RESEARCH ARTICLE

# Using Dynamic Multi-Task Non-Negative Matrix Factorization to Detect the Evolution of User Preferences in Collaborative Filtering

**Bin Ju**[1,2], **Yuntao Qian**[2]*, **Minchao Ye**[1], **Rong Ni**[2], **Chenxi Zhu**[2]

**1** Institute of Artificial Intelligence, College of Computer Science, Zhejiang University, Hangzhou, Zhejiang, P.R. China, **2** Health Information Center of Zhejiang Province, Hangzhou, Zhejiang, P.R. China

* ytqian@zju.edu.cn

**Citation:** Ju B, Qian Y, Ye M, Ni R, Zhu C (2015) Using Dynamic Multi-Task Non-Negative Matrix Factorization to Detect the Evolution of User Preferences in Collaborative Filtering. PLoS ONE 10 (8): e0135090. doi:10.1371/journal.pone.0135090

**Editor:** Zhaohong Deng, Jiangnan University, CHINA

**Received:** March 9, 2015

**Accepted:** July 17, 2015

**Published:** August 13, 2015

**Data Availability Statement:** Netflix data are available at http://figshare.com/articles/netflix_part01_rar/1451427 and http://figshare.com/articles/netflix_part02_rar/1451426. The Last.fm dataset is available at http://figshare.com/articles/lastfm_part01_rar/1451409 and http://figshare.com/articles/Last_fm_dataset/1451408.

**Funding:** The work described in this paper was supported by the following funding: 1) the Natural Science Foundation of Zhejiang Province of China (no. Y1101359); 2) the National Basic Research Program of China (no. 2012CB316400); and 3) the Major Projects of Zhejiang Provincial Science and

## Abstract

Predicting what items will be selected by a target user in the future is an important function for recommendation systems. Matrix factorization techniques have been shown to achieve good performance on temporal rating-type data, but little is known about temporal item selection data. In this paper, we developed a unified model that combines Multi-task Non-negative Matrix Factorization and Linear Dynamical Systems to capture the evolution of user preferences. Specifically, user and item features are projected into latent factor space by factoring co-occurrence matrices into a common basis item-factor matrix and multiple factor-user matrices. Moreover, we represented both within and between relationships of multiple factor-user matrices using a state transition matrix to capture the changes in user preferences over time. The experiments show that our proposed algorithm outperforms the other algorithms on two real datasets, which were extracted from Netflix movies and Last.fm music. Furthermore, our model provides a novel dynamic topic model for tracking the evolution of the behavior of a user over time.

## Introduction

Personalized recommender systems are widely used in e-commerce, such as by Amazon and Netflix, to identify interesting products for their customers. Recommendation systems are often based on *Collaborative Filtering* (CF) [1–3], which relies only on past user feedback data, e.g., users' previous transactions or item ratings. One type of user feedback data used in the CF literature is the explicit form of user-ratings, e.g., a 1–5 score. In this scenario, one of the challenges of CF of rating-type data is how to predict the missing values of the user-item score matrix effectively and efficiently, which has been explored widely [4–7]. However, in real-world scenarios most feedback is not explicit but implicit. Implicit feedback is tracked automatically, such as by monitoring clicks, view times, purchases, and other user activity. A common type of behavior in implicit feedback is user and item co-occurrence data reflecting user preferences, which we denote as selection data. There are many application scenarios in which a user can select the same item more than once, and we denote the behaviors in these scenarios as dynamic selection

Technology Program for Traditional Chinese Medicine (no. 2015ZZ005).

**Competing Interests:** The authors have declared that no competing interests exist.

data. In the latter scenario, the data are usually divided into time steps, and the data in different time steps reflect different user preferences. Considering the dynamic selection data, an urgent and interesting question emerges: given the past behavior of the users when selecting items, how can the items that will be selected by the target user in the next time period be predicted?

Tracking user preferences is a key issue in the task of prediction. A main approach to model user preference is to use latent factor models, e.g., latent semantic models [8–10] and matrix factorization models [4, 6], which learn a latent feature/factor vector for each user and each item in the dataset such that the inner product of these features minimizes an explicit or implicit cost function. This approach can also be considered as an example of co-clustering, where one cluster represents the item's latent factor and the other represents the user's latent factor. Fig 1(a) shows the result of performing topic modeling on user preferences without temporal considerations. The items selected by users $u_1$, $u_2$ and $u_3$ are clustered according to topics 1 and 2 based on the weights (i.e., instances) of edges between users and items. The cluster of items is similar to the cluster of topic concepts. In fact, a user selects an item according to his preference, which is similar to words belonging to a document according to its topic. The topic concept clusters were derived using Probabilistic Latent Semantic Analysis (PLSA) [11] and Latent Dirichlet Allocation (LDA) [12]. These models exploit co-occurrence patterns of words in documents to unearth semantically meaningful probabilistic clusters of words. However, in practice, user preferences can change over time; therefore, this problem requires a method that is capable of modeling the temporal dependence of selections, i.e., a model that will allow us to provide the edge weights across time steps and preserve the topic distributions at different time steps. Such model could combine the temporal selections and construct dependencies between different time steps. See the scenario shown in Fig 1(b) as an example. We commonly refer to this type of models as temporal latent factor models.

In general, there are two types of temporal latent factor models for CF: temporal probabilistic topic models, such as the Dynamic Topic Model (DTM) [13], and dynamic latent factor-based matrix factorization approaches [5, 7, 14, 15]. However, both of these models have shortcomings. For example, DTM does not consider the evolution of user behavior, which is the main assumption in our work. Specifically, we assume that the topic-item distributions remain static over time but the preferences of the users change over time. Considering the dynamic changes in the preferences of users, Sahooet et al. proposed a temporal topic-user model [16], that extended from static latent factors representing users' preferences in an Aspect Model [9] to dynamic latent factors with a Hidden Markov Model (HMM). Although this model considers dynamic changes in users' preferences, it cannot predict the state of the user every time because it usually places most of the probability mass over a few states [16]. In contrast, the matrix factorization approaches have always been focused on predicting users' ratings for items rather than users' selection data. As noted above, in the case of temporal rating, the user can only rate an item once, but in reality, a user can select the same item many times as the user's preference for the item changes over time. Therefore, these two types of models cannot be applied for modeling user item selection.

Fortunately, in recent years, Non-negative Matrix Factorization (NMF) has been extensively researched as a parts-based non-negative dictionary learning method [17]. Specifically, the non-negative constrain of NMF in latent factors space can be used to model a user's preferences. In fact, many researchers have adopted NMF and its variants to solve CF [18–20]. Historically, PLSA and NMF were developed independently, but researchers later proved that PLSA solves the problem of NMF with KL I-divergence [21, 22]. To extend NMF to a dynamic latent factor model for selection data, Chua et al. [23] presented an algorithm called Dynamic Matrix Factorization (DMF), which combines NMF and Linear Dynamical Systems (LDS). However, DMF is not a unified model: it first uses NMF to generate an item-factor matrix as a

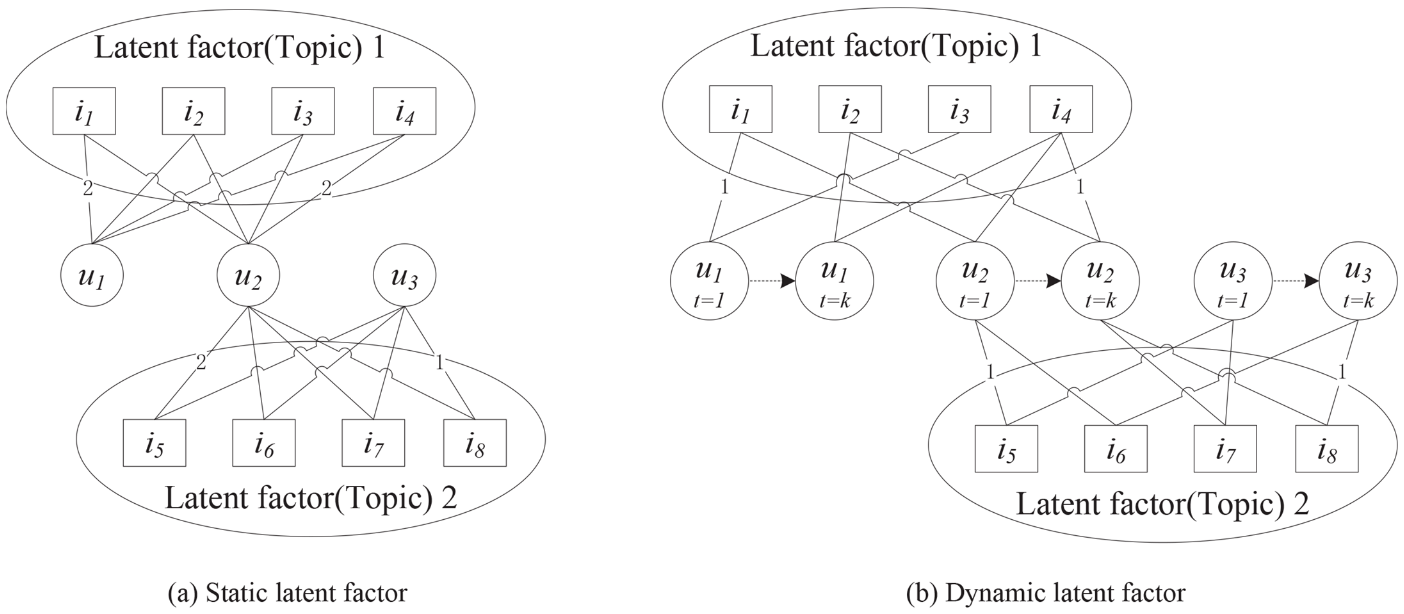


**Fig 1. Topic Modeling in User Item Selection.**

doi:10.1371/journal.pone.0135090.g001

global basis matrix for the observed data $\boldsymbol{X}^*$ and then uses LDS to generate a state-transition matrix that governs the evolution of the factor-user matrices.

Motivated by DMF, we model dynamic selection data using a unified model that combines multi-task non-negative matrix factorization and a state transition matrix derived from LDS. Multi-task learning has become more popular because this method encourages learning tasks in parallel using a shared representation, which helps for each task learn better by using the other tasks' information [24–26]. The multi-task NMF is a special case of non-negative tensor factorization (NTF) [27, 28], which is more flexible and useful in practice because it attempts to estimate only one common factor in the form of a basis matrix and some coefficient matrices over time [29]. In other words, the accumulative selection matrix $\boldsymbol{X}^*$ is factorized into a common basis matrix for discovering item-factor and multiple factor-user matrices whose temporal relationships are governed by a transition matrix. The main contributions of this paper can be summarized as follows:

1. Our model provides a new method to embed common item factors and temporal user factors information into a unified model, in which user preference is tracked by a state transition matrix and every user has his/her own preference's evolution profile.
2. Our work, which recovers the evolution of latent users' factors can be interpreted as a special case of the dynamic topic model.
3. Compared with state-of-the-art methods, the proposed approach demonstrates superior performance in the prediction of future selection behavior.

The remainder of this paper is organized as follows: The 'Analysis' section introduces our model as well as the advantages and physical interpretations of the model. The multiplicative update algorithms for our model are also derived in this section. Experimental results from real world data are demonstrated and analyzed in the 'Results' section. Finally, conclusions are drawn in the 'Discussion' section.

## Analysis

### Dynamic Multi-task NMF for CF

Here, we formulate the problem by focusing on the dynamic selection of data. Given $K$ time periods $\{t_1, t_2, \ldots, t_K\}$, in every time period, the users provide selection information about their preferences for known items, i.e., there are matrices $\boldsymbol{X}_k = \{\boldsymbol{X}_1, \boldsymbol{X}_2, \ldots, \boldsymbol{X}_K\}$ for $K$ time periods. In each $\boldsymbol{X}_k$, $\boldsymbol{X}_{k,\,ij} = x$ represents a user $i$ selecting $x$ instances of an item $j$ at time $k$. The accumulative selection matrix $\boldsymbol{X}^*$ is obtained by $\boldsymbol{X}^* = \sum_{1:K} \boldsymbol{X}_k$. The task of the time-sensitive recommender systems is to predict the items that a user will select in a future time period $K + 1$ given all of the selection matrices from the past $K$ time periods.

In NMF, the goal is to find entrywise non-negative matrices $\boldsymbol{W}$ and $\boldsymbol{H}$ such that

$$(\boldsymbol{W}, \boldsymbol{H}) = \underset{\boldsymbol{W},\boldsymbol{H}}{\arg\min}\, \mathcal{L}(\boldsymbol{X} \parallel \boldsymbol{W}\boldsymbol{H}) \tag{1}$$

where the function $\mathcal{L}$ is a suitable lost function and the $\boldsymbol{X}$ is the observation data matrix. The function $\mathcal{L}$ can be Euclidian distance or divergence [17, 30]. In our model, the selection matrix $\boldsymbol{X} \in R_+^{M\times N}$ is factorized into the item-factor matrix (or basis matrix) $\boldsymbol{W} \in R_+^{M\times D}$ and the factor-user matrix (or coefficient matrix) $\boldsymbol{H} \in R_+^{D\times N}$, where $M$ is the number of items, $N$ is the number of users, and $D$ represents the size of dimension of the latent factors space. The item-factor matrix $\boldsymbol{W}$ represents the projection from items space to latent factors space, whereas the factor-user matrix $\boldsymbol{H}$ represents the coefficients for a user's preferences for corresponding items [31]. We follow the classical assumption that the rows of the item-factor matrix $\boldsymbol{W}_i^T$ and the columns of the factor-user matrix $\boldsymbol{H}_j$ follow Gaussian prior distributions [4–6], as defined below:

$$p(\boldsymbol{W}_i^T) = \mathcal{N}(\boldsymbol{W}_i^T \mid \mu_w, \sigma_w^2 \mathbf{1}), \qquad p(\boldsymbol{H}_j) = \mathcal{N}(\boldsymbol{H}_j \mid \mu_h, \sigma_h^2 \mathbf{1}). \tag{2}$$

Although the observation data $\boldsymbol{X}_{ij}$ are integers, the residuals $\left(\boldsymbol{X}_{ij} - \boldsymbol{W}_i^T \boldsymbol{H}_j\right)$ should follow Gaussian distributions. Therefore, we define the conditional distribution over the observed data as

$$p(\boldsymbol{X} \mid \boldsymbol{W}, \boldsymbol{H}, \sigma^2) = \prod_{i=1}^{N} \prod_{j=1}^{M} [\mathcal{N}(\boldsymbol{X}_{ij} \mid \boldsymbol{W}_i^T \boldsymbol{H}_j, \sigma^2)]. \tag{3}$$

where $\sigma$ is the prior standard variance of the selection data.

A direct and simple way to solve dynamic selection prediction is to perform NMF independently for each time step. However, this process reduces the predictive power because the lower ranking factors are not related across different time steps. In fact, we assume that item factors evolve very slowly and can be considered constant over time, whereas each user factors changes over time [32, 33], i.e., given $K$ related input data matrices $\boldsymbol{X}_1, \boldsymbol{X}_2, \ldots, \boldsymbol{X}_K$, $K$ coefficient matrices $\boldsymbol{H}_1, \boldsymbol{H}_2, \ldots, \boldsymbol{H}_K$ and a common basis matrix $\boldsymbol{W}$ can be factorized. According to the structure illustrated in Fig 2, we have a unified probabilistic graphical model, as shown in Fig 3.

As shown in Fig 3, a classical multi-task NMF model can be used. Multi-task NMF was first proposed in [34] for extracting common gene profiles. In our model, multi-task NMF can be seen as a combination of several strongly related NMF tasks which share the common item-

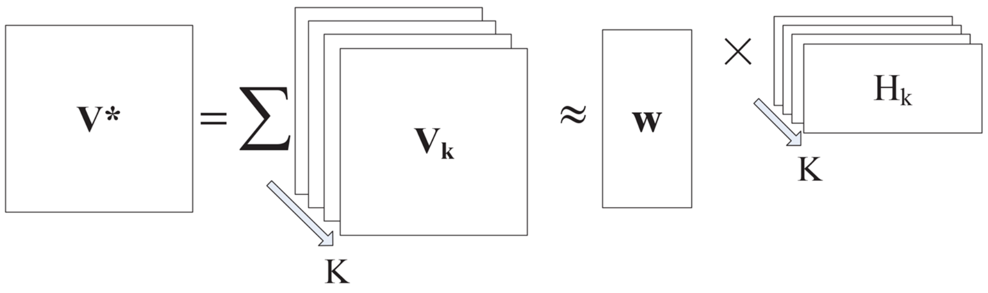


**Fig 2. The framework of DMNMF.**

doi:10.1371/journal.pone.0135090.g002

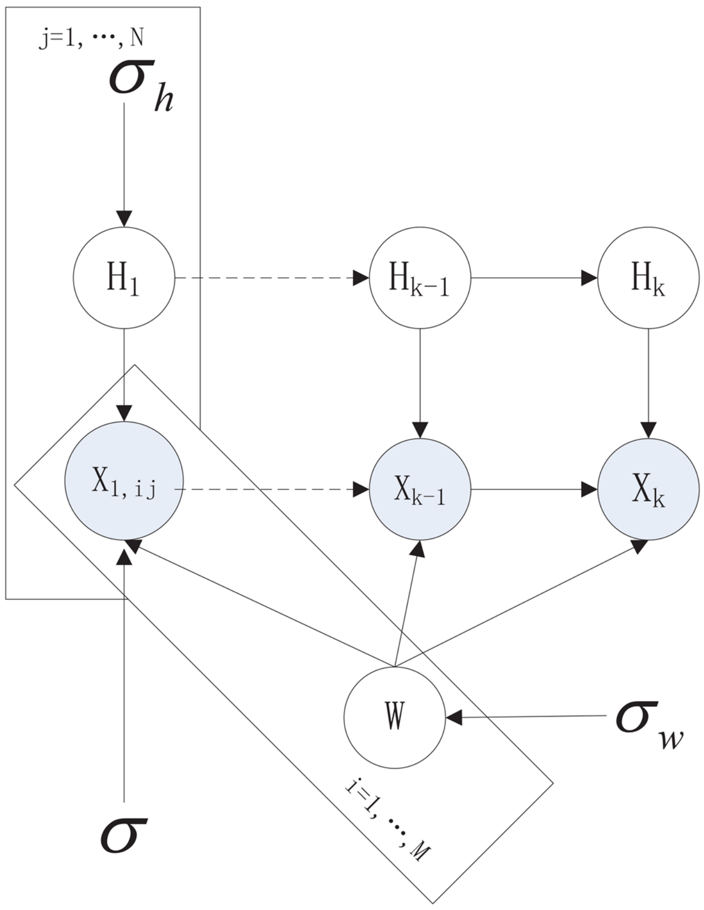


**Fig 3. Probabilistic graphical model of DMNMF.**

doi:10.1371/journal.pone.0135090.g003

factors matrix defined below:

$$p(\boldsymbol{X}_k \mid \boldsymbol{W}, \boldsymbol{H}_k, \sigma^2) = \prod_{k=1}^{K}\prod_{i=1}^{N}\prod_{j=1}^{M}[\mathcal{N}(\boldsymbol{X}_{k,ij} \mid \boldsymbol{W}_i^T \boldsymbol{H}_{k,j}, \sigma^2)] \tag{4}$$

where $\boldsymbol{H}_{k,\,j}$ is the preference of user $j$ during the $k^{th}$ time period.

Let us consider the relationship between $\boldsymbol{H}_{k-1,j}$ and $\boldsymbol{H}_{k,j}$, which represent the changes in a user's preferences from time period $k-1$ to $k$. We adopt the idea of LDS [14, 23], which represents the mapping of latent factors from time step $k-1$ to $k$ using a state transition matrix $\boldsymbol{A}_k$ with added Gaussian noise defined as follows:

$$p(\boldsymbol{H}_k \mid \boldsymbol{A}_k, \boldsymbol{H}_{k-1}, \sigma_h^2) = \prod_{k=2}^{K}\prod_{i=1}^{D}\prod_{j=1}^{N}[\mathcal{N}(\boldsymbol{H}_{k,ij} \mid \boldsymbol{A}_{k,i}^T \boldsymbol{H}_{k-1,j}, \sigma_h^2)] \tag{5}$$

where $\sigma_h$ is the prior standard variance of the factor-user latent variable and the matrix $\boldsymbol{A}_k$ is different for each time period.

LDS cannot ensure the non-negativity of each $H_k$, which is required by NMF. To avoid negativity, we propose the DMNMF model to combine multi-task NMF and LDS into a unified model using the log of the posterior distribution over the item-factor matrix, factor-user matrices and state transition matrix, as defined below:

$$\begin{aligned}\ln p(\boldsymbol{W}, \boldsymbol{H}_k, \boldsymbol{A} \mid \boldsymbol{X}, \sigma^2, \sigma_W^2, \sigma_H^2) \quad = \quad & -\frac{1}{2\sigma^2}\sum_{k=1}^{K}\sum_{i=1}^{N}\sum_{j=1}^{M}(\boldsymbol{X}_{ij} - \boldsymbol{W}_i^T \boldsymbol{H}_j)^2 \\ & -\frac{1}{2\sigma_h^2}\sum_{i=2}^{K}\sum_{i=1}^{D}\sum_{j=1}^{N}(\boldsymbol{H}_{k,ij} - \boldsymbol{A}_{k,i}^T \boldsymbol{H}_{k-1,j})^2 + Const.,\end{aligned} \tag{6}$$

where *Const*. is a constant that does not depend on the parameters. For the sake of simplicity, we fix $\boldsymbol{A}$ to avoid over-fitting and to capture interesting and important trends in the period. Maximizing the log-posterior with hyperparameters (i.e., the observation noise variance and prior variances) is equivalent to minimizing the sum-of-squared-errors cost function with quadratic regularization terms:

$$\begin{aligned} C(\hat{\boldsymbol{W}}, \hat{\boldsymbol{A}}, \hat{\boldsymbol{H}}_k) \quad = \quad & \min_{\boldsymbol{W},\boldsymbol{H}_k,\boldsymbol{A}} \frac{1}{2}\left(\sum_{k=1}^{K} \parallel \boldsymbol{V}_k - \boldsymbol{W}\boldsymbol{H}_k \parallel_F^2 + \lambda \sum_{k=2}^{K} \parallel \boldsymbol{H}_k - \boldsymbol{A}\boldsymbol{H}_{k-1} \parallel_F^2\right) \\ & \text{s.t.} \quad \boldsymbol{W} \geq 0, \boldsymbol{A} \geq 0, \boldsymbol{H}_k, \geq 0 \end{aligned} \tag{7}$$

where $C$ is the cost function, $\lambda = \sigma^2/\sigma_{\boldsymbol{h}}^2$, and $\|.\|_F^2$ denotes the Frobenius norm. The first item in the cost function represents the minimization of the errors between the observed data and the recovered data using the NMF $K$ times. The second item represents the minimization of the errors that occur while estimating the transition matrix over $K-1$ transitions.

Once the state transition matrix $\boldsymbol{A}$ is obtained, the coefficient matrix of the next time period $\boldsymbol{H}_{K+1}$ can be obtained by $\boldsymbol{H}_{K+1} = \boldsymbol{A}\,\boldsymbol{H}_K$. Subsequently, we obtain the prediction function $\boldsymbol{X}_{K+1,ij} = \delta(\boldsymbol{W}\,\boldsymbol{A}\,\boldsymbol{H}_K)_{i,j} = \{1, (\boldsymbol{W}\,\boldsymbol{A}\,\boldsymbol{H}_K)_{i,j} > \tau; 0, \text{otherwise.}\}$, where $\tau$ is a threshold. Here, $\boldsymbol{X}_{K+1,ij} = 1$ indicates that item $j$ most likely would be selected by user $i$ in the time period $K+1$.

## Algorithm for DMNMF

Several algorithms have been proposed to solve the NMF problem, including multiplicative update (MU) [35], alternating least squares (ALS) [36], and projected gradient (PG) method [37], among others. Following [38], we adopt the MU algorithm to solve the optimization problem (7). To provide the MU algorithm for DMNMF, we first define the notations of two matrix operations $\circledast$ and $\oslash$, which represent element-wise multiplication and division, respectively. Let us consider the Karush-Kuhn-Tucker (KKT) conditions of (7), i.e.,

$$\begin{cases} \boldsymbol{W} \geq 0 & (8) \\ \boldsymbol{A} \geq 0 & (9) \\ \boldsymbol{H}_k \geq 0, \qquad k = 1, \ldots, K & (10) \\ \nabla_{\boldsymbol{W}} C \geq 0 & (11) \\ \nabla_{\boldsymbol{A}} C \geq 0 & (12) \\ \nabla_{\boldsymbol{H}_k} C \geq 0, \qquad k = 1, \ldots, K & (13) \\ \boldsymbol{W} \circledast \nabla_{\boldsymbol{W}} C = 0 & (14) \\ \boldsymbol{A} \circledast \nabla_{\boldsymbol{A}} C = 0 & (15) \\ \boldsymbol{H}_k \circledast \nabla_{\boldsymbol{H}_k} C = 0, \quad k = 1, \ldots, K & (16) \end{cases}$$

where

$$\nabla_{\boldsymbol{W}} C = \sum_{k=1}^{K} (\boldsymbol{W}\boldsymbol{H}_k\boldsymbol{H}_k^{\mathrm{T}} - \boldsymbol{V}_k\boldsymbol{H}_k^{\mathrm{T}}) \tag{17}$$

and

$$\nabla_{\boldsymbol{A}} C = \sum_{k=2}^{K} (\boldsymbol{A}\boldsymbol{H}_{k-1}\boldsymbol{H}_{k-1}^{\mathrm{T}} - \boldsymbol{H}_k\boldsymbol{H}_{k-1}^{\mathrm{T}}). \tag{18}$$

Substituting (Eq 17) into (Eq 14), we have

$$\boldsymbol{W} \circledast \left( \sum_{k=1}^{K} (\boldsymbol{W}\boldsymbol{H}_k\boldsymbol{H}_k^{\mathrm{T}}) \right) = \boldsymbol{W} \circledast \left( \sum_{k=1}^{K} (\boldsymbol{V}_k\boldsymbol{H}_k^{\mathrm{T}}) \right). \tag{19}$$

Then, the MU rule for $\boldsymbol{W}$ is derived, i.e.,

$$\boldsymbol{W} \leftarrow \boldsymbol{W} \circledast \left( \sum_{k=1}^{K} (\boldsymbol{V}_k\boldsymbol{H}_k^{\mathrm{T}}) \right) \oslash \left( \sum_{k=1}^{K} (\boldsymbol{W}\boldsymbol{H}_k\boldsymbol{H}_k^{\mathrm{T}}) \right). \tag{20}$$

Likewise, we can obtain the MU rule for $\boldsymbol{A}$ via (Eq 15) and (Eq 18).

$$\boldsymbol{A} \leftarrow \boldsymbol{A} \circledast \left( \sum_{k=2}^{K} (\boldsymbol{H}_k\boldsymbol{H}_{k-1}^{\mathrm{T}}) \right) \oslash \left( \sum_{k=2}^{K} (\boldsymbol{A}\boldsymbol{H}_{k-1}\boldsymbol{H}_{k-1}^{\mathrm{T}}) \right). \tag{21}$$

There are three derived cases for $\boldsymbol{H}_k$. The first case is for $k = 1$,

$$\nabla_{\boldsymbol{H}_1} C = \boldsymbol{W}^{\mathrm{T}}(\boldsymbol{W}\boldsymbol{H}_1 - \boldsymbol{V}_1) + \lambda \boldsymbol{A}^{\mathrm{T}}(\boldsymbol{A}\boldsymbol{H}_1 - \boldsymbol{H}_2) \tag{22}$$

the second case for $k \in [2, K - 1]$,

$$\underset{2 \leq k \leq K}{\nabla_{\mathbf{H}_k}} C = \boldsymbol{W}^{\mathrm{T}}(\boldsymbol{W}\boldsymbol{H}_k - \boldsymbol{V}_k) + \lambda \boldsymbol{A}^{\mathrm{T}}(\boldsymbol{A}\boldsymbol{H}_k - \boldsymbol{H}_{k+1}) + \lambda(\boldsymbol{H}_k - \boldsymbol{A}\boldsymbol{H}_{k-1}) \tag{23}$$

and the third case is for $k = K$

$$\nabla_{H_K} C = W^{\mathrm{T}}(WH_K - V_K) + \lambda(H_K - AH_{K-1}). \tag{24}$$

Similarly, we can obtain the MU rule for $H_k$ via (Eq 16) and Eqs (22)(24) (23).

$$H_1 \leftarrow H_1 \circledast (W^{\mathrm{T}} V_1 + \lambda A^{\mathrm{T}} H_2) \oslash (W^{\mathrm{T}} WH_1 + \lambda A^{\mathrm{T}} AH_1) \tag{25}$$

$$H_k \leftarrow H_k \circledast (W^{\mathrm{T}} V_k + \lambda A^{\mathrm{T}} H_{k+1} + \lambda AH_{k-1}) \oslash (W^{\mathrm{T}} WH_k + \lambda A^{\mathrm{T}} AH_k + \lambda H_k) \tag{26}$$

$$H_K \leftarrow H_K \circledast (W^{\mathrm{T}} V_K + \lambda AH_{K-1}) \oslash (W^{\mathrm{T}} WH_K + \lambda H_K) \tag{27}$$

**Theorem 1** *The cost function C in* (Eq 7) *is non-increasing under the update rules* (Eq 20) (Eq 21) *and* Eqs (25)–(27).

The proof is given in the appendix.

Based on the fact that $C$ is non-increasing, the convergence of the MU algorithm is guaranteed. The detailed steps of the MU algorithm for the recommender are listed in Algorithm 1.

```
Algorithm 1 The MU algorithm for DMNMF
Input:
    Data matrices to be factorized $V_1, \ldots, V_K \in R_+^{N\times M}$,
    Dimension size $D$,
    Regularization parameter $\lambda$,
    Threshold $\tau$.
Output:
    Recommendation set $X$.
1: Initialize $W$, $H_1$, ..., $H_K$ and $A$ with random numbers in [0, 1].
2: repeat
3:      Fix $H_1$, ..., $H_K$ and $A$, update $W$ with MU rule (Eq 20)
4:      Fix $H_1$, ..., $H_K$ and $W$, update $A$ with MU rule (Eq 21)
5:      Fix $W$, $H_2$, ..., $H_K$ and $A$, update $H_1$ with MU rule (Eq 25)
6:      Fix $W$, $H_1$, ..., $H_{K-1}$ and $A$, update $H_K$ with MU rule (Eq 27)
7:      for $k = 2$ to $K-1$
8:        Fix $W$, $A$, $H_1$ and $H_K$, update $H_k$ with MU rule (Eq 26).
9:      end for
10: until the maximum number of iterations has been reached, or the change of
cost function (7) in this iteration is less than a predefined threshold
```

Eq (20) costs $O(KMND)$ time, Eq (21) costs $O(KD^2 N)$ time, and Eq (26) costs $O((K{-}2)(MND))$ time. In summary, the total time complexity of Algorithm 1 is $O(TKMND)$, where $T$ is the number of iterations and is often set as a constant.

## Results

In this section, we validate the effectiveness of DMNMF by comparing with NMF, HMM-CF and DMF-IA using two datasets, i.e., Netflix [39] and Last.fm [40], obtained from real-world data. Below, we discuss how the training sets and test sets are constructed and how to measure the performances of the algorithms before reporting the results. Each algorithm is trained on data up to a certain time period $K$. These algorithm tasks of algorithms are then used to predict what each user will select in time period $K + 1$.

Table 1. DATASET SUMMARY.

| Dataset | #users | #unique items | sparsity | Time span |
|---|---|---|---|---|
| Netflix | 1,015 | 10,000 | 99.6% | Jan.2004–Nov.2005 |
| Last.fm | 992 | 5,000 | 99.3% | Jan.2007–Nov.2008 |

doi:10.1371/journal.pone.0135090.t001

## Data Set

*Netflix Dataset*. Netflix has made available a dataset containing over 100 million ratings, containing 17,770 movies and approximately 480,000 users. The dataset consists of users' ratings for movies along with the timestamp of the rating and spans 86 months. Using this dataset, we predict which movies a target user will rate in a given test period. We define a month as the time period and select 24 months as the time span of the dataset, which divides the temporal training sets into 23 matrices and uses the $24^{th}$ month as the test set. For a test set, we consider predicting which movies the user will rate as predicting which movies he/she will watch. To keep the training dataset size manageable, we construct matrices $\boldsymbol{V}^{train}$ by reading the first 10,000 movie record files and selecting users who have rated at least 500 movies. The process for creating the training dataset is outlined as follows,

$$\left\{ \forall \text{user}_j : \left( \sum_{k=1}^{23} \sum_{i=1}^{10000} \boldsymbol{V}_{i,j}^{train(k)} \geq 500 \right) \right\} \tag{28}$$

which results in a dataset with 1,015 users and 10,000 movies.

*Last.fm Dataset*. Last.fm is an Internet-based personalized radio station and music recommendation system. When the users of the service listen to music through a supported music player, Last.fm collects their music listening behavior. The data is used by Last.fm to make personalized music recommendations for their online radio station. This dataset contains time stamped records of users' music listening activity. It has 992 users and 177,000 artists. The process of constructing the Last.fm training and test datasets is similar to that of Netflix.

We denote the sparsity of the dataset using the formula $sparsity = \frac{\sum_{i=1}^{M} (Item_i)}{N \times M}$, where $N$ is the total number of users, $M$ is the total number of items and $Item_i$ is the number of item $i$ that is selected by users. Summaries of the two training set of two datasets are shown on Table 1.

## Comparison

In this experiment, the following methods are compared:

1. NMF: As a static algorithm, NMF is the baseline for experiments. We feed NMF using an accumulated matrix $\boldsymbol{X}^{*}$, which sums up the frequencies of user-selected items from all time. After matrix completion by multiplying the two factorized matrices, the top-n sorted elements in the reconstructive matrix $\boldsymbol{X}$ with values greater than zero are recommended.

2. HMM-CF: The estimated HMM-CF with data observed up to $k$ can be used to compute the latent class distribution for each user in time period $k + 1$ and then compute the distribution over the observation of articles in time period $k + 1$. The probability that the item $i$ will be observed in $k + 1$ can be computed as $P\left(i \in I_u^{k+1}\right) = \sum_{d=1}^{D} P\left(Z_u^{k+1} = d\right) P\left(i \in I_u^{k+1}; a_d, b_d, \ \boldsymbol{\theta}_d\right)$. Then, the items that are most likely

to be observed in period $k + 1$ can be recommended to the user [16]. HMM-CF assumes that the distribution of how many items will be selected by a certain user in a month is a specific negative binomial distributions (NBD); therefore, the recommended number taken from the NBD for each user is a parameter, that is arbitrarily set by selecting each user's top 5 or top 10 highest scoring items to recommended.

3. DMF-IA: According to the literature [23], DMF-IA is the variant of DMF that has the best performance. We first generate a basis matrix W using NMF; then, we use Kalman filtering to generate the fixed dynamic matrix $\boldsymbol{A}$. Finally, the top-n sorted elements in the reconstructive matrix $\boldsymbol{X}$, $\boldsymbol{X} = \boldsymbol{W}\,\boldsymbol{A}\,\boldsymbol{V}_K$ with values greater than zero are recommended.
4. DMNMF: We run the DMNMF algorithm as described above. In DMNMF, the sorted elements in the reconstructive matrix $\boldsymbol{X} = \boldsymbol{W}\,\boldsymbol{A}\,\boldsymbol{H}_K$ with values are greater than the threshold are recommended.

The performances of all of the algorithms are measured by their precision and recall scores. Only the items that user $i$ actually selected in the time period $K + 1$ are considered correct recommendations. The precision, $P$, of the algorithm is the fraction of the recommended set that is correct and is defined as below:

$$P = \frac{\sum_{i=1}^{L} hitItems_i}{\sum_{i=1}^{L} pItems_i}. \tag{29}$$

where $L$ is the number of all users who are in the recommended set, $pItems_i$ refers to the number of items recommended to user $i$, $hitItems_i$ refers to the number of items that are actually selected from the recommended set. The recall, $R$, of the algorithm is the fraction of the correct set that is recommended and is defined as below:

$$R = \frac{\sum_{i=1}^{L} hitItems_i}{\sum_{i=1}^{B} bItems_i} \tag{30}$$

where $B$ is the number of all users who in the correct set and $bItems_i$ refers to the number of items that the user $i$ actually selected from the correct set. If more items are recommended, the precision will decrease, but the recall will increase. The harmonic mean is called the $F1$ score, see (Eq 31). The higher the $F1$ score is, the better the prediction performance is [41].

$$F1 = \frac{2 \times P \times R}{P + R} \tag{31}$$

To reduce the effect of randomness, we repeat these trials 500 times and compare the algorithms based on their average performances, as shown in Table 2. DMNMF significantly outperforms the other algorithms mainly because it captures the temporal changes in user preference in the unified model.

We are not only interested in precision and recall measures at a threshold or top-N quality of the recommended items but also the quality of the algorithms over the entire test dataset whose ranking score is produced by the algorithms. A receiver operating characteristic (ROC) curve is an intuitive way to compare multiple algorithms [42]. To draw an ROC curve, we split the sorted items score every 5 percentage and calculate the fraction of incorrect items recommended (false positive rate) and the fraction of correct items recommended (true positive rate). An ROC curve is a two-dimensional depiction of the prediction performance. We calculate the area-under-the-curve (AUC) to reduce the ROC performance to a single scalar value [43]. To generate confidence intervals for the AUC, we generate each ROC point 10 times and

**Table 2. PERFORMANCE COMPARISON AMONG DIFFERENT ALGORITHMS.**

| Algorithms | Netflix | | | | Last.fm | | | |
|---|---|---|---|---|---|---|---|---|
| | *P* | *R* | *F1* | *AUC ** | *P* | *R* | *F1* | *AUC ** |
| NMF | 0.1277 | 0.0370 | 0.0574 | 0.623[0.516,0.730] | 0.1705 | 0.1467 | 0.1577 | 0.672[0.579,0.765] |
| HMM-CF | 0.1264 | 0.0585 | 0.080 | 0.703[0.621,0,785] | **0.2919** | 0.1573 | 0.2044 | 0.762[0.684,0.840] |
| DMF-IA | 0.1118 | 0.0556 | 0.0743 | 0.677[0.577,0.777] | 0.2109 | 0.1559 | 0.1793 | 0.780[0.700,0.86] |
| DMNMF | **0.1544** | **0.0834** | **0.1083** | **0.810[0.731, 0.831]** | 0.2644 | **0.1786** | **0.2312** | **0.896[0.871, 0.901]** |

*:95% confidence intervals of AUC.

doi:10.1371/journal.pone.0135090.t002

calculate the 95% confidence interval of the AUC using the method in [44]. According to Table 2, the average ROCs and the AUCs are shown in Fig 4. Fig 4 shows that DMNMF outperforms the other algorithms.

To obtain the runtime of these algorithms in Fig 4, when running the Netflix example, our new model takes 629s, which is comparable to HMM-CF (713s) and slightly slower than NMF (273s) and DMF-IA(395s).

## Parameter Learning for DMNMF

In our algorithm, there are three parameters: the latent factor sizes $D$, regularization parameter $\lambda$ and threshold $\tau$ in the DMNMF model. To choose appropriate and robust parameters for the model, we propose a strategy to estimate a stable scope of parameters for the time-serial datasets. Similar to the idea of n-fold cross-validation, we split the original dataset into 12 time-continuous subsets for parameter validation. One subset contains 11 continuous month data as training sets and the following one month data as validation set. For example, for the first

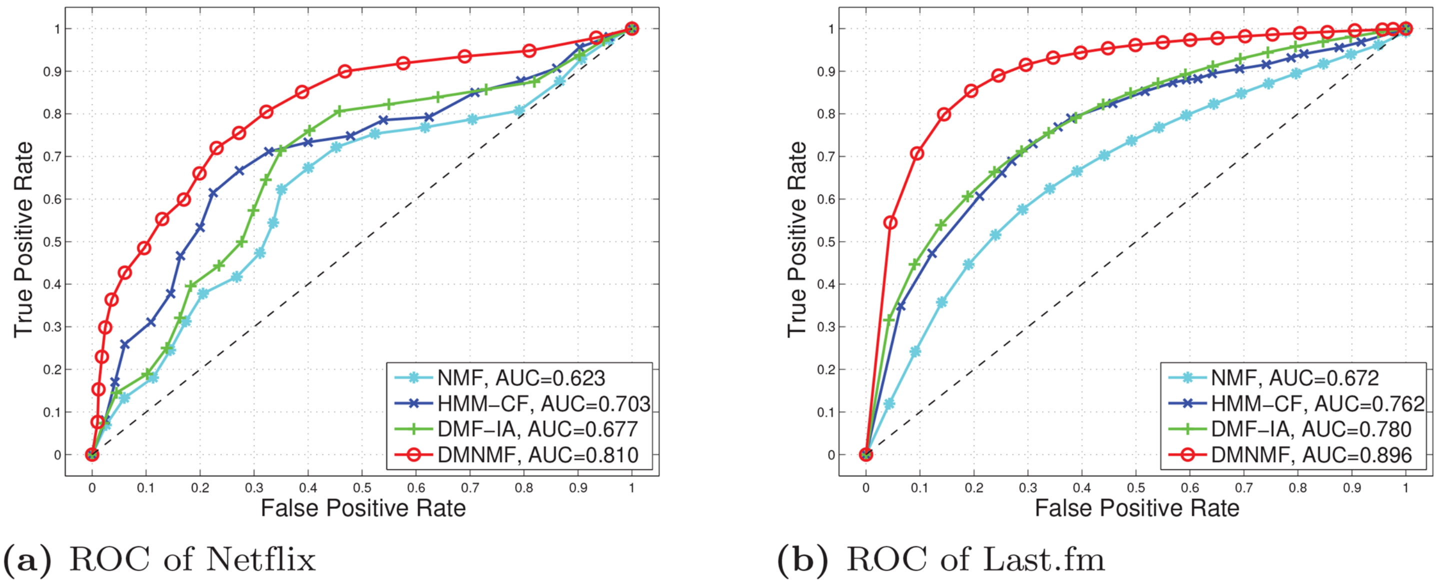


**Fig 4. The ROC curves for evaluating the quality of the algorithms over the entire test datasets.**

doi:10.1371/journal.pone.0135090.g004

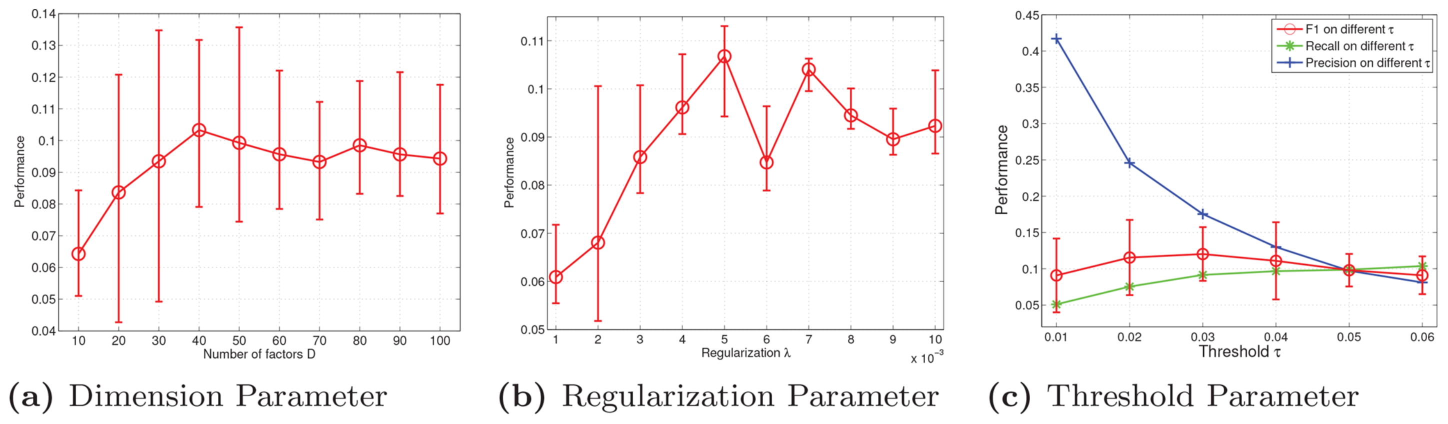


**Fig 5. Performance values with different key parameters on Netflix.**

doi:10.1371/journal.pone.0135090.g005

subset, the data set from the first month to the 11$^{th}$ month are used as the training sets and the 12$^{th}$ month is treated as the validation set, and for the second subset, the data set from the second to the 12$^{th}$ month are used as the training sets ant the 13$^{th}$ month is used as validation, and so on. As the original dataset is spanned of 24 months, we obtained 12 validation sets. The performances were then averaged over these validation sets.

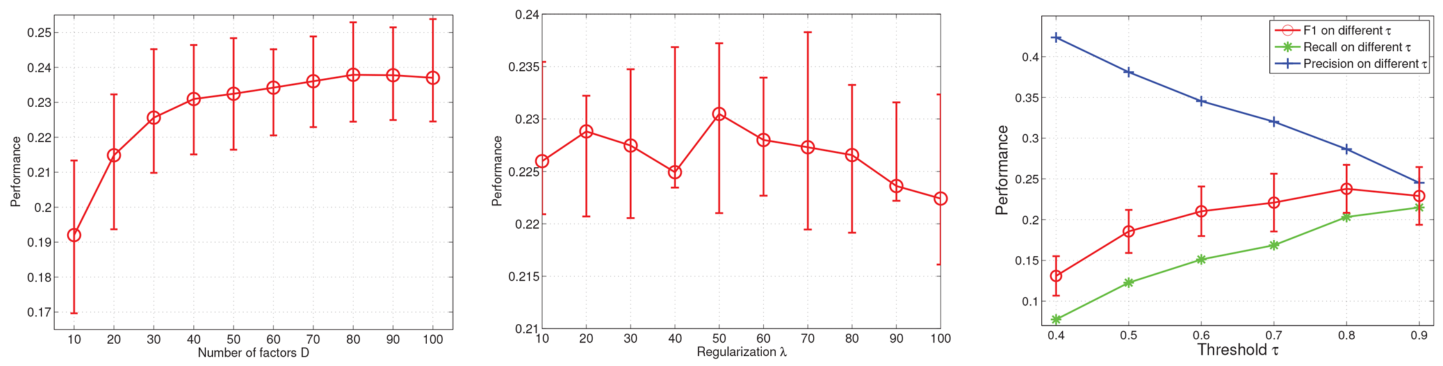
For the sake of conciseness, we use the Netflix dataset as an example to demonstrate how the parameters were determined. Following the settings in the BPMF (Bayesian Probabilistic Matrix Factorization) model [7], which was run on the Netflix dataset, we set the $D$ over the interval [10, 100] with step sizes of 10. As show in Fig 5(a), the predictive performance achieved the optimal performance when $D = 40$. As the number of latent factors increased, overfitting occurred.

The role of regulation $\lambda$ is to balance the multi-task NMF term and LDS term in the cost function. Similar to [6], $\lambda = \sigma^2/\sigma_{\boldsymbol{h}}^2$, indicating the ratio of the standard deviation of the observation data to the prior standard deviation of the factorized factor-user. Here, we choose $\lambda$ from 0.001 to 0.01 with a step size of 0.001. As shown in Fig 5(b), $\lambda$ is optimal at 0.005.

The threshold parameter $\tau$ controls the number of recommended items. The smaller the $\tau$ is, the larger the number of recommended items is. For example, when $\tau = 0.01$, the number of predicting items is 202533 and the number of hitting items is 85813. Although the precision of the algorithm is 42.37%, each user is recommended over 200 items, which is called *over-recommended* in the recommender. Whereas when $\tau = 0.04$, the number of predicting items is 22298 and the number of hitting items is 2898. Although the precision decreases substantially, only 22 items on average are recommended to each user, which results in the recommender providing a better user experience. We choose $\tau$ from 0.01 to 0.1 with a step size of 0.01. As shown in Fig 5(c), the threshold is optimal at 0.03. The confidence intervals of the precision and recall are omitted from Fig 5(c) for the sake of clarity.

The parameter learning for the Last.fm dataset is very similar to that of the Netflix dataset, which is shown in Fig 6.

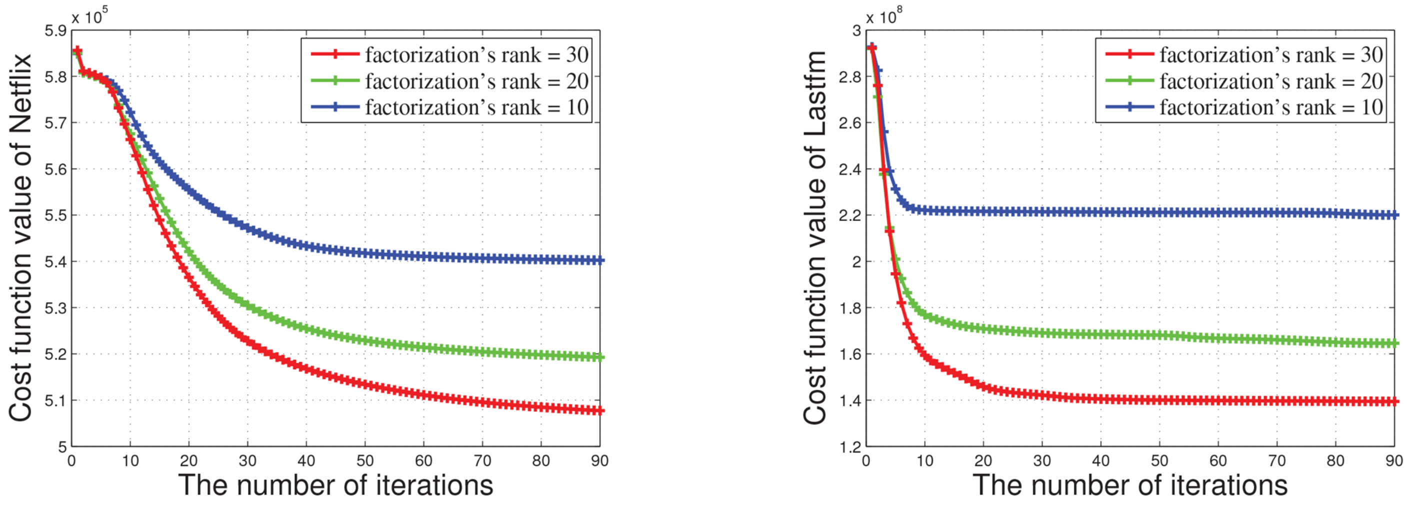
We investigate the convergence of our DMNMF. Fig 7 shows the convergence curve of DMNMF and its cost function. The values of the cost function $C$ with factorization ranks of 10, 20 and 30 were plotted. As shown in Fig 7, the non-increasing nature of $C$ is obvious, and it drops very fast after a few iterations. However, the DMNMF model cannot ensure that the



**(a)** Dimension Parameter **(b)** Regularization Parameter **(c)** Threshold Parameter

**Fig 6. Performance values with different key parameters on Last.fm.**

doi:10.1371/journal.pone.0135090.g006

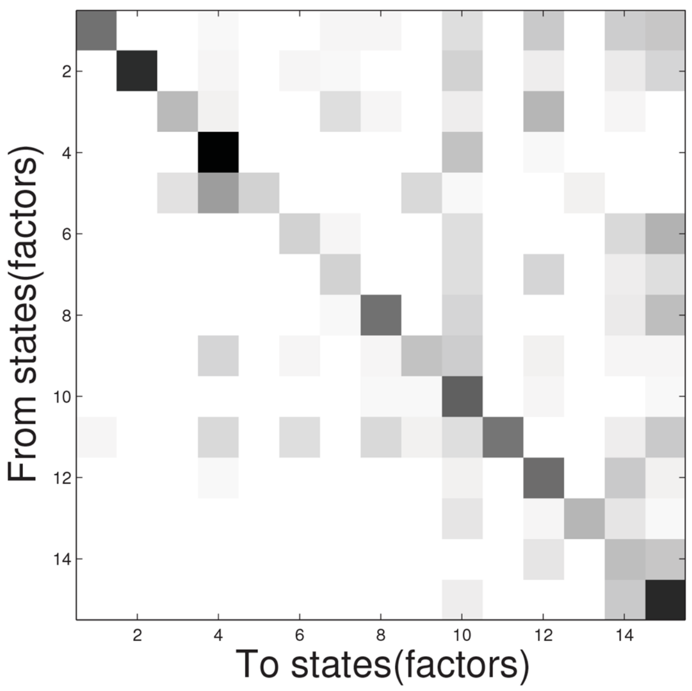


**(a)** The non-increasing nature of cost function for Netflix. **(b)** The non-increasing nature of cost function for Last.fm

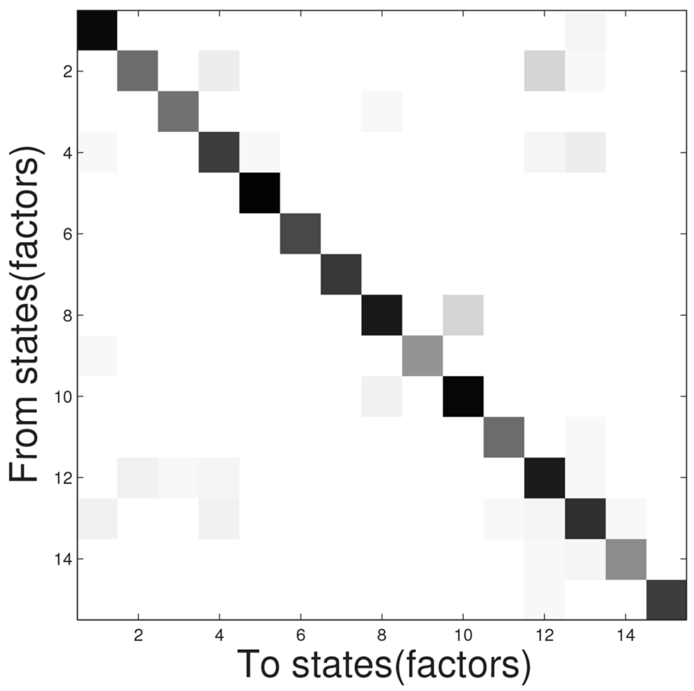
**Fig 7. The non-increasing nature of cost function at various ranks.**

doi:10.1371/journal.pone.0135090.g007

global minimum of the cost function is obtained. Therefore, there may be multiple local minimums, which depends on the initial points. Nevertheless, our numerical experiments showed that different initial values generate very similar results, which implies that the initial value might have only a small impact on the performance of the algorithm. Because the DMNMF cannot ensure that the global minimum is obtained when starting from a random initial condition, we examine the effects of the initial points on the final optimal solution. We randomly chosen the initial points 500 times to obtain the optimal solution of the DMNMF model. After initializing $\boldsymbol{W}, \boldsymbol{H}_1, \ldots, \boldsymbol{H}_K$ and $\boldsymbol{A}$ with random numbers in [0, 1], we obtain the optimal factored matrices corresponding to the minimum value of the object function $C$.



(a) The state transition matrix of Netflix



(b) The state transition matrix of Last.fm

**Fig 8. The graph of state transition matrix.**

doi:10.1371/journal.pone.0135090.g008

## Case Study

A main feature of the DMNMF formulation is the use of dynamic matrix $\boldsymbol{A}$ to capture the evolution of a user's latent factors $h_{i,\,k}$ from one time step to the next time step. The latent state at $k$ is given by $h_{i,\,k} = \boldsymbol{A}h_{i,\,k-1}$. The $k^{th}$ factor in $h_{i,\,k}$ is derived from the dot product of the $k^{th}$ row of $\boldsymbol{A}$ and $h_{i,\,k-1}$. The largest value in the $k^{th}$ row of $\boldsymbol{A}$ plays an important role in accounting for the value of the $k^{th}$ latent factor in $h_{i,\,k}$. The state transition matrix of the two real datasets is shown in Fig 8. In Fig 8, the darker the state, the larger the value of the element in $\boldsymbol{A}$. To further illustrate the effectiveness of our DMNMF model, a case study is demonstrated using the Netflix dataset. For the sake of the clear visualization of the state transition matrix, we choose a dimension size of 15. We randomly select a user whose ID is 1371451(we name him 'John') and explain the evolution of his latent factors from the first month to the $23^{th}$ month (Jan. 2004 to Nov. 2005). We extract a certain column corresponding to John from every factor efficient $\boldsymbol{H}_k$ and combine these into a matrix, then, we plot the matrix as an evolution of the preference profile that belongs to John. As shown in Fig 9, the larger the value of the element in the column, the darker the block and the more likely the corresponding preference factor. As time passes, the user's changing preferences are represented by the largest element in the column. We find that John became active in selecting movies in the first 3 months and was dormant for a while. Then John became active in the $23^{rd}$ month, which is consistent with his behavior on rating movies in the training set.

We regard a factor as a 'topics' describing the user's preference regarding certain properties of movies and select certain movies from corresponding topics that appear more black than others in Fig 9. For example, the top 10 largest elements (movies) are extracted from the

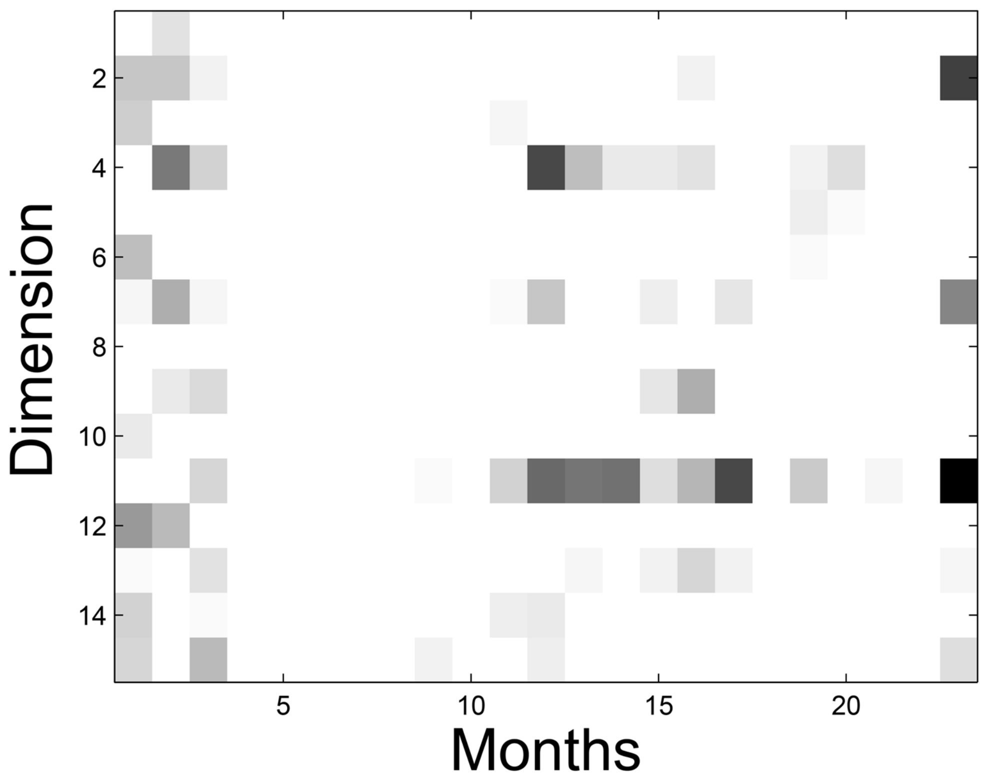


**Fig 9. John's preference profile.**

doi:10.1371/journal.pone.0135090.g009

corresponding column in the basis matrix $\boldsymbol{W}$, as shown in Table 3. Factor 2, factor 7, factor 11 can be interpreted as 'Action', 'Romantic comedies', 'Family', respectively. We demonstrate an example in which John shifted his preference from 'Action' to 'Romantic comedies' from Jan. 2004 to Nov. 2005. The $\boldsymbol{W}_{2,1}$ entry in the $1^{st}$ column of the preference graph has the second highest value of 1.73, whereas the others have a mean value of 0.12; thus, we infer that John was interested in topic 2. Likewise, the $\boldsymbol{W}_{7,23}$ entry in the $23^{rd}$ column of the preference graph has the highest value of 2.09, which indicates that John was interested in topic 7.

**Table 3. LATENT TOPICS.**

| *Factor 2(Action)* | *Factor 7(Romantic comedies)* | *Factor 11(Family)* |
|---|---|---|
| Alien | The Object of My Affection | Curly Sue |
| I, Robot | Drive Me Crazy | Look Who's Talking Too |
| Lord of the Rings: The Two Towers | Mickey Blue Eyes | Free Willy |
| Rambo: First Blood | Fools Rush In | Beethoven |
| Kill Bill | Down to You | Ray |
| Speed | Simply Irresistible | Finding Neverland |
| Star Trek | The Bachelor | The Forgotten |
| The Lost World: Jurassic Park | Green Card | Shark Tale |
| The Fifth Element | Mrs. Winterbourne | Napoleon Dynamite |
| X2: X-Men United | Picture Perfect | Junior |

doi:10.1371/journal.pone.0135090.t003

## Discussion

In collaborative filtering, item selection prediction is applied more widely than item rating prediction. This paper proposes an effective unified model called DMNMF to discover the latent factors behind users' selection behaviors and capture the transition of user preference in latent factor space. We develop an MU algorithm to solve DMNMF. Experimental results on popular CF databases demonstrate that our proposed algorithm outperforms NMF, HMM and DMF as well as their extensions. As noted in the results section, our model has a larger time cost than the traditional NMF and DMF-IA models, mainly due to the longer time required for each iteration in our model compared with the others. Finally, our model can be used as a novel form of dynamic topic models for tracking the evolution of user preferences over time. In the future, we will develop a hybrid method that integrates the tracking of evolution of user preferences and the similarities between user preferences.

## Appendix

Proof of Multiplicative Update Rules (Eq 25)–(Eq 27)

As the cost function (7) is separable in the columns of $\boldsymbol{H}_k$, we focus on one column of $\boldsymbol{H}_k$ alone, which is denoted by $\boldsymbol{h}_{(k)}$. To prove the non-increasing property of (Eq 26), an auxiliary function $G(\boldsymbol{h}_{(k)}, \boldsymbol{h}^t_{(k)})$ is defined.

**Lemma 1** *The objective function C does not increase under the following update rule*

$$\boldsymbol{h}^{t+1}_{(k)} = \arg\min_{\boldsymbol{h}_{(k)}} G(\boldsymbol{h}_{(k)}, \boldsymbol{h}^t_{(k)}) \tag{32}$$

*where* $G(\boldsymbol{h}_{(k)}, \boldsymbol{h}^t_{(k)})$ *is an auxiliary function satisfying* $G(\boldsymbol{h}_{(k)}, \boldsymbol{h}_{(k)}) = C(\boldsymbol{h}_{(k)})$ *and* $G(\boldsymbol{h}_{(k)}, \boldsymbol{h}^t_{(k)}) \geq C(\boldsymbol{h}_{(k)})$.

This was proven in [35].

**Lemma 2** *If* $\boldsymbol{Q}(\boldsymbol{h}^t_{(k)})$ *is a diagonal matrix*

$$\boldsymbol{Q}_{ab}(\boldsymbol{h}^t_{(k)}) = \delta_{ab}\Big((\boldsymbol{W}^{\mathrm{T}}\boldsymbol{W} + \lambda\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A} + \lambda\boldsymbol{I})\boldsymbol{h}^t_{(k)}\Big)_a / \boldsymbol{h}^t_{(k),a} \tag{33}$$

*then*

$$\begin{aligned} G(\boldsymbol{h}^t_{(k)}, \boldsymbol{h}_{(k)}) &= C\left(\boldsymbol{h}^t_{(k)}\right) + \left(\boldsymbol{h}_{(k)} - \boldsymbol{h}^t_{(k)}\right)^{\mathrm{T}} \nabla_{\boldsymbol{h}_{(k)}} C\left(\boldsymbol{h}^t_{(k)}\right) + \\ & \frac{1}{2}\left(\boldsymbol{h}_{(k)} - \boldsymbol{h}^t_{(k)}\right)^{\mathrm{T}} \boldsymbol{Q}\left(\boldsymbol{h}^t_{(k)}\right)\left(\boldsymbol{h}_{(k)} - \boldsymbol{h}^t_{(k)}\right) \end{aligned} \tag{34}$$

*is an auxiliary function for* (7).

**Proof**: Since $G(\boldsymbol{h}_{(k)}, \boldsymbol{h}^t_{(k)}) = C(\boldsymbol{h}_{(k)})$ is obviously, we need only show that $G(\boldsymbol{h}^t_{(k)}, \boldsymbol{h}_{(k)}) \geq C(\boldsymbol{h}_{(k)})$. According to Taylor expansion,

$$\begin{aligned} C(\boldsymbol{h}_{(k)}) &= C\left(\boldsymbol{h}^t_{(k)}\right) + \left(\boldsymbol{h}_{(k)} - \boldsymbol{h}^t_{(k)}\right)^{\mathrm{T}} \nabla_{\boldsymbol{h}_{(k)}} C\left(h^t_{(k)}\right) + \\ & \frac{1}{2}\left(\boldsymbol{h}_{(k)} - \boldsymbol{h}^t_{(k)}\right)^{\mathrm{T}} (\boldsymbol{W}^{\mathrm{T}}\boldsymbol{W} + \lambda\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A} + \lambda\boldsymbol{I})\left(\boldsymbol{h}_{(k)} - \boldsymbol{h}^t_{(k)}\right) \end{aligned} \tag{35}$$

we only has to proof

$$G\left(\boldsymbol{h}^t_{(k)}, \boldsymbol{h}_{(k)}\right) - C\left(\boldsymbol{h}_{(k)}\right) = \frac{1}{2}\left(\boldsymbol{h}_{(k)} - \boldsymbol{h}^t_{(k)}\right)^{\mathrm{T}} \boldsymbol{S}\left(\boldsymbol{h}_{(k)} - \boldsymbol{h}^t_{(k)}\right) \geq 0 \tag{36}$$

where

$$\boldsymbol{S} = \boldsymbol{Q}(\boldsymbol{h}^t_{(k)}) - (\boldsymbol{W}^{\mathrm{T}}\boldsymbol{W} + \lambda\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A} + \lambda\boldsymbol{I}) \tag{37}$$

To prove $\boldsymbol{S}$ is semipositive definite, consider the matrix:

$$\boldsymbol{M}_{ab}(\boldsymbol{h}^t_{(k)}) = \boldsymbol{h}^t_{(k),a}(\boldsymbol{S})_{ab}\boldsymbol{h}^t_{(k),b} \tag{38}$$

which is just a rescaling of the components of $\boldsymbol{S}$. Then $\boldsymbol{S}$ is semipositive definite if and only if $\boldsymbol{M}$ is, and we denote

$$\boldsymbol{Z} = \boldsymbol{W}^{\mathrm{T}}\boldsymbol{W} + \lambda\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A} + \lambda\boldsymbol{I} \tag{39}$$

then

$$\begin{aligned} \boldsymbol{v}^{\mathrm{T}}\boldsymbol{M}\boldsymbol{v} &= \sum_{ab}\boldsymbol{v}_a\boldsymbol{M}_{ab}\boldsymbol{v}_b \\ &= \sum_{ab}\boldsymbol{h}^t_{(k),a}(\boldsymbol{Z})_{ab}\boldsymbol{h}^t_{(k),b}\boldsymbol{v}_a^2 - \boldsymbol{v}_a\boldsymbol{h}^t_{(k),a}(\boldsymbol{Z})_{ab}\boldsymbol{h}^t_{(k),b}\boldsymbol{v}_b \\ &= \sum_{ab}(\boldsymbol{Z})_{ab}h^t_{(k),a}h^t_{(k),b}\left[\frac{1}{2}\boldsymbol{v}_a^2 + \frac{1}{2}\boldsymbol{v}_b^2 - \boldsymbol{v}_a\boldsymbol{v}_b\right] \\ &= \frac{1}{2}\sum_{ab}(\boldsymbol{Z})_{ab}h^t_{(k),a}h^t_{(k),b}(\boldsymbol{v}_a - \boldsymbol{v}_b)^2 \\ &\geq 0 \end{aligned} \tag{40}$$

Therefore, $G(\boldsymbol{h}^t_{(k)}, \boldsymbol{h}_{(k)}) \geq C(\boldsymbol{h}_{(k)})$ is proven according to (36).

**Proof**: [Proof of rules (25)–(27)] To minimize the auxiliary function $G$, we set

$$\nabla_{\boldsymbol{h}_{(k)}}G(\boldsymbol{h}^t_{(k)}, \boldsymbol{h}_{(k)}) = \nabla_{\boldsymbol{h}_{(k)}}C(\boldsymbol{h}^t_{(k)}) + \boldsymbol{Q}(\boldsymbol{h}^t_{(k)})\left(\boldsymbol{h}_{(k)} - \boldsymbol{h}^t_{(k)}\right) = 0 \tag{41}$$

Solving $\boldsymbol{h}_{(k)}$, we have

$$\begin{aligned} \boldsymbol{h}_{(k)} &= \boldsymbol{h}^t_{(k)} - \boldsymbol{Q}^{-1}(\boldsymbol{h}^t_{(k)})\nabla_{\boldsymbol{h}_{(k)}}C(\boldsymbol{h}^t_{(k)}) \\ &= \boldsymbol{h}^t_{(k)} - \mathrm{diag}\left(\boldsymbol{h}^t_{(k)} \oslash (\boldsymbol{W}^{\mathrm{T}}\boldsymbol{W} + \lambda\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A} + \lambda\boldsymbol{I})\boldsymbol{h}^t_{(k)}\right)\nabla_{\boldsymbol{h}_{(k)}}C(\boldsymbol{h}^t_{(k)}) \\ &= \boldsymbol{h}^t_{(k)} - \boldsymbol{h}^t_{(k)} \oslash \left(\boldsymbol{W}^{\mathrm{T}}\boldsymbol{W}\boldsymbol{h}^t_{(k)} + \lambda\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{h}^t_{(k)} + \lambda\boldsymbol{h}^t_{(k)}\right) \\ &\quad \circledast \left[\boldsymbol{W}^{\mathrm{T}}\left(\boldsymbol{W}\boldsymbol{h}^t_{(k)} - \boldsymbol{V}_k\right) + \lambda\boldsymbol{A}^{\mathrm{T}}\left(\boldsymbol{A}\boldsymbol{h}^t_{(k)} - \boldsymbol{h}_{(k+1)}\right) + \lambda\left(\boldsymbol{h}^t_{(k)} - \boldsymbol{A}\boldsymbol{h}_{(k-1)}\right)\right] \\ &= \boldsymbol{h}^t_{(k)} \circledast \left(\boldsymbol{W}^{\mathrm{T}}\boldsymbol{V}_k + \lambda\boldsymbol{A}^{\mathrm{T}}\boldsymbol{h}_{(k+1)} + \lambda\boldsymbol{A}\boldsymbol{h}_{(k-1)}\right) \oslash \left(\boldsymbol{W}^{\mathrm{T}}\boldsymbol{W}\boldsymbol{h}^t_{(k)} + \lambda\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{h}^t_{(k)} + \lambda\boldsymbol{h}^t_{(k)}\right) \end{aligned} \tag{42}$$

According to Lemma 1, objective function (7) is non-increasing under the update rule (26). Obviously, objective function (7) is non-increasing under the rules (25) and (27), because they are the special cases of the rule (26).

Proof of Multiplicative Update Rules (20) and (21)

Since the proof of update rule (20) is exactly the same as (21), we only present the proof of update rule (21). As the objective function (7) is separable in the columns of $\boldsymbol{A}$, we focus on one column of $\boldsymbol{A}$ alone, which is denoted by $\boldsymbol{a}$. To prove the non-increasing property of (21), an auxiliary function $G(\boldsymbol{a}, \boldsymbol{a}^t)$ is defined.

**Lemma 3** *If* $\boldsymbol{Q}(\boldsymbol{a}^t)$ *is a diagonal matrix*

$$\boldsymbol{Q}_{ij}(\boldsymbol{a}^t) = \delta_{ij}\left(\sum_{k=2}^{K}((\boldsymbol{a}^t\boldsymbol{H}_{k-1}\boldsymbol{H}_{k-1}^{\mathrm{T}})_i/\boldsymbol{a}_i^t)\right) \tag{43}$$

*then*

$$G(\boldsymbol{a},\boldsymbol{a}^t) = C(\boldsymbol{a}^t) + (\boldsymbol{a}-\boldsymbol{a}^t)^{\mathrm{T}}\nabla_{\boldsymbol{a}}C(\boldsymbol{a}^t) + \frac{1}{2}(\boldsymbol{a}-\boldsymbol{a}^t)^{\mathrm{T}}\boldsymbol{Q}(\boldsymbol{a}^t)(\boldsymbol{a}-\boldsymbol{a}^t) \tag{44}$$

*is an auxiliary function for* (7).

**Proof**: $G(\boldsymbol{a}, \boldsymbol{a}) = C(\boldsymbol{a})$ obviously holds. According to Taylor expansion,

$$C(\boldsymbol{a}) \quad = \quad C(\boldsymbol{a}^t) + (\boldsymbol{a}-\boldsymbol{a}^t)^{\mathrm{T}}\nabla_{\boldsymbol{a}}C(\boldsymbol{a}^t) + \frac{1}{2}(\boldsymbol{a}-\boldsymbol{a}^t)^{\mathrm{T}}\left(\sum_{k=2}^{K}(\boldsymbol{H}_{k-1}\boldsymbol{H}_{k-1}^{\mathrm{T}})\right)(\boldsymbol{a}-\boldsymbol{a}^t) \tag{45}$$

we have

$$G(\boldsymbol{a},\boldsymbol{a}^t) - C(\boldsymbol{a}) = \frac{1}{2}(\boldsymbol{a}-\boldsymbol{a}^t)\boldsymbol{S}(\boldsymbol{a}-\boldsymbol{a}^t)^{\mathrm{T}} \tag{46}$$

where

$$\begin{aligned}\boldsymbol{S} &= \boldsymbol{Q}(\boldsymbol{a}^t) - \sum_{k=2}^{K}(\boldsymbol{H}_{k-1}\boldsymbol{H}_{k-1}^{\mathrm{T}}) \\ &= \sum_{k=2}^{K}\left(\mathrm{diag}\left((\boldsymbol{a}\boldsymbol{H}_{k-1}\boldsymbol{H}_{k-1}^{\mathrm{T}})\oslash\boldsymbol{a}^t\right) - \boldsymbol{H}_{k-1}\boldsymbol{H}_{k-1}^{\mathrm{T}}\right)\end{aligned} \tag{47}$$

We denote

$$\boldsymbol{M}_k = \mathrm{diag}((\boldsymbol{a}\boldsymbol{H}_{k-1}\boldsymbol{H}_{k-1}^{\mathrm{T}})\oslash\boldsymbol{a}^t) - \boldsymbol{H}_{k-1}\boldsymbol{H}_{k-1}^{\mathrm{T}} \tag{48}$$

It was proven in [35] that each $\boldsymbol{M}_k$ is positive semidefinite. Since the sum of positive semidefinite matrices is positive semidefinite, $\boldsymbol{S} = \sum_{k=2}^{K}\boldsymbol{M}_k$ is a positive semidefinite matrix. Therefore, $G(\boldsymbol{a}, \boldsymbol{a}^t) \geq C(\boldsymbol{a})$ is proven according to (46).

**Proof**: [Proof of rules (21)] To minimize the auxiliary function $G$, we set

$$\nabla_{\boldsymbol{a}}G(\boldsymbol{a},\boldsymbol{a}^t) = \nabla_{\boldsymbol{a}}C(\boldsymbol{a}^t) + \boldsymbol{Q}(\boldsymbol{a}^t)(\boldsymbol{a}-\boldsymbol{a}^t) = 0 \tag{49}$$

Solving $\boldsymbol{a}$, we have

$$\begin{aligned}\boldsymbol{a} &= \boldsymbol{a}^t - \boldsymbol{Q}^{-1}(\boldsymbol{a}^t)\nabla_{\boldsymbol{a}}C(\boldsymbol{a}^t) \\ &= \boldsymbol{a}^t - \mathrm{diag}\left(\boldsymbol{a}^t\oslash\sum_{k=2}^{K}(\boldsymbol{a}^t\boldsymbol{H}_{k-1}\boldsymbol{H}_{k-1}^{\mathrm{T}})\right)\nabla_{\boldsymbol{a}}C(\boldsymbol{a}^t) \\ &= \boldsymbol{a}^t - \boldsymbol{a}^t\oslash\sum_{k=2}^{K}(\boldsymbol{a}^t\boldsymbol{H}_{k-1}\boldsymbol{H}_{k-1}^{\mathrm{T}})\circledast\sum_{k=2}^{K}(\boldsymbol{a}^t\boldsymbol{H}_{k-1}\boldsymbol{H}_{k-1}^{\mathrm{T}} - \boldsymbol{H}_{k-1}\boldsymbol{H}_{k-1}^{\mathrm{T}}) \\ &= \boldsymbol{a}^t\circledast\left(\sum_{k=2}^{K}(\boldsymbol{H}_{k-1}\boldsymbol{H}_{k-1}^{\mathrm{T}})\right)\oslash\left(\sum_{k=2}^{K}(\boldsymbol{a}^t\boldsymbol{H}_{k-1}\boldsymbol{H}_{k-1}^{\mathrm{T}})\right)\end{aligned} \tag{50}$$

According to Lemma 1, objective function (7) is non-increasing under the update rule (21).

## Acknowledgments

The author wishes to thank Shaoxiao Jian for useful discussions and comments on the manuscript.

## Author Contributions

Conceived and designed the experiments: BJ YQ MY. Performed the experiments: RN BJ. Analyzed the data: CZ BJ. Contributed reagents/materials/analysis tools: BJ YQ. Wrote the paper: BJ MY RN.

## References

1. Resnick P, Varian HR. Recommender systems. Communications of the ACM. 1997; 40(3):56–58. doi: 10.1145/245108.245121
2. Su X, Khoshgoftaar TM. A survey of collaborative filtering techniques. Advances in artificial intelligence. 2009; 2009:4. doi: 10.1155/2009/421425
3. Adomavicius G, Tuzhilin A. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. Knowledge and Data Engineering, IEEE Transactions on. 2005; 17(6):734–749. doi: 10.1109/TKDE.2005.99
4. Koren Y, Bell R, Volinsky C. Matrix factorization techniques for recommender systems. Computer. 2009;(8: ):30–37. doi: 10.1109/MC.2009.263
5. Koren Y. Collaborative filtering with temporal dynamics. Communications of the ACM. 2010; 53(4):89–97. doi: 10.1145/1721654.1721677
6. Mnih A, Salakhutdinov R. Probabilistic matrix factorization. In: Advances in neural information processing systems; 2007. p. 1257–1264.
7. Salakhutdinov R, Mnih A. Bayesian probabilistic matrix factorization using Markov chain Monte Carlo. In: Proceedings of the 25th international conference on Machine learning. ACM; 2008. p. 880–887.
8. Hofmann T. Latent semantic models for collaborative filtering. ACM Transactions on Information Systems (TOIS). 2004; 22(1):89–115. doi: 10.1145/963770.963774
9. Hofmann T, Puzicha J. Latent class models for collaborative filtering. In: IJCAI. vol. 99; 1999. p. 688–693.
10. Si L, Jin R. Flexible mixture model for collaborative filtering. In: ICML. vol. 3; 2003. p. 704–711.
11. Hofmann T. Probabilistic latent semantic analysis. In: Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence. Morgan Kaufmann Publishers Inc.; 1999. p. 289–296.
12. Blei DM, Ng AY, Jordan MI. Latent dirichlet allocation. the Journal of machine Learning research. 2003; 3:993–1022.
13. Blei DM, Lafferty JD. Dynamic topic models. In: Proceedings of the 23rd international conference on Machine learning. ACM; 2006. p. 113–120.
14. Sun J, Parthasarathy D, Varshney K. Collaborative Kalman Filtering for Dynamic Matrix Factorization. 2012;.
15. Xiong L, Chen X, Huang TK, Schneider JG, Carbonell JG. Temporal Collaborative Filtering with Bayesian Probabilistic Tensor Factorization. In: SDM. vol. 10. SIAM; 2010. p. 211–222.
16. Sahoo N, Singh PV, Mukhopadhyay T. A hidden Markov model for collaborative filtering. MIS Quarterly. 2012; 36(4):1329–1356.
17. Lee DD, Seung HS. Learning the parts of objects by non-negative matrix factorization. Nature. 1999 Oct; 401(6755):788–791. doi: 10.1038/44565 PMID: 10548103
18. Gu Q, Zhou J, Ding CH. Collaborative Filtering: Weighted Nonnegative Matrix Factorization Incorporating User and Item Graphs. In: SDM. SIAM; 2010. p. 199–210.
19. Chen G, Wang F, Zhang C. Collaborative filtering using orthogonal nonnegative matrix tri-factorization. Information Processing & Management. 2009; 45(3):368–379. doi: 10.1016/j.ipm.2008.12.004
20. Zhang S, Wang W, Ford J, Makedon F. Learning from Incomplete Ratings Using Non-negative Matrix Factorization. In: SDM. SIAM; 2006. p. 549–553.
21. Gaussier E, Goutte C. Relation between PLSA and NMF and implications. In: Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval. ACM; 2005. p. 601–602.

**22.** Ding C, Li T, Peng W. Nonnegative matrix factorization and probabilistic latent semantic indexing: Equivalence chi-square statistic, and a hybrid method. In: Proceedings of the national conference on artificial intelligence. vol. 21. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999; 2006. p. 342.

**23.** Chua FCT, Oentaryo RJ, Lim EP. Modeling Temporal Adoptions Using Dynamic Matrix Factorization. In: Data Mining (ICDM), 2013 IEEE 13th International Conference on. IEEE; 2013. p. 91–100.

**24.** Jiang Y, Deng Z, Chung FL, Wang S. Multi-task TSK fuzzy system modeling using inter-task correlation information. Information Sciences. 2015; 298:512–533. doi: 10.1016/j.ins.2014.12.007

**25.** Jiang Y, Chung FL, Ishibuchi H, Deng Z, Wang S. Multitask TSK Fuzzy System Modeling by Mining Intertask Common Hidden Structure. 2014;.

**26.** Evgeniou A, Pontil M. Multi-task feature learning. Advances in neural information processing systems. 2007; 19:41.

**27.** Cichocki A, Zdunek R, Phan AH, Amari Si. Nonnegative matrix and tensor factorizations: applications to exploratory multi-way data analysis and blind source separation. John Wiley & Sons; 2009.

**28.** Xie S, Lu H, He Y. Multi-task co-clustering via nonnegative matrix factorization. In: Pattern Recognition (ICPR), 2012 21st International Conference on. IEEE; 2012. p. 2954–2958.

**29.** Cichocki A, Zdunek R. NMFLAB for signal processing. Laboratory for Advanced Brain Signal Processing, BSI RIKEN, Saitama, Japan www Report, Décember. 2006;.

**30.** Devarajan K, Wang G, Ebrahimi N. A unified statistical approach to non-negative matrix factorization and probabilistic latent semantic indexing. Machine Learning. 2014;p. 1–27.

**31.** Lathia N, Hailes S, Capra L. Temporal collaborative filtering with adaptive neighbourhoods. In: Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval. ACM; 2009. p. 796–797.

**32.** Sun JZ, Varshney KR, Subbian K. Dynamic matrix factorization: A state space approach. In: Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on. IEEE; 2012. p. 1897–1900.

**33.** Li R, Li B, Jin C, Xue X, Zhu X. Tracking User-Preference Varying Speed in Collaborative Filtering. In: AAAI; 2011..

**34.** Badea L. Extracting gene expression profiles common to colon and pancreatic adenocarcinoma using simultaneous nonnegative matrix factorization. In: Proc. Pac. Symp. Biocomput.; 2008. p. 279–290.

**35.** Lee DD, Seung HS. Algorithms for non-negative matrix factorization. In: Advances in neural information processing systems; 2001. p. 556–562.

**36.** Chu M, Diele F, Plemmons R, Ragni S. Optimality, computation, and interpretation of nonnegative matrix factorizations. In: SIAM Journal on Matrix Analysis. Citeseer; 2004..

**37.** Lin CJ. Projected gradient methods for nonnegative matrix factorization. Neural computation. 2007; 19 (10):2756–2779. doi: 10.1162/neco.2007.19.10.2756 PMID: 17716011

**38.** Badea L. Extracting gene expression profiles common to colon and pancreatic adenocarcinoma using simultaneous nonnegative matrix factorization. In: Pacific Symposium on Biocomputing. vol. 290. Citeseer; 2008. p. 279–290.

**39.** Bennett J, Lanning S. The netflix prize. In: Proceedings of KDD cup and workshop. vol. 2007; 2007. p. 35.

**40.** Celma Herrada Ò. Music recommendation and discovery in the long tail. 2009;.

**41.** Herlocker JL, Konstan JA, Terveen LG, Riedl JT. Evaluating collaborative filtering recommender systems. ACM Transactions on Information Systems (TOIS). 2004; 22(1):5–53. doi: 10.1145/963770.963772

**42.** Fawcett T. An introduction to ROC analysis. Pattern recognition letters. 2006; 27(8):861–874. doi: 10.1016/j.patrec.2005.10.010

**43.** Bradley AP. The use of the area under the ROC curve in the evaluation of machine learning algorithms. Pattern recognition. 1997; 30(7):1145–1159. doi: 10.1016/S0031-3203(96)00142-2

**44.** Macskassy S, Provost F. Confidence bands for ROC curves: Methods and an empirical study. Proceedings of the First Workshop on ROC Analysis in AI. August 2004; 2004.