and TimeSVD++[1] on the three tasks: 1) DMF-B and DMF-I for *Prediction of missing temporal adoptions*, 2) DMF-B and DMF-I for *Prediction of all total adoptions given*

[1]The TimeSVD++ we used is the implementation from GraphLab

missing temporal adoptions and 3) DMF-A and DMF-IA for *Prediction of missing total adoptions*. Evaluations on tasks 1 and 2 use the same training and testing sets while task 3 uses a different training and testing sets. In this section, we will discuss how the training and testing sets are constructed before reporting the results for the three tasks.

A. Data Set

We use a subset of publications from DBLP and ACM Digital Library (ACMDL). Using papers published in the Journal of ACM (JACM) as a seed set, we grow this seed set by including their authors and their non-JACM publications. We also include the co-authors of JACM authors, and the publications of these co-authors. We collect the titles and abstracts (for ACMDL only) of all the above publications.

The statistics of our data sets are given in Table II. In this experiment, we use authors and title/abstract words as users and items respectively. Each year is considered a time step. DBLP has twice as many authors as ACMDL due to the longer history of publications maintained by DBLP. DBLP covers a larger scope than ACMDL as the latter focuses only on ACM-related publications. However, ACMDL has many more unique words than DBLP, because ACMDL has both titles and abstracts, whereas DBLP only has titles.

Table II
DATASET SIZES

Data set	# authors	# unique non-stop words	# non-zero entries in Y	time steps
DBLP	52,754	20,080	4,085,265	1936–2012
ACMDL	24,569	33,044	8,721,385	1952–2011

Training and Testing sets for Task 1. To evaluate Task 1 (Prediction of missing temporal adoptions), we divide the temporal adoption matrix Y into five (training set, testing set) pairs, ($Y(i)^{train}$, $Y(i)^{test}$) for i=1 to 5. The process for creating these data sets is outlined as follows,

```
1) $Y(0)^{train} = Y$, $Y(0)^{test} = \emptyset$
2) For $i$=1 to 5
   a) $Y(i)^{train} = Y(i-1)^{train}$
      $Y(i)^{test} = Y(i-1)^{test}$
   b) For each $y(i)^{train}_{m,n,t} > 0$, with probability 0.1, do
      i) $y(i)^{test}_{m,n,t} = y(i)^{train}_{m,n,t}$
      ii) $y(i)^{train}_{m,n,t} = 0$
```

We deliberately hide 10% of adopted items in each time step. We then iteratively grow the testing set by shifting 10% of the adoptions in the training set to the testing set. This way, we can ensure that subsequent testing set is always a superset of the previous set. That makes the difficulty of predicting for the missing adoptions in the test set consistently more difficult than the previous set. We obtain five sets of testing data { 10%, 19%, 27%, 34%, 41% } with their respective training data.

Training and Testing sets for Task 2. For Task 2 Prediction of all total adoptions given missing temporal adoptions, we simply collapse the above $Y(i)^{train}$ and $Y(i)^{test}$ across time steps. That is, for each i, the training and test sets are defined by,

$$y^*(i)^{train}_{m,n} = \sum_t y(i)^{train}_{m,n,t} \qquad y^*(i)^{test}_{m,n} = \sum_t y(i)^{test}_{m,n,t}$$

Training and Testing sets for Task 3. For task 3, we divide the temporal adoption matrix Y into training sets $Y(j)^{train}$ and testing sets $Y(j)^{test}$, for j=1 to 5. The process for creating these data sets is listed as follows,

```
1) $Y(0)^{train} = Y$, $Y(0)^{test} = \emptyset$
2) $Y^*(0)^{train} = Y^*$, $Y^*(0)^{test} = \emptyset$
3) For $j$=1 to 5
   a) $Y(j)^{train} = Y(j-1)^{train}$
      $Y(j)^{test} = Y(j-1)^{test}$
   b) $Y^*(j)^{train} = Y^*(j-1)^{train}$
      $Y^*(j)^{test} = Y^*(j-1)^{test}$
   c) For each $y^*(j)^{train}_{m,n} > 0$, with probability 0.1, do
      i) $y^*(j)^{test}_{m,n} = y^*(j)^{train}_{m,n}$
         $y^*(j)^{train}_{m,n} = 0$
      ii) For $t$=1 to $T$
         A) $y(j)^{test}_{m,n,t} = y(j)^{train}_{m,n,t}$
         B) $y(j)^{train}_{m,n,t} = 0$
```

We create the testing set by randomly including 10% of the item m-user n pairs with non-zero $y^*_{m,n}$ from Y^*. The selected pairs are also excluded from the training set by setting $y_{m,n,t} = 0$ for all t. The size of the training and testing sets is then varied by randomly selecting another 10% from the training set and shifting it to the testing set.

B. Results for Prediction of Missing Temporal Adoptions

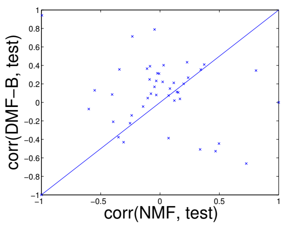
We used $Y(i)^{train}$ for training DMF-B/DMF-I and per time step data from $Y(i)^{train}$ for training NMF. The models then predict the missing adoptions for each time step $y(i)^{test}_{m,n,t}$ for all $y(i)^{test}_{m,n,t} > 0$. The purpose of this experiment is to show that even when NMF is applied independently to each time step, DMF-B and DMF-I are still able to outperform NMF. This indicates that the relationship between the user latent factors of adjacent time steps $x_{n,t}$ and $x_{n,t-1}$ captured by the dynamics matrix is necessary to more accurately predict missing temporal adoptions.

The predicted values given by NMF, DMF-B and DMF-I are denoted by $\hat{y}(i)^{nmf}_{m,n,t}, \hat{y}(i)^{dmf-b}_{m,n,t}$ and $\hat{y}(i)^{dmf-i}_{m,n,t}$ respectively. We compare the *Pearson Correlation Coefficient* (PCC) of the predicted values for each time step against $y(i)^{test}_{m,n,t}$ of each time step. PCC is preferred over *Root Sum Squared Error* (RSSE)[2] because the total adoptions when divided into multiple time steps have many small count values dominating RSSE over the large count values that are deemed more important.

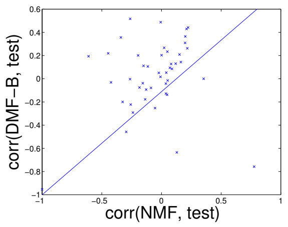
Figure 2 shows the result of DMF-B against the baseline NMF using ACMDL dataset for different proportions of test data. In the plot, The x-axis represents the PCC of NMF predicted values against the test (or ground truth) values

[2]Root Sum Squared Error is defined by the root of squared errors, i.e., $\sqrt{\sum_k error_k^2}$.

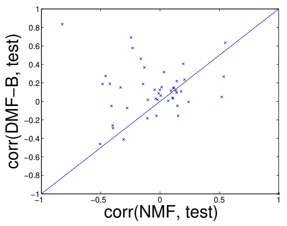
while y-axis represents the PCC of DMF- predicted values against the test values. Each dot represents the respective results of a year. If the dot lies on the upper-left side of graph, it indicates that for that year DMF-B performs better than NMF. Figure 2 indeed shows that for the four plots, most of the dots lie on the upper left side of the figure.



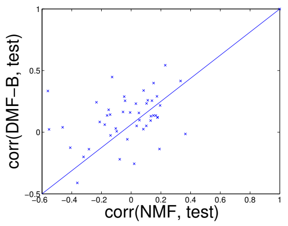
(a) For i=2, test size=19%



(b) For i=3, test size=27%



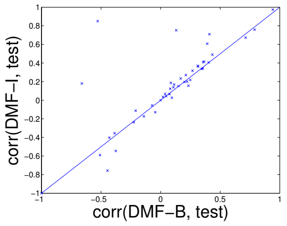
(c) For i=4, test size=34%



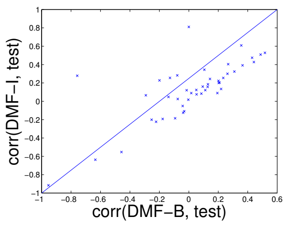
(d) For i=5, test size=41%

Figure 2. PCC of DMF-B against NMF for Task 1 (ACMDL)

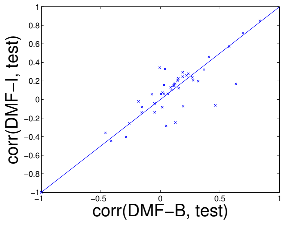
Figure 3 shows the results of DMF-I against DMF-B. The results show that most of the dots lie on the upper left side of the figures. This indicates that using a scaled item-factor matrix C achieve a better estimation of the latent factors. The two figures show that for most years, DMF-I outperforms DMF-B while DMF-B outperforms NMF. Due to space constraints and the small adoption values in each time step, we do not include DBLP for this task.



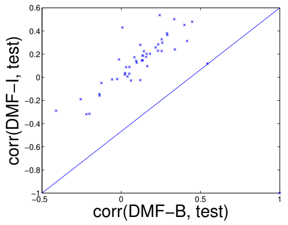
(a) For i=2, test size=19%



(b) For i=3, test size=27%



(c) For i=4, test size=34%



(d) For i=5, test size=41%

Figure 3. PCC of DMF-I against DMF-B for Task 1 (ACMDL)

C. Results for Prediction of Total Adoptions with Missing Temporal Adoptions

In this evaluation task, the training of DMF-B and DMF-I uses temporal adoptions in $Y(i)^{train}$ while training of NMF uses total adoptions in $Y^*(i)^{train}$. We evaluate how accurate these models predict the total adoptions in $y^*_{m,n}$ for all $y^*(i)^{test}_{m,n} > 0$ where

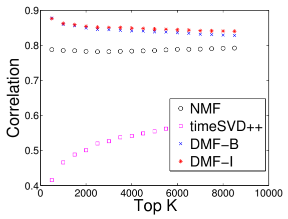
$$y^*_{m,n} = \sum_{t=1}^{T} y_{m,n,t}$$
$$y_{m,n,t} = y(i)^{train}_{m,n,t} + y(i)^{test}_{m,n,t}, \text{ for all } i = 1 \text{ to } 5$$

Using DMF-B or DMF-I, we can compute the value of $\hat{y}_{m,n,t}$ for each different time step t. Then an estimate of $\hat{y}^*_{m,n}$ is obtained by summing the predicted value across all time steps.

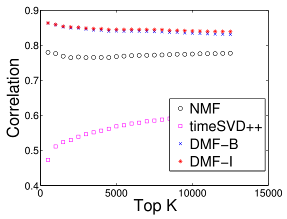
$$\hat{y}^*_{m,n} = max(\sum_{t=1}^{T} \hat{y}_{m,n,t}, 0)$$

If $\hat{y}^*_{m,n}$ is negative, it is unlikely user m adopts item n and we set the predicted adoption value to zero.

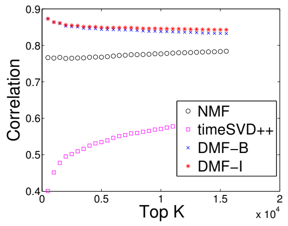
We evaluate the predicted adoption values against the test (ground truth) values using Pearson correlation coefficient (PCC) and Root Sum Squared Error (RSSE) for k largest test adoption values where k is varied from 1 to the number of test cases with adoption values not smaller than 20, ignoring the less important small adoption values.



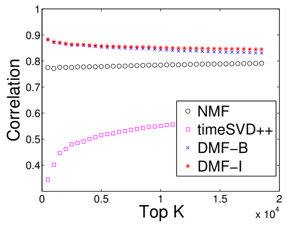
(a) Results for i=2, 19%



(b) Results for i=3, 27%



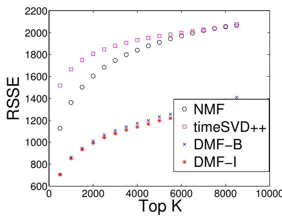
(c) Results for i=4, 34%



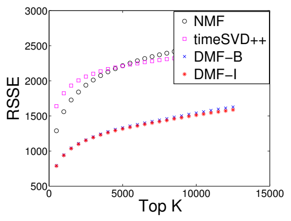
(d) Results for i=5, 41%

Figure 4. PCC of Task 2 (ACMDL)

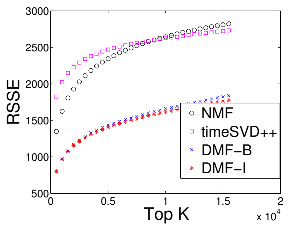
Figures 4 and 5 show the correlation and RSSE results for Task 2. The results show that DMF-I outperforms DMF-B by a very small margin and the DMF-B and DMF-I outperform NMF and TimeSVD++ by a large margin. This indicates that the two DMF models can recover the total adoptions more accurately when some temporal adoptions are missing. The PCC and RSSE performance reduces as we increase k



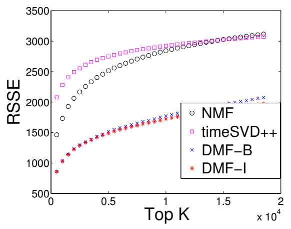
(a) Results for i=2, 19%



(b) Results for i=3, 27%

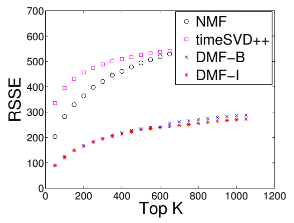


(c) Results for i=4, 34%

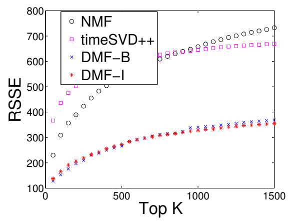


(d) Results for i=5, 41%

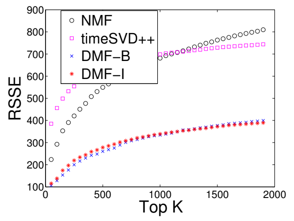
Figure 5. RSSE of Task 2 (ACMDL)



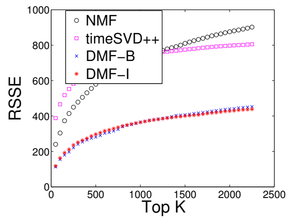
(a) For i=2, Test size=19%



(b) For i=3, Test size=27%



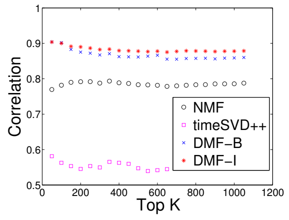
(c) For i=4, Test size=34%



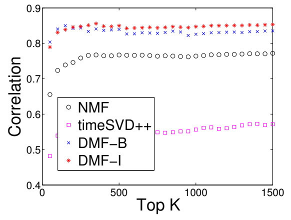
(d) For i=5, Test size=41%

Figure 7. RSSE of Task 2 (DBLP)

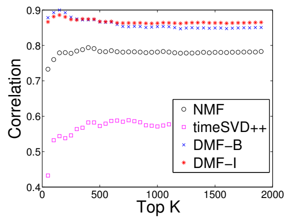
adding more errors to the measures. We also perform similar experiments on the DBLP data set. As shown in Figures 6 and 7, we also observe that DMF-B and DMF-I outperforms NMF and TimeSVD++ significantly by PCC and RSSE.



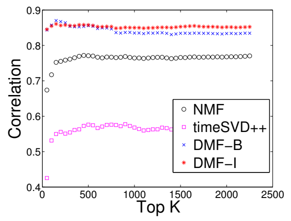
(a) For i=2, Test size=19%



(b) For i=3, Test size=27%



(c) For i=4, Test size=34%



(d) For i=5, Test size=41%

Figure 6. PCC of Task 2 (DBLP)

While it is expected that NMF will perform poorly on task 2 due to the lack of temporal considerations, we are surprised that TimeSVD++ also performs as poor as NMF on task 2. Manual inspection of the predicted values given by TimeSVD++ shows that TimeSVD++ predicts almost the same adoption values $\hat{y}^*_{m,n,t}$ for all time steps t where user n is active in. Given that for task 2, user adoption values for an item m is missing in some but not all of the time steps, an adoption model should cope with such variations in item user adoption values throughout the entire temporal duration. Since TimeSVD++ was originally developed for user-item rating prediction, it assumes that once item has been rated by user, the rating remains the same throughout the entire temporal duration. Such an assumption violates the conditions necessary for good prediction in task 2.

D. *Results for Missing Total Adoptions*

For this task, we train DMF-A and DMF-IA using $Y(j)^{train}$ and train NMF using $Y^*(j)^{train}$. We want to investigate if the temporal adoption data of known user-item pairs can help to predict the missing total adoption for a given user-item pair. The test total adoptions to be predicted are $y^{*,test}_{m,n}$ for all $Y^*(j)^{test} > 0$ where

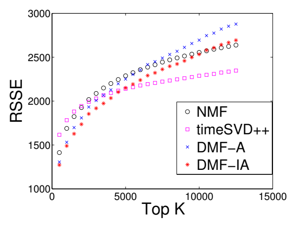
$$y^{*,test}_{m,n} = \sum_{t=1}^{T} y^{test}_{m,n,t}$$

DMF-A and DMF-IA are required to predict the values of $\hat{y}_{m,n,t}$ for each different time steps t. They then give an estimate of $\hat{y}^*_{m,n}$ by summing the predicted value across all time steps.

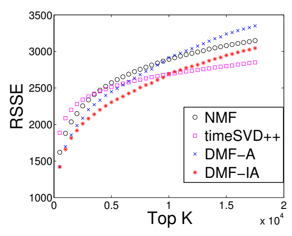
$$\hat{y}^*_{m,n} = max(\sum_{t=1}^{T} \hat{y}_{m,n,t}, 0)$$

If $\hat{y}^*_{m,n}$ is negative, we set it to zero. The predicted total adoptions are then compared with test (ground truth) adoptions $y^*_{m,n}$ by Root Sum Squared Error (RSSE).

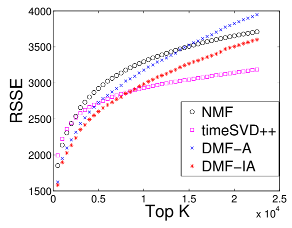
We again evaluate the predicted adoption values against the test (ground truth) values using PCC and Root Sum Squared Error RSSE for k largest test adoption values where k is varied from 1 to the number of test cases with adoption values not smaller than 20, ignoring the less important small adoption values. Figures 8 and 9 shows the RSSE results for the ACMDL and DBLP data set. DMF-IA is observed to have smaller RSSE than DMF-I showing that fixed dynamics matrix and scaled item factors are required to yield more accurate predictions than DMF-I and NMF for this task. DMF-I again outperforms NMF for PCC and RSSE predictions. However, TimeSVD++ have better performance for task 3 in the comparison of RSSE values. Since for task 3, the adoption values of item m and user n are consistently missing for all time steps, the adoption model does not have to make different prediction values for different time steps. When comparing against the aggregated item adoption values $Y^*(i)$, this hides the weakness of rating prediction models such as TimeSVD++.



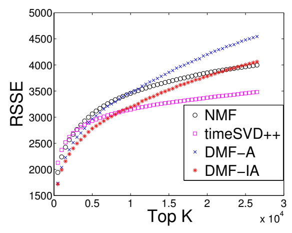
(a) For j=2, Test size=19%



(b) For j=3, Test size=27%

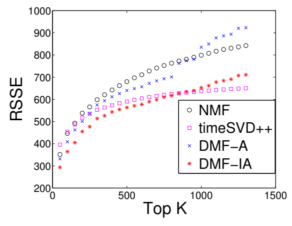


(c) For j=4, Test size=34%

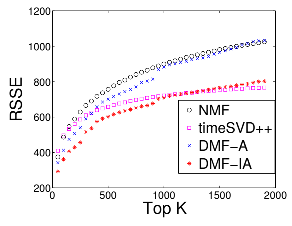


(d) For j=5, Test size=41%

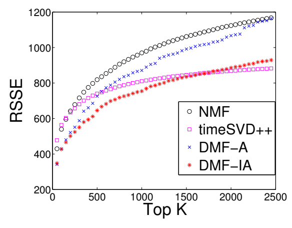
Figure 8. RSSE of Task 3 (ACMDL)



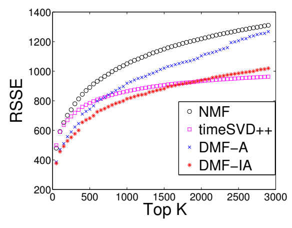
(a) For j=2, Test size=19%



(b) For j=3, Test size=27%



(c) For j=4, Test size=34%



(d) For j=5, Test size=41%

Figure 9. RSSE of Task 3 (DBLP)

E. *Case Study*

A main feature of DMF formulation is the use of dynamics matrix $A_{n,t}$ to capture the evolution of user n's latent factors $x_{n,t}$ from one time step to the next time step. The latent state at t is given by

$$x_{n,t} = A_{n,t} \cdot x_{n,t-1}$$

The k^{th} factor in $x_{n,t}$ is derive by the dot product of the k^{th} row of $A_{n,t}$ and $x_{n,t-1}$. The largest value in the k^{th} row of $A_{n,t}$, say the (k,l) value, tells us that the j^{th} latent factor in $x_{n,t-1}$ plays a significant role in explaining for the value of the k^{th} latent factor in $x_{n,t}$.

We explain the evolution of *Duminda Wijesekera*'s latent factors for the years (2000 to 2001) and (2001 to 2002). From the item factor matrix C, we can derive the underlying topics of some latent factors as shown in Table III. *Duminda Wijesekera* has research interests in security, multimedia, networks, etc.. His 6^{th} latent factor, corresponding to security topic, evolves from 2.25 in 2000 to 3.23 in 2001, and later to 9.10 in 2002. We also notice that the $(6,20)^{th}$ entry in the 6^{th} row of $A_{Duminda,2001}$ has the highest value of 0.347 while the other entries in the same row have a mean value of 0.0387. In addition, in the 6^{th} row of $A_{n,2002}$, the $(6,6)^{th}$ entry has the highest value of 0.3625 while the other values have mean value of 0.1418. This suggests that *Duminda Wijesekera* shifted his research from databases to security from 2000 to 2001. Then from 2001 onwards, the security topic continues to be his main research topic.

Consider another well known author *Christos Faloutsos* who has published widely in databases, data mining and graph mining. The 23^{th} factor of *Christos Faloutsos*, corresponding to graph mining, increased from 2.90 in year 2006 to 14.83 in year 2007. By inspecting his dynamics matrix $A_{Christos,2007}$, we noticed that the $(23,20)^{th}$ entry of the 23^{th} row has the highest value of 0.5055 while the mean value of other entries in the same row is 0.1056. This indicates that *Christos Faloutsos*'s increased research in the graph mining comes from his previous research interest in databases.

Table III
LATENT FACTORS

Factor 6	Factor 20	Factor 23	Factor 18
access	data	mining	network
control	large	graph	networks
paper	database	cache	wireless
systems	approach	graphs	nodes
based	techniques	frequent	sensor
model	algorithms	patterns	traffic
information	efficient	memory	infiniband
security	stream	vertices	routing
system	query	pattern	mobile
policies	problem	vertex	node

Finally, we observe that another database researcher *Beng Chin Ooi* has shifted his research interests from database to mobile systems between the years 2003 to 2004. The

18^{th} factor (corresponding to mobile systems) of his latent state increased from 1.6907 to 9.1483 between 2003 and 2004. In the 18^{th} row of *Beng Chin Ooi*'s dynamics matrix $A_{Beng\ Chin,2004}$, the $(18, 20)^{th}$ entry shows a large magnitude of 0.2441 while the rest of the other factors give a mean value of 0.0933. This indicates that the increase in mobile systems came from previous involvement with databases.

We stress again that without the use of NMF for DMF, we will not be able to observe such case studies for individual authors.

V. Conclusion

We have highlighted the differences between rating prediction and adoption prediction. When the data given contains temporal information, we proposed the use of Dynamic Matrix Factorization (DMF) for modeling the dynamics of latent states for every user. The empirical results show that using DMF gives overall better performance over NMF and state of the art method such as TimeSVD++. Our case study shows three examples of well-known researchers who changed the focus of their research career from a particular field to other fields in Computer Science. By analyzing the different latent states at different time steps, we can notice the years which indicate a tipping point in their focus. Then by further analyzing the dynamics matrix for the tipping point years, we can observe which fields they contributed to the interest in their respective new fields. Without the non-negative constraints in the item factor matrix for DMF, we will not be able to obtain latent factors that can be interpreted as topics of interests for the users. Therefore, the models proposed here can be used as a form of dynamic topic models for tracking the evolution of users' behavior over time. Temporal data sets have also been gaining attention [19] and the models we highlighted here could be applied to other social media data sets as well.

Acknowledgment

We will like to thank Sun et al. [1] for sharing their code. This research is supported by the Singapore National Research Foundation under its International Research Centre @ Singapore Funding Initiative and administered by the IDM Programme Office, Media Development Authority (MDA).

References

[1] J. Z. Sun, K. R. Varschney, and K. Subbian, "Dynamic matrix factorization: A state space approach," in *IEEE International Conference on Speech and Signal Processing*, 2012, pp. 1897–1900.

[2] H. Ma, H. Yang, M. R. Lyu, and I. King, "Sorec: social recommendation using probabilistic matrix factorization," in *CIKM*. New York, NY, USA: ACM, 2008, pp. 931–940.

[3] Y. Koren, "Factorization meets the neighborhood: a multifaceted collaborative filtering model," in *SIGKDD*, 2008, pp. 426–434.

[4] ——, "Collaborative filtering with temporal dynamics," in *SIGKDD*, New York, NY, 2009, pp. 447–456.

[5] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *IEEE Computer*, vol. 42, no. 8, pp. 30–37, 2009.

[6] L. Xiong, X. Chen, T.-K. Huang, J. Schneider, and J. G. Carbonell, "Temporal collaborative filtering with Bayesian probabilistic tensor factorization," in *SDM*, Columbus, OH, 2010, pp. 211–222.

[7] R. Salakhutdinov and A. Mnih, "Probabilistic matrix factorization," in *NIPS*. Cambridge, MA: MIT Press, 2008, pp. 1257–1264.

[8] ——, "Bayesian probabilistic matrix factorization using Markov chain Monte Carlo," in *ICML*, Helsinki, Finland, 2008, pp. 880–887.

[9] W. Xu, X. Liu, and Y. Gong, "Document clustering based on non-negative matrix factorization," in *SIGIR*, ser. SIGIR '03. New York, NY, USA: ACM, 2003, pp. 267–273.

[10] C. Liu, H.-c. Yang, J. Fan, L.-W. He, and Y.-M. Wang, "Distributed nonnegative matrix factorization for web-scale dyadic data analysis on mapreduce," in *WWW*. New York, NY, USA: ACM, 2010, pp. 681–690.

[11] D. D. Lee and H. S. Seung, "Learning the parts of objects by non-negative matrix factorization," *Nature*, vol. 401, no. 6755, pp. 788–791, 1999.

[12] ——, "Algorithms for nonnegative matrix factorization," in *Proceedings of the Advances in Neural Information Processing Systems*, vol. 13. MIT Press, 2001, pp. 556–562.

[13] B. Cao, D. Shen, J.-T. Sun, X.-H. Wang, Q. Yang, and Z. Chen, "Detect and track latent factors with online nonnegative matrix factorization," in *IJCAI*, 2007, pp. 2689–2694.

[14] F. Wang, H.-H. Tong, and C.-Y. Lin, "Towards evolutionary nonnegative matrix factorization," in *AAAI*, 2011, pp. 501–506.

[15] Z. Lu, D. Agarwal, and I. S. Dhillon, "A spatio-temporal approach to collaborative filtering," in *Proceedings of the ACM Conference on Recommender Systems*, New York, NY, 2009, pp. 13–20.

[16] H. E. Rauch, C. T. Striebel, and F. Tung, "Maximum likelihood estimates of linear dynamic systems," *Journal of the American Institute of Aeronautics and Astronautics*, vol. 3, no. 8, pp. 1445–1450, Aug. 1965.

[17] Z. Ghahramani and G. E. Hinton, "Parameter estimation for linear dynamical systems," *University of Toronto technical report CRGTR962*, vol. 6, no. CRG-TR-96-2, pp. 1–6, 1996.

[18] R. E. Kalman, "A new approach to linear fitering and prediction problems," *Transactions of the ASME–Journal of Basic Engineering*, vol. 82 (Series D), no. 1, pp. 35–45, 1960.

[19] H. Gao, J. Tang, X. Hu, and H. Liu, "Exploring temporal effects for location recommendation on location-based social networks," in *RecSys*, 2013.