# Time-dependent Models in Collaborative Filtering based Recommender System

Liang Xiang
*Chinese Academy of Sciences*
*Institute of Automation*
*National Laboratory of Pattern Recognition*
*Beijing, China*
*xlvector@gmail.com*

Qing Yang
*Chinese Academy of Sciences*
*Institute of Automation*
*National Laboratory of Pattern Recognition*
*Beijing, China*
*qyang@nlpr.ia.ac.cn*

## Abstract

*In recent years, time information is more and more important in collaborative filtering (CF) based recommender system because many systems have collected rating data for a long time, and time effects in user preference is stronger. In this paper, we focus on modeling time effects in CF and analyze how temporal features influence CF. There are four main types of time effects in CF: (1) time bias, the interest of whole society changes with time; (2) user bias shifting, a user may change his/her rating habit over time; (3) item bias shifting, the popularity of items is changing with time; (4) user preference shifting, a user may change his/her attitude to some types of items. In this work, these four time effects are used in factorized model, which is called TimeSVD. Moreover, many other time effects are used by simple methods. Our time-dependent models are tested on Netflix data from Nov. 1999 to Dec. 2005. Experiment results show that prediction accuracy in CF can be improved significantly by using time information.*

## 1. Introduction

Recommender systems are programs that help us find new information and filter useless information. Collaborative filtering (CF) is an important technology [1] in recommender system that provides personalized recommendations by analyzing historical data of user preferences. In recent years, rapid growth of e-commerce brings an increasing interest in CF and many larger e-commerce web sites have used CF as an important tool in their recommender systems. Examples of these web sites include recommending books at Amazon [13], news at Google [4], movies at Yahoo [15], and CDs at Netflix [3].

In the last two decades, various methods have been developed for CF. Neighborhood methods, also known as kNN, are the first type of algorithms widely used in CF and they include two main type of algorithms: user-based algorithms [18][10] and item-based algorithms [21][5]. The key idea of user-based algorithms is that a user will prefer those items that like-minded users prefer, or dissimilar users don't prefer. Item-Based algorithms assume that a user will prefer similar items that he/she prefer previously. Therefore, the main step in neighborhood method is calculating user-user similarity and item-item similarity. Neighborhood methods are easy to implement and widely used in many recommender systems [13][3][18].

Another type of method often used in CF is matrix factorization, which is also known as latent class model [8][7]. Its key idea is using a low rank matrix to approximate real rating matrix. Singular Value Decomposition (SVD) [14][17][22]is often used to calculate low rank rating matrix, thus these factorized methods are often called SVD. Many researches about CF show factorization based methods can produce more accurate predictions than neighborhood based methods.

Nowadays, many recommender system have collected user preference data for a long time and time information is more and more important in making recommendation. Time information influences CF from four different ways. Firstly, the interest of whole society changes with time. Secondly, rating habit of users change with time. For example, a man may firstly give 5 stars to those items he likes, but after a period of time, he will give no more than 4 stars to those items he enjoys. This means, he is pickier when time goes on. Thirdly, the items' popularity changes with time. A movie may lost popularity because it is too old or get popularity because it wins some awards or its actor becomes popular. The last time effect is, users may change their preferences with time. Many events can cause a user changes his/her preferences. For example, a boy likes watching cartoon when he is young, but he enjoys war films when he grows up. These four time effects are the most important time effects in CF because they represent the main change patterns of users' rating behavior.

Beside four time effects above, there are many other time effects. For example, a user's rating habit is different in different months because of season or festival. Old users and fresh users may have different rating behavior. New movies and old movies may get different ratings. There are many such examples in recommender system. All of these time effects will be used by a simple model in our predictor and

they are denoted by STE (simple time effects).

The main problem is how to use these time effects to build a time-dependent predictor. Previous studies have proposed many ways to use time effects. One approach [2] views these time effects as global effects and uses these effects by simple models, such as linear regression. Another approach [6] is based on neighborhood methods. This approach assumes that recent data is more important to predict users' future preferences than old data. Thus, when calculating item-item similarity, recent rated items will be over-weighted. Moreover, Koren [19][20] have proposed an alternative method which divides rating data into bins by time and trains latent factor model in every bin.

In this paper, four main time effects are modeled by factorization and STE are used as global effects. Our experiments are done on Netflix data [3] which is released by Netflix Prize. It contains more than 100 million movie ratings, and for every rating, it also provides the date when this rating is assigned. Therefore, this data set can be used to evaluate time effects in CF. Our experiment results show, prediction accuracy can be improved by using time-dependent predictors and time effects play important roles in CF.

The remainder of the paper is organized as follows. Section 2 briefly gives some definitions and description of Netflix data. In section 3, we propose our factorized time-dependent model (TimeSVD). Many other time effects will be discussed in Section 4. Section 5 presents our experimental work, analyzes time information in Netflix data, and compares result of temporal models with non-temporal models. In the final section, we make a conclusion and point out directions for future work.

## 2. Preliminaries

We are given ratings about $m$ users and $n$ items. The user set is denoted by $\mathcal{U}$ and item set is denoted by $\mathcal{I}$. $t_u$ denotes the first time when user $u$ assign ratings to items and $t_i$ is the first time when item $i$ is rated. $R(u)$ is the number of items user $u$ have rated and $R(i)$ is the number of users who have rated item $i$. A rating $r_{ui}$ indicates the preference of user $u$ for item $i$, where high values mean strong preference. Netflix data uses 5 stars rating system where 1 star indicating no interest and 5 stars means strong interest. Given a user-item pair $(u, i)$, $t_{ui}$ is the date when user $u$ rated item $i$ and the prediction of $u$'s preference on $i$ is denoted by $\hat{r}_{ui}$. Most of ratings in CF are not known and the observed rating set is defined by $\mathcal{K} = \{(u, i)|r_{ui} \text{ is known}\}$.

### 2.1. Factorized Models

Factorized models, also known as latent factor models [12] and latent class models [8][7], are widely used in CF. Given a rating matrix $R \in \mathbb{R}^{m \times n}$, where $R[u][i] = r_{ui}$, the main idea of factorized models is using a low rank matrix to approximate $R$. That means, finding two matrices $U_{m \times f}, V_{n \times f}$ which can minimize the Frobenius norm $||R - UV^T||_F$. In classical methods, low rank matrix are always found by doing SVD [14][17] or PCA [9] on matrix $R$. However, applying SVD directly in the CF domain raises difficulties due to the high portion of missing ratings. Previous approaches [17] rely on imputation to fill in missing ratings and make the rating matrix dense. However, imputation can be very expensive as it significantly increases the amount of data. In addition, the data may be considerably distorted due to inaccurate imputation. Hence, more recent works suggested modeling directly on the observed ratings, while avoid over-fitting through an regularized model.

Our time-dependent model is based on a regularized SVD model with bias proposed in [12]. In this model, the prediction of user $u$'s preference on item $i$ is made by:

$$\hat{r}_{ui} = \mu + b_u + b_i + p_u^T q_i \tag{1}$$

where $\mu$ is average rating of all knowing ratings, $b_u$ is user bias, $b_i$ is item bias, $p_u \in \mathbb{R}^f$ is user factor vector and $q_i \in \mathbb{R}^f$ is item-factor vector. This model is trained by minimizing following cost function on observed ratings:

$$\begin{aligned} &\min_{b*,p*,q*} \sum_{(u,i)\in\mathcal{K}} (r_{ui} - \mu - b_u - b_i - p_u^T q_i)^2 + \\ &\lambda(\sum_u b_u^2 + \sum_i b_i^2 + \sum_u ||p_u||^2 + \sum_i ||q_i||^2) \end{aligned} \tag{2}$$

where the second term is regularizing term and $\lambda$ is a regularization parameter to avoid over-fitting. This optimization problem can be solved by simple gradient descent method [24]. In the following sections, the model defined in Equation 1 is denoted by RSVD.

### 2.2. The Netflix Data

Our algorithms will be tested on the Netflix data [1] which contains 100,480,507 ratings on a scale 1 to 5 for 17,770 movies and 480,189 users. The total average rating of this data set is 3.6, this means most of users tend to rate these movies they enjoy. This data set not only provides ratings of different movies by different users, but also provides the date when these ratings are assigned. Hence, this data set can help us evaluate time effects in CF. The data contains rating from Nov 11, 1999 to Dec 31, 2005 and the distribution of rating number in different years are shown in Figure 1.

In order to test algorithms, Netflix has provided two test set. One is Probe test set (Probe) and the other is Qualifying test set (Quiz). Probe test set contains 1.4 million user-movie pairs, for which the ratings are known. Qualifying test set contains 2.8 million user-movie pairs, for which the ratings are missing. Both of the two sets contain many ratings by

1. http://www.netflixprize.com/

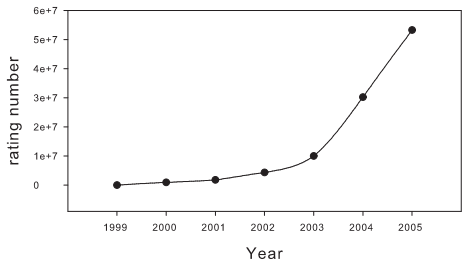


Figure 1. Rating number per year in Netflix data.

users that do not rate much and are hard to predict. The quality of results is measured by root mean squared error (RMSE):

$$RMSE = \frac{\sum_{(u,i)\in TestSet}(r_{ui} - \hat{r}_{ui})^2}{|TestSet|}$$

## 3. Factorized Time-Dependent Model

In recent years, many researchers have focused on time effects in CF and provided many ways to use time information. Ding et al [6] thought a user's recent ratings are more important to predict this user's future preferences than his/her old ratings. Therefore, they used a time-weighted Pearson correlation to measure item-item similarity which will decay the influence of old ratings. Töscher et al [25] pointed out that prediction accuracy can be improved by shrinking rating toward recent ratings. Koren et al [20] introduced three main time effects and used different methods to model these effects. In the following sections, we will give detailed description of our methods.

### 3.1. Time Bias

The interest and habit of the whole society change with time. At different ages, peoples enjoy different things. In recommender system, average rating of all items changes with time. The average rating of all items in time $t$ is denoted by $\bar{r}_t$ and its distribution in Netflix data is shown in Figure 2.

Figure 2 shows $\bar{r}_t$ varies between 3.2 and 3.8. However, it is obvious to see, $\bar{r}_t$ varies between 3.25 and 3.5 before March 2004 and $\bar{r}_t$ varies between 3.55 and 3.75 after March 2004. Many reasons can cause this effect. For example, movies quality is improved after March 2004 or peoples like those movies released after March 2004. This effect is called time bias and in time-dependent model, it is used by adding a scalar $b_t$ to Equation 1:

$$\hat{r}_{ui} = \mu + b_u + b_i + b_t + p_u^T q_i \tag{3}$$

where $t = t_{ui}$.

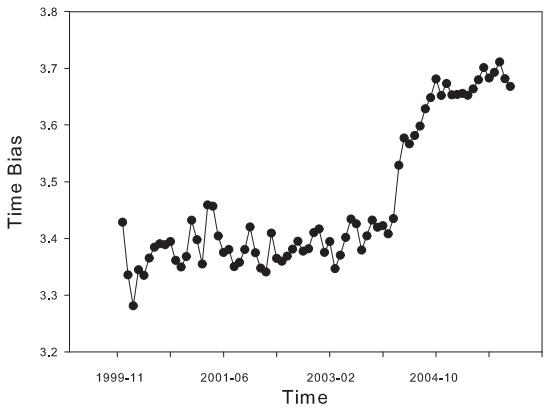


Figure 2. Distribution of $\bar{r}_t$ in Netflix data.

| user id | rated items number |
|---|---|
| 305344 | 17653 |
| 387418 | 17436 |
| 2439493 | 16565 |

Table 1. Basic information of three users.

### 3.2. User Bias Shifting

Users may change their rating habit with time. For example, a user may rate items they like in a period of time and rate items they dislike in another period of time. Furthermore, some user may firstly tend to give no more than 4 stars to the items they enjoy, but after a period of time, they tend to give 5 stars to the items they like. This time effect is called user bias shifting. In order to analyze this time effect, we study how average ratings of users change with time. The average rating of user $u$ at time $t$ is denoted by $\bar{r}_{ut}$. In Netflix data, distribution of $\bar{r}_{ut}$ is shown in Figure 3. There are 400K users in Netflix data and only three users who have rated more than 15K items are selected. The basic information of three users is listed in Table 1. Figure 3 shows user bias is shifting obviously with time. For example, user 305344 tends to give high score to items before 2003 but give low score to items after 2003.

In order to model this time effect, the user bias vector $b_u$ in Equation 3 is replaced by a time-dependent function $b_{u\tau}$:

$$b_{u\tau} = b_u + x_u^T z_\tau \tag{4}$$

where $\tau = \tau_{ui} = t_{ui} - t_u$ is the number of days after $u$ enters the recommender system and $x_u, z_\tau \in \mathbb{R}^f$ are two latent factor vectors for user $u$ and time $\tau$.

In this way, the RSVD model becomes:

$$\hat{r}_{ui} = \mu + b_u + b_i + b_t + p_u^T q_i + x_u^T z_\tau \tag{5}$$

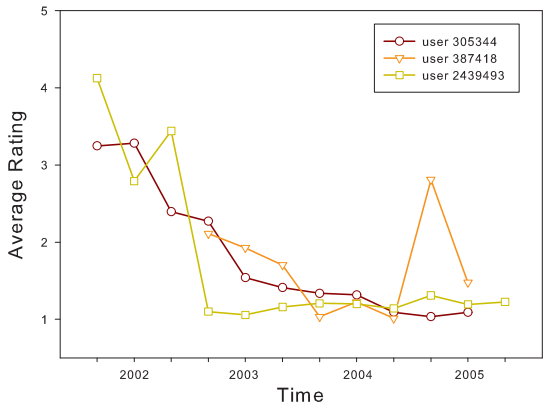


Figure 3. Distribution of $\bar{r}_{ut}$ in Netflix data.

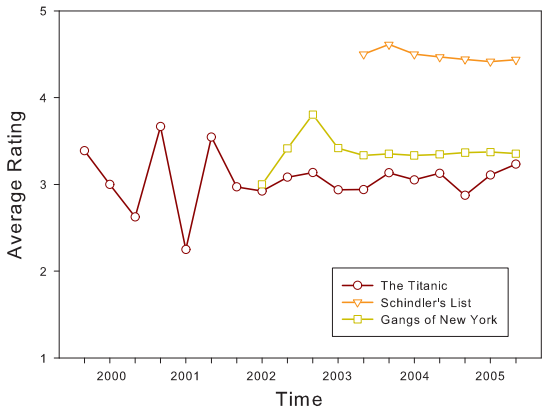


Figure 4. Distribution of $\bar{r}_{it}$ in Netflix data.

### 3.3. Item Biases Shifting

The popularity of items change with time. Several events can cause an item to become more or less favorable. For example, if an actor wins Oscar's best actor award, his previous movies may become more favorable. However, movies will loss popularity when time goes on generally.

The average rating of item $i$ at time $t$ is denoted by $\bar{r}_{it}$. In Netflix data, the distribution of $\bar{r}_{it}$ is shown in Figure 4. There are 17K movies in Netflix data and only three movies of them are selected, they are, The Titanic, Schindler's List, Gange of New York.

In Figure 4, there is an interesting phenomenon. For every movie, in the first few months, its average rating changes severely, but after a period of time, its average rating is nearly static. This means, users attitude to an item will be fixed after a period of time.

In this way, the item bias $b_i$ is also a function that changes with time. It is easier to capture the time effect in item bias by using a time-dependent item bias model:

$$b_{i\omega} = b_i + s_i^T y_\omega \tag{6}$$

where $\omega = \omega_{ui} = t_{ui} - t_i$ is the number of days after item $i$'s first rating was assigned. $s_i, y_\omega \in \mathbb{R}^f$ are two latent factor vectors for item $i$ and time $\omega$. Here, $s_i^T y_\omega$ represents the fluctuation of item $i$'s popularity with time $\omega$.

After adding item bias shifting model into Equation 5, the time-dependent model becomes:

$$\hat{r}_{ui} = \mu + b_u + b_i + b_t + p_u^T q_i + x_u^T z_\tau + s_i^T y_\omega$$

### 3.4. User Preference Shifting

Users change their preferences with time. For example, a boy likes cartoon when he is young and rates a cartoon "Toy Story", released in 1995, the highest score 5 stars in 1998. However, when he grows up, he does not like cartoon and rates another cartoon "The Incredibles", released in 2004, the low score 2 stars in 2005. Furthermore, users may change their attitude toward actors and directors. This time effect is called user preference shifting. In RSVD model, $p_u^T q_i$ represents the preference of user $u$ on item $i$. However, this model is time-independent. Therefore, in order to model user preference shifting, $p_u^T q_i$ is replaced by a time related model:

$$\text{preference}(u,i) = p_u^T q_i + \sum_{k=1}^{f} g_{u,k} \cdot l_{l,k} \cdot h_{\tau,k}$$

where $g_u, l_i, h_\tau \in \mathbb{R}^f$ are three latent factors for user $u$, item $i$ and time $\tau$. A similar model is used by Takács et al

**Algorithm 1**: TimeSVD model training algorithm

**Input**: observed rating set $\mathcal{K}$, latent factor number $f$, iteration times #Iter
**Output**: TimeSVD model parameters
**for** $count = 0; count < \#Iter; ++count$ **do**
  **foreach** $(u,i) \in \mathcal{K}$ **do**
    $e_{ui} = r_{ui} - \hat{r}_{ui}$;
    $t = t_{ui}$;
    $\tau = t_{ui} - t_u$;
    $\omega = t_{ui} - t_i$;
    $b_u = b_u + \eta \cdot (e_{ui} - \lambda b_u)$;
    $b_i = b_i + \eta \cdot (e_{ui} - \lambda b_i)$;
    $b_t = b_t + \eta \cdot (e_{ui} - \lambda b_t)$;
    **for** $k = 0; k < f; ++k$ **do**
      $p_{u,k} = p_{u,k} + \eta \cdot (e_{ui} \cdot q_{i,k} - \lambda p_{u,k})$;
      $q_{i,k} = q_{i,k} + \eta \cdot (e_{ui} \cdot p_{u,k} - \lambda q_{i,k})$;
      $x_{u,k} = x_{u,k} + \eta \cdot (e_{ui} \cdot z_{\tau,k} - \lambda x_{u,k})$;
      $z_{\tau,k} = z_{\tau,k} + \eta \cdot (e_{ui} \cdot x_{u,k} - \lambda z_{\tau,k})$;
      $s_{i,k} = s_{i,k} + \eta \cdot (e_{ui} \cdot y_{\omega,k} - \lambda s_{i,k})$;
      $y_{\omega,k} = y_{\omega,k} + \eta \cdot (e_{ui} \cdot s_{i,k} - \lambda y_{\omega,k})$;
      $p_{u,k} = g_{u,k} + \eta \cdot (e_{ui} \cdot l_{i,k} \cdot h_{\tau,k} - \lambda g_{u,k})$;
      $q_{i,k} = l_{i,k} + \eta \cdot (e_{ui} \cdot g_{u,k} \cdot h_{\tau,k} - \lambda l_{i,k})$;
      $h_{\tau,k} = h_{\tau,k} + \eta \cdot (e_{ui} \cdot g_{u,k} \cdot l_{i,k} - \lambda h_{\tau,k})$;
  $\eta = 0.9\eta$;

[23].

In this way, the time-dependent RSVD model becomes:

$$\hat{r}_{ui} = \mu + b_u + b_i + b_t + p_u^T q_i + x_u^T z_\tau + s_i^T y_\omega + \sum_k g_{u,k} \cdot l_{i,k} \cdot h_{\tau,k} \quad (7)$$

This is the finial model and is called TimeSVD.

TimeSVD has integrated four main time effects: time bias, user bias shifting, item bias shifting and user preference shifting. This model is trained by minimizing the following regularized cost function:

$$\sum_{(u,i)\in\mathcal{K}} \big\{(r_{ui} - \hat{r}_{ui})^2 + \lambda(b_u^2 + b_i^2 + b_t^2 + ||p_u||^2 + ||q_i||^2 + ||x_u||^2 + ||z_\tau||^2 + ||s_i||^2 + ||y_\omega||^2 + ||g_u||^2 + ||l_i||^2 + ||h_\tau||^2)\big\}$$

where $\lambda$ is regularization parameter which is chosen by cross-validation. In order to solve this optimization problem, we apply a simple gradient descent method which is often used in other studies [23], [12], [11]. Model parameters are estimated and updated by Algorithm 1. In this algorithm, all parameters are initialize randomly following Gaussian distribution $N(0, \sigma^2)$ and updated by gradient method.

## 4. Simple Time Effects

In this section, many other time effects are proposed. They are called simple time effects (STE) because they are used by simple models instead of factorized model.

### 4.1. Year, Month Effect

In previous discussion, three types of time metric are used for a user-item pair $(u, i)$:

- absolute time $t_{ui}$ : number of days since Nov 11, 1999 (This is the date when first rating is assigned in Netflix data).
- user time $\tau_{ui}$ : $\tau_{ui} = t_{ui} - t_u$ is number of days since user $u$'s first rating. This time metric is used when modeling user behavior shifting.
- item time $\omega_{ui}$ : $\omega_{ui} = t_{ui} - t_i$ is number of days since item $i$'s first rating. This time metric is used when modeling items' popularity shifting.

However, there are many other time metrics, such as year, month, week and so on. All of these time metrics can cause different time effects. For example, in China, some types of movies, such as new year's film, is very popular in January and February because of Chinese spring festival. Year 2005 is 60th anniversary of World War II victory and movies about world war II are popular in that year. Moreover, user bias also changes with these time metric. For example, users enjoy watching different types of movies in weekday and weekend. There are many examples about such time effects and all of these effects are used by a simple method.

We take item-month effect for example to analyze how to use this effect. Given a user-item pair $(u, i)$, $t_m(u, i) \in [1, 12]$ is the month when this rating is assigned. Then, the average prediction error $ave_{i,t_m}$ of item $i$ at month $t_m$ is defined by:

$$ave_{i,t_m} = \frac{\sum_{(u,i)\in\mathcal{K}, t_m(u,i)=t} e_{ui}}{n_{i,t_m}}$$

where $e_{ui} = r_{ui} - \hat{r}_{ui}$ is residual of pervious predictors and $n_{i,t_m}$ is number of ratings of item $i$ in month $t_m$. In order to avoid over-fitting, $ave_{i,t_m}$ is shrank toward zero by:

$$ave_{i,t_m} = \frac{ave_{i,t_m} \cdot n_{i,t_m}}{n_{i,t_m} + \alpha}$$

where $\alpha$ is shrinking parameter which is chosen by cross validation. This shrinking method is proposed by Koren et al to use global effects. Detailed discussion of shrinking can be found in [2]. Then, predictor $\hat{r}$ is updated by:

$$\hat{r}_{ui} \leftarrow \hat{r}_{ui} + ave_{i,t_m(u,i)}$$

Three other time effects similar to item-month effect are used in this work. They are :

- item-year effect: This effect represents the item-bias in different years.
- user-month effect: A user has different rating habit in different month. Season, festival, and many other reasons may cause a user changes his/her rating bias.
- user-year effect: This effect represents the user-bias in different years.

### 4.2. Loyalty, Activity and Popularity Effect

Beside above four time effects, another three time related effects are used in time-dependent model:

- user loyalty: a user $u$'s loyalty $\tau_u = \max\{t_{ui} - t_u\}$ measures how long this user is active in the recommender system. This metric can distinguish old users from fresh users. Old users and fresh users play different roles in recommender system and their rating behavior is also different. Old users have fixed rating habit and their preferences are easy to capture while fresh users only rate few movies and their preference is hard to predict. The average user loyalty of Netflix data is 36 days and distribution of user loyalty in Netflix data is shown in Figure 5. Figure 5 shows old users are very few and most of users' loyalty is less than a month.
- user activity: this metric measures the activity of users in a recommender system. Given a user $u$, his/her activity is defined by $\frac{R(u)}{\tau_u}$, where $R(u)$ is the number of ratings $u$ has assigned. It's obvious to see, this metric measures users activity by their average rating number every day.

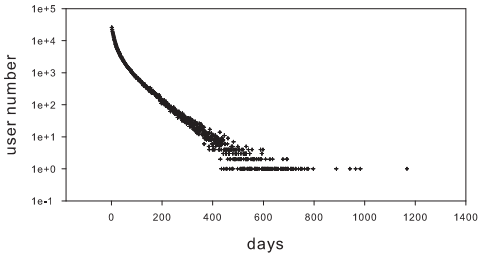


Figure 5. Distribution of user loyalty in Netflix data

- item popularity: If an item is rated by many people, it is popular. However, this metric is not accurate because old items will receive more ratings than new items. Therefore, the popularity of an item $i$ is defined as $\frac{R(i)}{\omega_i}$, that is the average rated times of item $i$ every day.

We take user activity effect as example to show how to use three effects above. Firstly, users are sorted by their activity. Secondly, they are divided into 100 bins where every bin contains nearly the same number of users. Then, we calculate average prediction error of every bin on the residual of previous predictions. For example, given a bin $bin_k$, $ave_{i,k}$ is calculated by:

$$ave_{i,k} = \frac{\sum_{(u,i)\in\mathcal{K}, u\in bin_k}(r_{ui} - \hat{r}_{ui})}{n_{i,k}}$$

where $n_{i,k} = |\{(u,j) \in \mathcal{K} : j = i, u \in bin_k\}|$. In order to avoid over-fitting, $ave_{i,k}$ is shrank toward zero by:

$$ave_{i,k} = ave_{i,k}\frac{n_{i,k}}{n_{i,k} + \alpha}$$

where $\alpha$ is shrinking parameter which can be estimated by cross-validation. At last, predictor is updated by:

$$\hat{r}_{ui} \leftarrow \hat{r}_{ui} + ave_{i,k}$$

where $k$ is the bin user $u$ belongs to.

In the experiment, TimeSVD is applied firstly, and then, at every step, one "effect" is estimated on the residuals of previous predictions.

## 5. Experiments

In this section, TimeSVD model is tested firstly and STE are applied on the residuals of TimeSVD predictor. Then time-dependent models are also tested on Quiz set. Finally, we test TimeSVD model on another famous data set MovieLens in order to show its effectiveness on different data set.

### 5.1. Result of TimeSVD Model

Firstly, experiments are done to compare TimeSVD model and RSVD model. Results are listed in Table 2. Table 2 shows that prediction accuracy of both RSVD and TimeSVD are improved when latent factors increased and TimeSVD can produce more accurate predictions than RSVD. The improvement of TimeSVD is also increased when lantent factors increased. TimeSVD can reduce the RMSE of RSVD by 0.0036 when $f = 20$ while it can reduce the RMSE of RSVD by 0.0055 when $f = 100$. TimeSVD is implemented by algorithm described in Algorithm 1 with learning rate $\eta = 0.007$ and regularization parameter $\lambda = 0.0127$. The algorithm will converge after 30 iterations.

| latent factors $f$ | TimeSVD | RSVD | Improvement |
|---|---|---|---|
| 20 | 0.9092 | 0.9128 | 0.0036 |
| 60 | 0.9028 | 0.9073 | 0.0045 |
| 100 | 0.9013 | 0.9068 | 0.0055 |

Table 2. Results of TimeSVD model and RSVD model RMSE on Probe with different latent factors number

| | I | II | III |
|---|---|---|---|
| RSVD | 0.9360 | 0.9341 | 0.8930 |
| TimeSVD | 0.9322 | 0.9277 | 0.8892 |
| Imporvement | 0.0038 | 0.0064 | 0.0038 |

Table 3. Results of TimeSVD model and RSVD model on Probe for users with different loyalty

TimeSVD can provide better predictions than RSVD. However, what type of users is benefit from TimeSVD? Many people think TimeSVD can produce more accurate predictions for old users (with high loyalty) than fresh users (with low loyalty). An experiment is done to verify this conclusion. Users are divided into three groups by their loyalty. The first group (I) contains users whose loyalty is less than 30 days, the second group (II) contains users whose loyalty is more than 30 days but less than 180 days, and the third group (III) contains users whose loyalty is more than 180 days. Then we apply RSVD and TimeSVD on Netflix, and calculate RMSE of three groups of users on Probe. The result is shown in Table 3. Table 3 shows, TimeSVD produce more accurate predictions for users in group II than users in group I,III. This because, users in group III have rated a lot of movies that both RSVD and TimeSVD can also make good predictions for them, users in group I have rated few movies that both RSVD and TimeSVD can not make good predictions for them. Only users in group II have rated many movies for a period of time that TimeSVD can make better prediction for them than RSVD.

### 5.2. Result of Simple Time Effects

After applying TimeSVD, we use simple time effects (STE) on the residual of TimeSVD result. At every step, only one effect is used. The results are listed in Table 4. The order

| Method (60 factors) | RMSE on Quiz |
|---|---|
| RSVD | 0.9090 |
| TimeSVD | 0.9045 |
| TimeSVD + STE | 0.9027 |

Table 5. Results of TimeSVD model and RSVD model on Quiz

of the time effects may influences the finial result, so if one applies them in a different order, the result will be different at all. However, we only test the ordering in Table 4. Results show every effect can improve the prediction accuracy of previous predictor.

### 5.3. Result on Quiz

Furthermore, some of algorithms proposed in this paper is tested on Quiz set which is a test set used by Netflix Prize to evaluate the performance of predictors. Three models, RSVD, TimeSVD, TimeSVD+STE, are used to predict ratings in Quiz and results are listed in Table 5. Results show, RSVD get RMSE=0.909 on Quiz while time-dependent models get RMSE=0.9027 on Quiz, which reduce the RMSE of RSVD by 0.0063. This is a significant improvement which indicates the importance of time information.

### 5.4. Result of Blending Method

In Netflix Prize, it is difficult to get very low RMSE by a single models. Most of researchers blend different methods together to get a good result. There are many blending methods, such as linear regression [16], NNblend [25]. We use a factorized linear regression blending method which get RMSE=0.8754 on Quiz by blending 35 models together. Among 35 models, 8 of them are time-dependent models (some of them are TimeSVD with different latent factor numbers, and others are models that only consider one of the four time-effects proposed in Section 3). In order to see how time information influences recommendation quality, these 8 predictors are removed form predictor set and the blending model of the rest 27 predictors get RMSE=0.8788 on Quiz. This result also indicates the importance of time effects and temporal features.

### 5.5. Result of TimeSVD on MovieLens data

Previous discussions about time effects in CF are based on Netflix data. However, MovieLens data is also used in our experiments. MovieLens is a web-based research recommender system that debuted in Fall 1997. MovieLens data set contains 1 million ratings for 3900 movies by 6040 users on scale 1 to 5. This data set is denser than Netflix data. The sparsity level of a data set is defined as [21] $1 - \frac{|\mathcal{K}|}{|\mathcal{U}||\mathcal{I}|}$ Then, the sparsity level of MovieLens data is 0.9575 while the sparsity level of Netflix data is 0.9882. In order to test time-dependent models, we divide MovieLens data into two part and use 90 percent of data as training data and 10 percent of data as test data.

| latent factors $f$ | TimeSVD | RSVD | Improvement |
|---|---|---|---|
| 20 | 0.8407 | 0.8451 | 0.0044 |
| 40 | 0.8377 | 0.8420 | 0.0043 |
| 60 | 0.8364 | 0.8405 | 0.0041 |

Table 6. Results of TimeSVD model and RSVD model RMSE on MovieLens data

Then, TimeSVD model is tested on MovieLens data. Results are listed in Table 6. In RSVD model, we choose learning rating $\eta = 0.02$ and regularization parameter $\lambda = 0.025$. In TimeSVD model, we choose learning rating $\eta = 0.02$ and regularization parameter $\lambda = 0.035$. Results of MovieLens data also indicate the effectiveness of TimeSVD model.

## 6. Conclusion and Future Work

In this work, we focus on how to use time information to make accurate predictions in CF based recommender system. Four main types of time effect (time bias, user bias shifting, item bias shifting, user preference shifting) are proposed and modeled by factorized model which is called TimeSVD. Experiment results show that TimeSVD can make more accurate predictions than RSVD, which is a time-independent model widely used in CF. Furthermore, many other time effects, such as year effect, month effect, user loyalty effect, user activity effect and item popularity effect are used in finial time-dependent model. All of these effects contribute to the finial improvement of prediction accuracy.

However, we do not consider neighborhood methods in this paper. Some researchers have proposed methods to use time information in item-based kNN method [6]. However, factorized model always perform better than kNN. In the future, we will focus on how to model time effect between items or users in factorized neighborhood methods.

## References


[1] Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Trans. on Knowl. and Data Eng.*, 17(6):734–749, 2005. Member-Adomavicius,, Gediminas and Member-Tuzhilin,, Alexander.

[2] R. M. Bell and Y. Koren. Scalable collaborative filtering with jointly derived neighborhood interpolation weights. pages 43–52, Washington, DC, USA, 2007. IEEE Computer Society.

| Method | $\alpha$ | 60 factors | | 100 factors | |
|---|---|---|---|---|---|
| | | RMSE on Probe | Improvement | RMSE on Probe | Improvement |
| RSVD | – | 0.90735 | – | 0.90685 | – |
| TimeSVD | – | 0.90289 | 0.00446 | 0.90127 | 0.00558 |
| Item Year Effect | 80 | 0.90216 | 0.00073 | 0.90051 | 0.00076 |
| Item Month Effect | 350 | 0.90174 | 0.00042 | 0.90007 | 0.00044 |
| User Year Effect | 50 | 0.90148 | 0.00026 | 0.89983 | 0.00024 |
| User Month Effect | 150 | 0.90142 | 0.00006 | 0.89979 | 0.00004 |
| User Loyalty | 700 | 0.90131 | 0.00011 | 0.89969 | 0.00010 |
| User Activity | 100 | 0.90112 | 0.00019 | 0.89952 | 0.00017 |
| Item Popularity | 500 | 0.90110 | 0.00002 | 0.89950 | 0.00002 |

Table 4. Results of Simple Time Effects on Probe


[3] J. Bennet and S. Lanning. The netflix prize. 2007.

[4] A. S. Das, M. Datar, A. Garg, and S. Rajaram. Google news personalization: scalable online collaborative filtering. *WWW '07: Proceedings of the 16th international conference on World Wide Web*, pages 271–280, 2007.

[5] M. Deshpande and G. Karypis. Item-based top-n recommendation algorithms. *ACM Trans. Inf. Syst.*, 22(1):143–177, 2004.

[6] Y. Ding and X. Li. Time weight collaborative filtering. In *CIKM '05: Proceedings of the 14th ACM international conference on Information and knowledge management*, pages 485–492, New York, NY, USA, 2005. ACM.

[7] T. Hofmann. Latent semantic models for collaborative filtering. *ACM Trans. Inf. Syst.*, 22(1):89–115, 2004.

[8] T. Hofmann and J. Puzicha. Latent class models for collaborative filtering. pages 688–693, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc.

[9] D. G. Ken Goldberg, Theresa Roeder and C. Perkins. Eigentaste: A constant time collaborative filtering algorithm. *Information Retrieval*.

[10] J. A. Konstan, B. N. Miller, D. Maltz, J. L. Herlocker, L. R. Gordon, and J. Riedl. Grouplens: applying collaborative filtering to usenet news. *Commun. ACM*, 40(3):77–87, 1997.

[11] Y. Koren. Factor in the neighbors : Scalable and accurate collaborative filtering.

[12] Y. Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. pages 426–434, New York, NY, USA, 2008. ACM.

[13] G. Linden, B. Smith, and J. York. Amazon.com recommendations: item-to-item collaborative filtering. *Internet Computing, IEEE*, 7(1):76–80, Jan/Feb 2003.

[14] B. Mehta, T. Hofmann, and W. Nejdl. Robust collaborative filtering. pages 49–56, New York, NY, USA, 2007. ACM.

[15] S.-T. Park and D. M. Pennock. Applying collaborative filtering techniques to movie search for better ranking and browsing. pages 550–559, New York, NY, USA, 2007. ACM.

[16] A. Paterek. Improving regularized singular value decomposition for collaborative filtering. pages 39–42, 2007.

[17] J. D. M. Rennie and N. Srebro. Fast maximum margin matrix factorization for collaborative prediction. pages 713–719, New York, NY, USA, 2005. ACM.

[18] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl. Grouplens: an open architecture for collaborative filtering of netnews. pages 175–186, New York, NY, USA, 1994. ACM.

[19] C. V. R.M. Bell, Y. Koren. The bellkor solution to the netflix prize. 2007.

[20] C. V. R.M. Bell, Y. Koren. The bellkor 2008 solution to the netflix prize. 2008.

[21] B. Sarwar, G. Karypis, J. Konstan, and J. Reidl. Item-based collaborative filtering recommendation algorithms. pages 285–295, New York, NY, USA, 2001. ACM.

[22] N. Srebro and T. Jaakkola. Weighted low rank approximation. *20th International Conference on Machine Learning*, 2003.

[23] G. Takács, I. Pilászy, B. Németh, and D. Tikk. On the gravity recommendation system.

[24] G. Takács, I. Pilászy, B. Németh, and D. Tikk. Major components of the gravity recommendation system. *SIGKDD Explor. Newsl.*, 9(2):80–83, 2007.

[25] A. Töscher and M. Jahrer. The bigchaos solution to the netflix prize 2008. 2008.