IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received January 6, 2017, accepted February 3, 2017, date of publication February 24, 2017, date of current version March 13, 2017.

Digital Object Identifier 10.1109/ACCESS.2017.2669243

Dynamic Model Adaptive to User Interest Drift Based on Cluster and Nearest Neighbors

BAOSHAN SUN AND LINGYU DONG
School of Computer Science and Software Engineering, Tianjin Polytechnic University, Tianjin 300387, China
Corresponding author: B. Sun (sunbaoshan@tipu.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61173032 and in part by the Science and Technology Commissioner Project of Tianjin under Grant 15JCTPJC58100.

ABSTRACT A recommendation system provides personalized recommendations on products and services to users. In the traditional recommendation system, the user interest is regarded as constant over time, while in fact, the user interest changes over time. Hence, tracking the user interest drift becomes key in designing the dynamic recommendation system. However, it is a challenge to find an accurate and effective method that can predict the user interest drift. To solve the prediction problem of the user interest drift, this paper adopts clustering and time impact factor matrix to monitor the degree of user interest drift in the class and more accurately predict an item's rating. We add a time impact factor to the original baseline estimates and use the linear regression to predict the user interest drift. Our comparative experiments are conducted on three big data sets: MovieLens100K, MovieLens1M, and MovieLens10M. The experimental results show that our proposed approach can efficiently improve the prediction accuracy.

INDEX TERMS Collaborative filtering, recommender systems, concept drift, time weight, MovieLens.

I. INTRODUCE

With the development of information technology and Internet, people gradually entered an information-explosion era from the past information-scarce one. Everyday, people will get a lot of information from different ways, but most are useless noise and valuable information is mixed in the information noise. In order to improve the efficiency, people need to filter out useless information by some filtering techniques. When user has clear demands, classified directory and search engine can be a good way to solve the problem of information overload [1]. However, most user demands are not clear in real life, recommendation system came into being. Recommendation system uses user modeling and user historical data to predict the user's favorite information. This method speeds up transmission efficiency of useful information and highlights the individuation, comparing with friend recommendation. At the same time, recommendation system is advantageous to the information producer to carry on the user population localization. Recommendation system is often divided into several categories:

Content-based [2]: the user will be recommended the items that are similar to his favorite items in the past.

Collaborative filtering [3]: the user will be recommended the items that are liked by people who have the same interests and hobbies with the user.

Hybrid: it combines two recommendation models to make recommendations.

Collaborative filtering recommendation system has received more attention in the past ten years, because it doesn't require too much professional knowledge and has the ability to discover models which are more complex and difficult to be discovered. Collaborative filtering system has two classical models: nearest neighbor model and latent factor model.

Nearest neighbor model uses neighbor relationships between people or between items. In user-based neighbor model, the user's nearest neighbors are used to predict the item's rating that the user will give. In item-based neighbor model, the user's rating for the item is predicted by the item's nearest neighbor ratings. Nearest neighbor model pays more attention to the relationships between people or between items, instead of the relationships between people and items, thereby it reduces a lot of complex calculations. But nearest neighbor model has its own shortcoming: it doesn't carefully analyze the users' rating, so that this model is good at recommending item and not good at predicting rating. Because use nearest neighbors are generated by clustering. If user u doesn't rate item i, most neighbors of user u don't rate item i. We can't get more accurate rating prediction by nearest neighbor method.

After the Netflix Prize, latent factor model has been rapidly developed. Compared with nearest neighbor model, latent factor model pays more attention to the implied semantics of the data set. The most common one is the matrix decomposition model: Rating matrix is decomposed into users - features and items - features two matrices. It can predict the ratings by using the matrix multiplication. Latent factor model try to use user features and item features to explain the reason of user ratings. Latent factor model is an effective prediction model that can be applied to the most recommender systems. However, the Latent factor model is difficult to produce detailed recommendation reasons, because user features and item features are difficult to express. Initial latent factor methods such as SVD, SVD++ [4] have a fatal flaw that it can't capture the user interest drift. These methods simply consider the ratings without considering the time. Therefore, we call these methods are static methods.

The user's preference for item is always changing with time or place, which leads to frequently redefine the user's interest. Dynamic recommendation system become a new trend of the present recommendation system design. And the concept of user interest drift [5] becomes important. Many scientists have begun to study the dynamic recommendation system. For example, Keoren [6] created capturing time drift model on the basis of the original papers to improving accuracy of recommenders in 2009. He considers user interest drift in the aspects of user, item, nearest neighbor and matrix decomposition, which makes the model quite complicated. And Koren believes items that are rated by same user are related and these relationships can be calculated in data set, but it's not the truth. Wei *et al.* [7] proposed a time-aware collaborative filtering framework for making recommendations based on user feedback data collected over time. Several effective time weighting techniques are applied to the prediction algorithm. But time weights only affect nearest neighbor model and its time decay model is too simple. In 2014, Lin and Liu [8] finished the user model prediction with an item clustering method. But they use a single time function and they use an inappropriate method to prevent over-fitting in their algorithm.

Motivated by above problems, we fully consider interest drift characteristic and utilize clustering and time decay model to overcome the problems. In this paper, we propose a model that uses the time impact factor matrix to predict user's rating, we call it dynamic time drift model (DTDM).

The contributions of this paper are summarized as follows.

- We apply the short-term, long-term, and periodic effects to the time impact factor matrix to improve the prediction accuracy.
- We present a novel idea that uses time distribution of dataset to calculate the time decay function.
- We improved the existing clustering algorithms, improved clustering algorithm makes the element numbers relatively average in each class, in order to achieve better prediction for user interest drift.
- We propose a new method to prevent over-fitting to replace the least-square method in our algorithm.
- We experimentally show that our proposed algorithm signifcantly improve prediction effect, compared with some mainstream recommendation algorithms.

The rest of this paper is organized as follows. In section II, we introduce some notations and basic methods. In section III, we introduce the overall structure of the DTDM algorithm. In section IV, we show the formula of core algorithm. In section V, through the contrast test results, we carry out some detailed analyses and put forward some new ideas. In section VI, we adjust the parameters and the results of our algorithm are compared with some mainstream recommendation algorithms. In section VII, we describe the conclusion and future work.

II. PRELIMINARIES

A. MEANING OF SYMBOL

Our recommendation system uses movie rating data set, we assume that users will use 1-5 ratings to represent their liking for items. Our work is to establish a model that can accurately predict users' ratings on the test set.

Movies dataset suppose m users and n items. The size of the rating matrix is $m * n$. We use special symbols to distinguish between users, items and time: for users u, v, for items i, j, and for time t. Rating $r_{u,i}(t)$ indicates the user u for item i rating at time t, the higher the rating, the more like. We use the set $S = \{(u, i, t)|r_{u,i}\ is\ known\}$ to represent the information that is known. The time stamp t is used to calculate the time impact factor.

B. BASELINE ESTIMATES

In the previous collaborative filtering system research, a large number of users and items data have some characteristics. Some people will give higher ratings than others, and some of the items will get higher ratings than the similar items. Keoren team [9] calculate these impacts with a linear method, and experiments show that this method is effective. As shown in Eq.(1), μ indicates the average rating of the whole data set, $b_{u,i}$ indicates a linear estimate for unknown ratings.

$$b_{u,i} = \mu + b_u + b_i \tag{1}$$

The parameters b_u and b_i are expressed separately the observed deviation of user u and item i. In order to facilitate the calculation, Keoren uses a simple method to calculate b_u and b_i.

$$b_i = \frac{\sum_{u:(u,i)\in S} (r_{u,i} - \mu)}{\beta_1 + |\{u|(u, i) \in S\}|} \tag{2}$$

$$b_u = \frac{\sum_{i:(u,i)\in S} (r_{u,i} - \mu - b_i)}{\beta_2 + |\{i|(u, i) \in S\}|} \tag{3}$$

β_1 and β_2 are smoothing factors, these parameter are required to adjust to achieve the best results. We will use $b_{u,i}$ to fill sparse matrix.

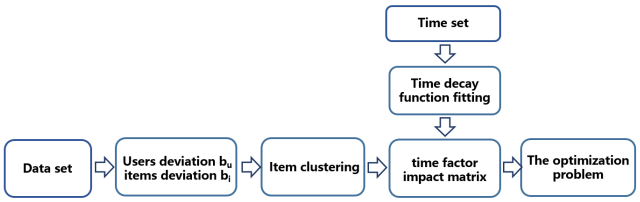


FIGURE 1. Algorithm framework of DTDM.

C. SOLVING METHOD OF THE PSEUDO INVERSE MATRIX

In the main algorithm, we use the method of solving the analytical solution. E.g.

$$R = T \cdot X \tag{4}$$

Rating matrix R and time impact factor matrix T are known, we need to solve the time factor coefficient matrix X. Time impact factor matrix T is obtained by time set and rating set. The calculation of T is described in section 4.1.

When T is a square matrix, we can use the adjoint matrix to find the inverse matrix.

$$A^{-1} = \frac{A^*}{|A|} \tag{5}$$

Then using Eq.(6), X can be obtained

$$T^{-1} \cdot R = T^{-1} \cdot T \cdot X = X \tag{6}$$

When T and the R matrix are non-square matrix, we can't simply use the adjoint matrix to solve the inverse matrix and need to be based on the T matrix is the row full rank matrix or the column full rank matrix to solve the pseudo inverse matrix.

Suppose A is full rank matrix of $m * n$, A^+ is the pseudo-inverse matrix:

$$if\ m < n : A^+ = A^T(AA^T)^{-1}$$
$$if\ m > n : A^+ = (AA^T)^{-1}A^T$$

III. ALGORITHM FRAMEWORK OF DTDM

In this section we describe algorithm framework, shown in Fig.1. After reading the data set, the value of b_u, b_i is calculated from the known rating set S, Eq.(2) and Eq.(3). Then the algorithm will complete the work of the item clustering. At the same time, we use time set to fit time decay function. Then algorithm establishes the time impact factor matrix T based on the item clustering result and time decay function. We obtain the time factor coefficient matrix X by analytic solution. In the optimization process, the algorithm searches for user nearest neighbors and uses its time factor coefficient matrix X to prevent over-fitting. We will discuss details in the following steps.

A. ITEM CLUSTERING

Because of the movies set characteristics, we can hardly find the same user repeated ratings for a movie. Without multiple ratings of the movie, we can't judge whether the user still interested in this movie. So we design another method to replace the observation of a movie that rated many times, this method use the item clustering: similar items will be in the same class, using item neighbor ratings to indirectly predict user interest drift degree.

Algorithm 1 K-Mean Improved Algorithm

Require: n vectors $r_1, r_2, \cdots, r_n$ representing n items, class number k_c

Ensure: k_c categories of items

1: Randomly initialize $3 * k_c$ class centers
2: Repeat:
3: Allocate each item to the nearest center
4: For each class, if having no item, then randomly pick one item from the largest class to it
5: Recalculate the class centers
6: Until converged
7: Repeat:
8: Incorporating A class that has the least amount of elements to B class that is nearest to A
9: Until the class number reduces to k_c

This prediction method will have a certain error. In order to reduce the error, two problems should be overcome in the item clustering: 1. In the results of the item clustering, elements are not very similar in each class, so that the core part of our algorithm can't have its due effect. 2. Elements in some classes are so scarce that it is difficult for us to find out the user interest drift .

The main reason for the first problem is the sparsity of the rating matrix. We can use the MovieLens 100k data set as an example. This data set has 100000 data, 943 users and 1682 movies. On average, each user comments on 106 movies, it accounts for 6.3% of total movies. And each movie is commented by 59 users, it accounts for 6.25% of total users. In other words, about 93.7% of the matrix is free. If the rating matrix is directly used for clustering, many similar movies may be not in the same class, our algorithm may partly failure. In order to solve this problem, we intend to use the filling method, as follows:

$$\underset{(u,i)\notin S}{r'_{u,i}} = \mu + b_u + b_i \tag{7}$$

$r'_{u,i}$ indicates linear prediction rating to fill empty rating, meaning and calculation of μ, b_u, b_i have been mentioned in Eq.(1), $S = \{(u, i, t)|r_{u,i}\ is\ known\}$. We have found that this formula can achieve better result in comparative experiments. In section V, all expressions of filling method were compared and analyzed.

The second problem is that there are too few elements in a class to predict well the user interest drift. We use a K-means improved algorithm shown in Algorithm 1 to solve the problem.

This algorithm is actually a combination of K-means algorithm and agglomerative hierarchical Algorithm. We first

generate $3 * k_c$ classes, every time we set the class containing the fewest elements incorporated into its nearest class, until class number condensed into k_c. The algorithm can effectively guarantee that the element number in each class is relative average.

B. ESTABLISHING THE TIME IMPACT FACTOR MATRIX

This part is the core of our algorithm, we will explain it in section 4.1 and 4.2, but considering the continuity of the depiction, we briefly describe here.

From Eq.(1), rating can be divided into overall average μ, users deviation b_u and items deviation b_i. But it doesn't completely fit the user rating. They have great difference, we assume that the difference comes from the time factor. So we define:

$$R_{u,i} = r_{u,i} - \mu - b_u - b_i \tag{8}$$

$R_{u,i}$ indicates the rating that is affected by the time factor. This part of the rating can be positive or negative, positive value indicates that user u loves item i more than the linear prediction, negative value indicates that user u doesn't love item i more than the linear prediction. In some ways we will use the method of matrix decomposition for function fitting, we take SVD as an example:

$$R_{u,i} = p_u \cdot q_i \tag{9}$$

p_u is a f dimensional feature vector related to the user u, q_i is a f dimensional feature vector related to the item i. f is an artificial parameter that needs to be adjusted.

But this model has a fatal flaw, the significance of p_u and q_i can not be fully explained, and people can't change these parameters properly based on experience to reduce the over-fitting. For example, we make an assumption that the value of Nth dimension is 3 in the p_u, but you can't figure out what it means, and you can't change this value artificially.

In this paper, we presents a time impact factor matrix that can be adjusted. The advantage of the matrix is that we can further reduce the prediction error by manual adjustment, it is different from the SVD method that the vector parameters can not be explained and can not be adjusted.

We have generated a non-square matrix $T_{u,t}$ composed of time impact factors. So we use analytical solution instead of matrix decomposition to solve the problem. Note the method of solving $T^+_{u,t}$,we have talked about in 2.4.

$$R_{u,i}(t) = T_{u,t} \cdot X_{u,t} \tag{10}$$

$$T^+_{u,t} \cdot R_{u,i}(t) = T^+_{u,t} \cdot T_{u,t} \cdot X_{u,t} = X_{u,t} \tag{11}$$

C. USER NEAREST NEIGHBOR SELECTION

First, we explain why to select the user neighborhood. After solving the time factor coefficient matrix $X_{u,t}$ in using Eq.(11). The process of obtaining analytical solution did not join the regularization constraints that may lead to over-fitting. So we need to use the nearest neighbors' time factor coefficient matrix and the original matrix to prevent over-fitting and make the prediction results more accurate.

In this part, we use the k-nearest neighbor algorithm, the general idea is to select a similarity calculation method, then it calculates similarity between a user and other users, and select the k_n highest similarity users as the user's nearest neighbor elements. k_n is an artificial value. There are many kinds of similarity calculation methods, in our test results, we found that the performance of the Pearson correlation coefficient is better. By using the Pearson correlation coefficient, we obtain user u nearest neighbor similarity set S_{nei}, and calculate the user u all nearest neighbors' similarity weight.

$$w_{uv} = \frac{s_{uv}}{\sum\limits_{v \in S_{nei}} s_{uv}} \tag{12}$$

s_{uv} indicates the similarity between u and v. w_{uv} indicates the weight of v in the user u nearest neighbor set. We will explain in detail the use of w_{uv} in 3.4.

D. SOLVING THE PROBLEM OF OPTIMAL SOLUTION

In solving the optimization problem, the international leading method is using the least square method. We take SVD as an example:

$$r_{u,i} = \mu + b_u + b_i + p_u \cdot q_i \tag{13}$$

p_u, q_i are vectors with f parameters. Every user have his own p_u, and every item have its own q_i, This algorithm eventually generate $(m + n) * f$ parameter space. To prevent over-fitting, the mainstream algorithms adopt the method of adding penalty term, as shown in Eq.(14)

$$\begin{aligned} min \sum_{(u,i \in S)} & (r_{u,i} - \mu - b_u - b_i - p_u \cdot q_i)^2 \\ & + \lambda_1(|p_u|^2 + |q_i|^2 + b_u^2 + b_i^2) \end{aligned} \tag{14}$$

λ_1 is a regularization parameter. After adding a penalty term, the algorithm not only is required to high fitting for the rating, but also has some restrictions on p_u, q_i, b_u, b_i.

In the least square method, algorithm requires multiple iterations to reach an optimal solution and achieve the accurate fitting. Our algorithm use the analytical solution method, the advantage of analytic solution is that it doesn't need multiple iterations, algorithm can achieve a relatively stable value after one iteration. However, the analytical solution method also has its own problem that it can't use the Eq.(14) to prevent over-fitting. Therefore, we used an alternative approach to prevent over-fitting, this method use the user nearest neighbors' weighting mentioned in 3.3. This method is proved to be effective on preventing over-fitting in the experiment. Neighbor weighting formula is as follows.

$$\bar{X}_{u,t} = (1 - \alpha) \cdot X_{u,t} + \alpha \cdot \sum_{v \in S} w_{uv} X_{v,t} \tag{15}$$

$X_{u,t}$ indicates the time factor coefficient matrix of user u calculated by Eq.(11). $X_{v,t}$ indicates the time factor coefficient matrix of user v that is the neighbor of user u. w_{uv} indicates the weight of user v for user u that can be calculated by Eq.(12). Using Eq.(15), we can get an adjusted $\bar{X}_{u,t}$ matrix.

Adjusted $\bar{X}_{u,t}$ is closer to the real situation. It can prevent the over-fitting problem caused by the analytical solution, α is an adjustable parameter of 0-1.

IV. ALGORITHM CORE

In this section, we show the core part of our algorithm. In 3.2, we just do a simple introduction. In this section we will divide it into two parts to make complete explanation.

A. TIME IMPACT FACTOR CALCULATION FORMULA

In 3.2, we mentioned the time impact factor matrix, but did not refer to the form of this matrix and the calculation formula. First of all, we give a detailed explanation of the Eq.(10).

$$R_{u,i}(t) = T_{u,t} \cdot X_{u,t} \tag{10}$$

$R_{u,i}(t)$ indicates the rating that user u give the item i in time t. Note that the $R_{u,i}(t)$ is obtained according to the Eq.(8) and is the time-related rating.

$T_{u,t}$ is the time impact factor matrix, which indicates the preference of user u for each class at t time.

$X_{u,t}$ is the time factor coefficient matrix, which provides us a linear combination coefficient. This matrix is obtained by solving the analytic solution.

Here we explain generating method of $T_{u,t}$. We assume that there are three models affecting the user rating for the item. they are time interval, number interval and seasonal impact. For the three models, we come up with three formulas to calculate the degree of impact. Three time factors are calculated as follows respectively.

$$T1_{u,k,t} = \frac{\sum_{i \in k} R_{u,i}(a \cdot e^{-\lambda|t-t'|} + b)}{\sum_{i \in k}(a \cdot e^{-\lambda|t-t'|} + b)} \tag{16}$$

In Eq.(16), we calculate the impact of time interval for user u. $T1_{u,k,t}$ indicates the time interval impact factor of user u to class k in time t. t' indicates the time of rating item i. a, b and λ are adjusting parameters, these will be introduced in details in the section 4.2.

When there are a lot of items belonging to the same class k, the longer the time interval between t and t' is, the smaller the impact on the user's preferences will be. the shorter the time interval of t and t' is, the greater the impact on the user's preferences will be.

$$T2_{u,k,l} = \frac{\sum_{i \in k} R_{u,i}(a \cdot e^{-\lambda|l-l'|} + b)}{\sum_{i \in k}(a \cdot e^{-\lambda|l-l'|} + b)} \tag{17}$$

In Eq.(17), we calculate the impact of number interval for user u. $T2_{u,k,l}$ indicates the number interval impact factor of user u to class k. We sort items which the user u commented with the time sequence. According to the location of the item in the user's time sequence, algorithm arranges the corresponding position parameters l to each item.

In our framework, $T1_{u,k,t}$ and $T2_{u,k,l}$ are complementary to each other, then the algorithm can achieve a relatively stable prediction. $T1_{u,k,t}$ is more in favor of a long-term stable time factor, it is a function that change over time, and is not affected by the user rate the other items. so it has its own shortcomings: After user u watched a movie of k category,user u watched 5 movies of the $k+1$ category. But $T1_{u,k,t}$ does not have a mechanism for the number interval, the user interest maybe have changed, but the model does not detect the change in time. The $T2_{u,k,l}$ can solve this problem, it is more in favor of a short time factor, it also has its own problems, it can't be monitored in time interval. When $T1_{u,k,t}$ and $T2_{u,k,l}$ are combined, the combination of temporary interest drift and long term interest drift is completed. On this basis, we try to find the periodic trend of rating, so we made another time impact factor function $T3_{u,k,t}$.

$$T3_{u,k,t} = \frac{\sum_{i \in k} R_{u,i} \cdot cos(2\pi \cdot \frac{t-t'}{86400*365})}{\sum_{i \in k} cos(2\pi \cdot \frac{t-t'}{86400*365})} \tag{18}$$

In Eq.(18), we calculate the impact of seasonal factors for user u, $T3_{u,k,t}$ indicates the seasonal impact factor of user u to class k.

We assume that u users rated some items that belong to k class in time t, then on the same day next year, $T3_{u,k,t}$ will have higher value, but after six months from the time t, the $T3_{u,k,t}$ will have the lower negative weight. This formula guarantees that the function has the cyclical changes, so that it can predict some item categories that has periodic characteristics.

We can define the time impact factor matrix $T_{u,t}$ under the impact of three factors:

$$T_{u,t} = [T1_{u,1,t}, T2_{u,1,l}, T3_{u,1,t}, T1_{u,2,t}, T2_{u,2,l}, T3_{u,2,t}, \cdots, T1_{u,k,t}, T2_{u,k,l}, T3_{u,k,t}]$$

Through the time impact factor matrix that we give ,we can see that each class have their own $T1_{u,k,t}$, $T2_{u,k,l}$, $T3_{u,k,t}$, so each class will have their own $X1_{u,k,t}$, $X2_{u,k,l}$, $X3_{u,k,t}$. The following is the expression of the time factor coefficient matrix $X_{u,t}$:

$$X_{u,t} = [X1_{u,1,t}, X2_{u,1,l}, X3_{u,1,t}, X1_{u,2,t}, X2_{u,2,l}, X3_{u,2,t}, \cdots, X1_{u,k,t}, X2_{u,k,l}, X3_{u,k,t}]$$

Note that $T_{u,t}$, $X_{u,t}$ are for the user u. In other words, each user has its own $T_{u,t}$ and $X_{u,t}$ vector.

B. TIME DECAY FUNCTION FITTING

In 4.1, Eq.(16) and Eq.(17) mentioned three parameters a, b, λ, but we did not give their exact values, because these three parameters are automatically adjusted based on the data sets and algorithm. We propose a fitting algorithm for the time decay function. The main function is as follows.

$$f(t) = a \cdot e^{-\lambda t} + b \tag{19}$$

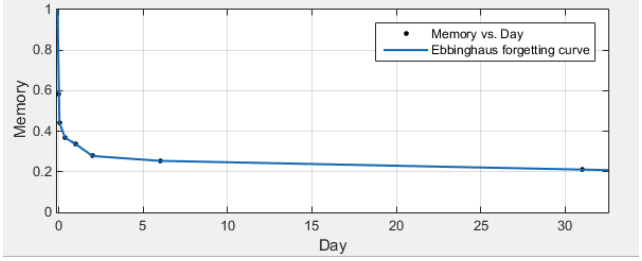


FIGURE 2. Ebbinghaus forgetting curve.

Algorithm 2 Time Decay Factor Algorithm

Require: Rating Set,Time Set

Ensure: a, b, λ

1: Computing time span of the data set
2: Establish items-time matrix for storing the comment number of each item in every day
3: Fill items - time matrix by the rating set and time set
4: Set a weight γ , $N = \gamma \cdot n$
5: Calculate $TOP - N$ items of everyday.
6: Calculate the similarity of T day and $T + t$ day, as shown in Eq.(20)
7: Calculate the average similarity of the same time span, as shown in Eq.(21)
8: Using the Exponential approximation fitting function

In order to obtain the time decay function, we made three reasonable assumptions:

1. User interest drift will have some relationships with users' time memory, we take the German psychologist Ebbinghaus's forgetting curve [10] as the core of the model. We research the change characteristics of time forgetting curve model, as shown in Fig. 2. We think that the user's interest in the movie is the biggest and the interest value is 1, when he has just commented on this movie. With the change of time, the weight begin to decreased significantly, and then tend to a steady value b.

2. People interest have different drift speed in different data sets, which leads to a result that different data sets have different timeliness. For example, People interest have different drift speed in the news data set and the movie data set.

3. Interest drift speed for data set can be well reflected in data set daily TOP-N. So we can use the similarity of TOP-N to estimate the regression function model in the algorithm, to achieve the purpose that algorithm can automatically generate function model.

Based on three assumptions, we made a relatively reasonable time decay factor algorithm which is given in Algorithm 2.

In the algorithm, we have given a parameter γ. The main purpose of this parameter is to control the value of N in the $TOP - N$. This value will be taken as a certain percentage of item number in the date set. No matter how large or small data set is, N will be a relatively reasonable value without major changes.

In algorithm step 6, we assume T can be any day in the time span and t indicates the number of time span between two specified days. We need to show our similarity calculation formula. We use $TOPN_T$ to represent the $TOP - N$ list of the T day.

$$S_{T,T+t} = \frac{|TOPN_T \cap TOPN_{T+t}|}{|TOPN_T \cup TOPN_{T+t}|} \tag{20}$$

In algorithm step 7, we calculate the average similarity of the same time span, as shown in Eq.(21). A simple understanding that time span of the first and second day is 1, time span of second and third days is 1, the algorithm calculates the average similarity of all time span 1, to make the algorithm result more reliable.

S_t indicates the average similarity of the t time span.

$$S_t = \frac{\sum_{T=1}^{maxT-t-1} S_{T,T+t}}{|S_{T,T+t}|} \tag{21}$$

At the end we get a series of data and a formula model, but don't have parameter. We use an exponential approximation to adjust the parameters.

V. ALGORITHM DETAIL ANALYSIS

In the previous part, we have introduced the whole algorithm, but the detail problems are not analyzed. In this section, we try to solve these problems through experiments, we put forward some solutions to every problem. Some of them are new ideas to solve these problems, here to show you.

A. RATING FILL METHOD

In 3.1, we have shown that we use Eq.(7) to fill matrix. Due to the filling work, the matrix becomes very dense and item clustering effect is improved, but if you use a wrong filling method, it doesn't improve the effectiveness of clustering. We believe that the four formulas below is very effective.

$$r_{\substack{u,i \\ (u,i)\notin S}} = \bar{r}_i \tag{22}$$

$$r_{\substack{u,i \\ (u,i)\notin S}} = \bar{r}_u \tag{23}$$

$\bar{r}_i$ indicates the average rating of item i, $\bar{r}_u$ indicates the average rating of user u. Mean can represent the users' general attitude to the items. In our experiments, it is found that these simple methods can obtain better results than the global mean filled or not filled. The $\bar{r}_u$ filling will get better effect than $\bar{r}_i$. Through our analysis, the $\bar{r}_u$ can reflects the user's attitude better than the $\bar{r}_i$ in the item clustering.

$$r_{\substack{u,i \\ (u,i)\notin S}} = \mu + b_u + b_i \tag{7}$$

Further, we think out the Eq.(7) to solve the filling matrix problem, we found that this filling method has a better clustering effect. After our analysis, we believe that the personalized rating filling is a good way to enhance the clustering effect. In Eq.(7), the rating filling already has a very high personalization, which will make the item clustering effect

more prominent. With this research direction, we found that this method is not the best personalized rating, because the b_u and b_i aren't the best way to fit. We have come up with a weighted method of nearest neighbors' rating:

$$\underset{(u,i)\notin S}{r_{u,i}} = \sum_{v\in U_{nei}} w_{uv} r_{v,i} \tag{24}$$

U_{nei} indicates the set of the user's nearest neighbors. In our experiments, we found that the Eq.(24) is not very effective. Through our research for the experimental data, we think the problem is that we didn't use the filling method in the process of user clustering, it lead to a bad result. If user u has not rated item i, many nearest neighbors of user u has not rated item i. That leads to inaccurate rating, so that the algorithm effect is not as good as the previous three algorithms.

Through this part of the research, we have a certain understanding to fill method, personalized prominent and stable fill method is easier to get a good clustering results.

B. ITEM CLUSTERING METHOD

In the item clustering, we mainly solve 2 problems: 1.Similarity algorithm of clustering. 2. The clustering algorithm.

1) SIMILARITY ALGORITHM OF CLUSTERING

First, we compare the similarity mainstream clustering algorithm, respectively: Euclidean distance, cosine similarity, Pearson correlation.

Euclidean formula as in Eq.(25)

$$E_{ij} = \sqrt{\sum_{u=1}^{n} (r_{u,i} - r_{u,j})^2} \tag{25}$$

Cosine similarity formula as in Eq.(26)

$$sim(i,j) = \frac{\sum_1^n r_{u,i} \cdot r_{u,j}}{\sqrt{\sum_1^n r_{u,i}^2} \cdot \sqrt{\sum_1^n r_{u,j}^2}} \tag{26}$$

Pearson correlation formula as in Eq.(27)

$$sim(i,j) = \frac{\sum_1^n (r_{u,i} - \bar{r}_i) \cdot (r_{u,j} - \bar{r}_j)}{\sqrt{\sum_1^n (r_{u,i} - \bar{r}_i)^2} \cdot \sqrt{\sum_1^n (r_{u,j} - \bar{r}_j)^2}} \tag{27}$$

Comparing these algorithms in the experiments, we found that Pearson correlation algorithm is better than the cosine algorithm and Euclidean distance. After our analysis, we think that the cosine algorithm does not take into account the relationship between the rating and the average rating, it only considers the angle problem instead of the real neighborhood. For example, i_1, i_2, i_3 three items respectively get the same three users' ratings, ratings vector is i_1 (4,4,4), i_2 (1,1,1), i_3 (4,4,5). It is obvious that i_2 and i_3 are more similar, but the cosine algorithm calculate that i_1 and i_2 are more similar. Because this method doesn't take into account the relationship between the rating and the average rating, which can show user attitude for the movie. Euclidean distance appears similar problem. Without considering the user attitude, the clustering is not accurate. Instead, we can find that Pearson coefficient take into account the problem of average rating. Following this direction, we find another algorithm: Adjusted Cosine Similarity. Like Pearson correlation, The range of Adjusted Cosine Similarity is -1 to 1. The difference between the adjusted Cosine Similarity and the Pearson correlation is the selection of the average rating. We have mentioned above, in fact, the users' attitude will be a great influence on clustering. Finally the experimental result of the adjusted Cosine Similarity is better than Pearson correlation.

Adjusted Cosine Similarity is given by Eq.(28)

$$sim(i,j) = \frac{\sum_1^n (r_{u,i} - \bar{r}_u) \cdot (r_{u,j} - \bar{r}_u)}{\sqrt{\sum_1^n (r_{u,i} - \bar{r}_u)^2} \cdot \sqrt{\sum_1^n (r_{u,j} - \bar{r}_u)^2}} \tag{28}$$

2) THE CLUSTERING ALGORITHM

In 3.1, we have mentioned that this algorithm is the combination of K-means algorithm and agglomerative hierarchical Algorithm. But in the popular aggregation algorithm, the algorithm will combine the two closest class. At the beginning we used this aggregate algorithm. In the process of the experiment we found that this aggregation algorithm is not very suitable for our algorithm, because what we need is relatively big data amounts in each class that can be used to observe user interest drift. If we use the popular aggregation algorithm, we can find that there are isolated point or small groups monopolizing a class, which is not conducive to the observation of interest drift. On this basis, we use the mode of merging minimal class and its nearest class to replace another mode of merging the nearest class. The advantage of this method is that the element numbers in each class are relatively average, it facilitate the observation of interest drift.

C. CALCULATION OF TIME IMPACT FACTOR

In 4.1, we put forward three calculation formulas about time factor, and give the time impact factor matrix. We think that the three time impact models are effective. But in the experiment, we found that is not the case. In the MovieLens data set, the effect of using $T1_{u,k,t}$, $T2_{u,k,l}$ is better than using $T1_{u,k,t}$, $T2_{u,k,l}$, $T3_{u,k,t}$. Based on the results of this test, the reasons are analyzed. We believe that the main reasons for the $T3_{u,k,t}$ failure are as follows:

1. The long time periodic characteristics of movie data set may be not exist , the majority of users will not change their own interests over season or month. Therefore, periodic change does not help predict very well.

2. The periodic change model $T3_{u,k,t}$ is too simple, or it is not suitable for the interest drift prediction after clustering.

D. TIME DECAY FUNCTION FITTING

In 4.2, we discuss a new method for obtaining time decay function. In the hypothesis, we think this is a good algorithm, it can regulate time decay function based on different databases, in order to better fit the data set. But the effect of the algorithm isn't good in the experiment. We found that the time decay function can't achieve better results compared with taking $a = 1, b = 0, \lambda = 1$ directly. Through the

analysis of the experimental results, we found that the following problems may affect the final experimental result:

(1) User selection: MovieLens data sets are cleaned data sets, users who rate less than 20 movies completely deleted in this data set. In this case, the statistics of $TOP-N$ have a certain error, this error can interfere the final fitting result.

(2) Item selection: MovieLens data sets not only clean users, but also screen movies with certain rules, this type of screening is artificial. We found that many movies already exist before the initial starting time of the data set. The data set has a certain degree of closure. There is a problem: after the movie exists a year or a few years, the number of evaluation will become stable, it will affect the results and cause error fitting.

(3) Data set: we choose relatively stable movie data sets, rather than news data set that has more real-time effect. In this stable data set, the algorithm is very sensitive to the weight γ of $TOP-N$, especially like MovieLens 100k that is small closed and stable data set. The adjustable parameter of γ became a problem remaining to be solved.

(4) Algorithm: in addition to the Eq.(20) and Eq.(21), we don't have a better method to calculate the similarity. This similarity calculation uses average. In the large data set, the average similarity will lose its due effect, because the similarity degree will be more closer to γ after many days.

(5) Model: we have used the Eq.(8) to remove the non-time impact factor. In the process of eliminating, We may have removed the part that is affected by the parameter b in Eq.(19), which led to the parameters b can't play its effect.

VI. PARAMETER ADJUSTMENT AND EXPERIMENTAL COMPARISON

In this section we will do the work of Adjustment parameter, and we will compare our algorithm with some mainstream algorithms.

A. EVALUATION METRICS AND DESCRIPTION OF TEST SET

To estimate the quality of each algorithm and compare these algorithms, we have used internationally accepted measurement methods: RMSE (Mean Squared Error Root) and MAE (Mean Absolute Error):

$$RMSE = \sqrt{\frac{\sum_{(u,i)\in S}(r_{u,i}-\hat{r}_{u,i})^2}{|S|}} \tag{29}$$

$$MAE = \frac{\sum_{(u,i)\in S}|r_{u,i}-\hat{r}_{u,i}|}{|S|} \tag{30}$$

Compared with MAE method, RMSE method increases the punishment of the large error. $r_{u,i}$ indicates the actual rating of the item. $\hat{r}_{u,i}$ indicates the predictive value of the item.

B. DATASET PREPARING

Our experiments are based on three well-known movie rating data sets, MovieLens 100k, MovieLens 1M and MovieLens 10M [11]. The three data sets were collected by the University of Minnesota as a research project. The advantage of using these data sets is that these data sets have been cleaned in advance, which can ensure the low noise of the data set. It helps to better observe the effectiveness of the algorithm. We describe the related data of these data sets in table 1.

TABLE 1. Characteristics of the datasets.

dataset	users	items	ratings	time span	sparsity
MovieLens 100k	943	1682	100000	214 days	93.7%
MovieLens 1M	6040	3900	1000209	1038 days	95.75%
MovieLens 10M	71567	10681	10000054	5109 days	98.69%

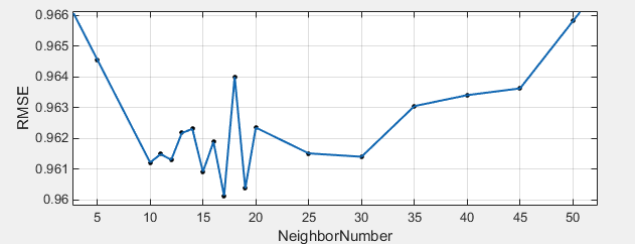


FIGURE 3. When clustering number k_c is 5, Neighbor number and RMSE.

In test set, we use five latest data of each users as the test set elements. So MovieLens 100k test set has 4715 elements, MovieLens 1M test set has 30200 elements, MovieLens 100k test set has 357835 elements. These data sets have considerable test sets, which can detect the effectiveness of the algorithm.

C. ADJUSTMENT OF PARAMETER

In the section V, we made the experiment method comparisons, and also give the reasons for the effect improvement. In the parameter adjustment stage, we can only adjust parameters based on experimental results: we first made adjustments in a large range, then we adjust further on these effective intervals. There are two important parameters to be adjusted. User neighbor number k_n of k nearest neighbor algorithm and the clustering number k_c of items clustering algorithm.

1) ADJUSTMENT WORK OF k_n

In the choice of neighbor number k_n, we first tested from 0 to 50 with interval of 5. When the neighbor number k_n is 0, we found the RMSE is 1.034084199 that is not a good prediction effect. This result also indirectly proves that we can use the nearest neighbor model to prevent over-fitting. We found a small value, when $k_n = 15$. We have done experiments for 10-20. The results are shown in Figure 3.

Finally we get the RMSE minimum 0.960140846, when $k_n = 17$.

2) ADJUSTMENT WORK OF k_c

In the choice of neighbor number k_c, we first tested from 1 to 15 with interval of 1.

We found that the algorithm will reach a relatively stable value in clustering number k_c is 3-7. The results are shown in Figure 4. After several tests, we found that RMSE will reach the minimum value 0.921690948, when the $k_c = 4$.

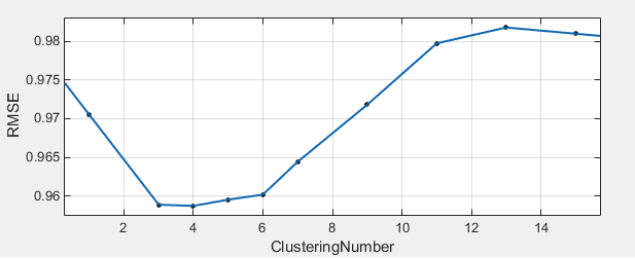


FIGURE 4. When Neighbor number k_n is 17,Clustering number and RMSE.

D. COMPARISON WITH MAINSTREAM ALGORITHMS

In this section we will compare the DTDM algorithm with some mainstream static and dynamic algorithms.

1) FUNK-SVD MODEL

The whole rating is decomposed into two matrices, which are user - feature and item - feature, and using matrix multiplication predict unknown rating. Its formula is as follows:

$$r_{u,i} = p_u \cdot q_i \tag{31}$$

2) BiasSVD MODEL

This is a model that is a combination of Linear estimate model and Funk-SVD model:

$$r_{u,i} = \mu + b_u + b_i + p_u \cdot q_i$$

linear estimate model contains the global average μ, user deviation b_u and item deviation b_i.

3) SVD++ MODEL

On the basis of BiasSVD model, koren added feedback of user's historical behavior.

$$r_{u,i} = \mu + b_u + b_i + q_i^T \cdot (p_u + \frac{1}{\sqrt{|N(u)|}} \sum_{j \in N(u)} y_j)$$

$N(u)$ indicates movie set of user ratings. y_j is a vector that is used to measure the relationship between i and j.

Note that the former three methods are static methods, which don't have any time factors. These methods do not obtain different predictions in different time. But the dynamic method will consider the time parameters, so we introduce to you the three mainstream dynamic methods.

4) ITEM TIME MODEL

This model is mainly to consider the user interest drift over time in the item level.

$$r_{u,i} = \mu + b_u + b_i + b_{i,bin(t)}$$

In this model, $b_{i,bin(t)}$ is divided into several stages. In different stages, the value is not the same, so that this model has some dynamic characteristics.

TABLE 2. Experimental results of RMSE.

	MovieLens 100K	MovieLens 1M	MovieLens 10M
Funk-SVD model	1.038179778	0.960586361	0.836428634
BiasSVD model	1.001614153	0.938440456	0.815669158
SVD++ model	1.001235966	0.923075148	0.806759582
Item time model	1.000958229	0.939434318	0.812781762
Time linear model	1.001084024	0.937824303	0.817791159
TimeSVD++ model	0.987954103	0.934014218	0.808740258
DTDM model	0.952012147	0.921690948	0.804584319

TABLE 3. Experimental results of MAE.

	MovieLens 100K	MovieLens 1M	MovieLens 10M
Funk-SVD model	0.823826287	0.762299854	0.671881501
BiasSVD model	0.796115725	0.742443631	0.656524374
SVD++ model	0.796071635	0.731028117	0.650106657
Item time model	0.795820452	0.742812954	0.653005455
Time linear model	0.795544169	0.741871912	0.65780201
TimeSVD++ model	0.780357582	0.73472479	0.646372117
DTDM model	0.751386825	0.723966162	0.64396973

5) TIME LINEAR MODEL

This model considers not only the user interest drift in the item but also the user interest drift in the user.

$$r_{u,i} = \mu + b_u + \alpha_u dev_u(t) + b_i + b_{i,bin(t)}$$

α_u is a adjustment parameter, $dev_u(t)$ calculation formula is as follows

$$dev_u(t) = sign(t - t_u) \cdot |t - t_u|^\beta$$

The timestamp t_u is the user u rating time, β is a adjustment parameter.

6) TimeSVD++ MODEL

This algorithm is a more complex dynamic model based on SVD++ model.

$$r_{u,i} = \mu + b_u(t) + b_i(t) + q_i^T \cdot (p_u(t) + \frac{1}{\sqrt{|N(u)|}} \sum_{j \in N(u)} y_j)$$

Through the formula below, we can learn the calculation of $b_u(t)$, $b_i(t)$ and $p_u(t)$.

$$\begin{aligned} b_u(t) &= b_u + \alpha_u dev_u(t) + b_{u,bin(t)} + b_{u,period(t)} \\ dev_u(t) &= sign(t - t_u) \cdot |t - t_u|^\beta \\ b_i(t) &= b_i + b_{i,bin(t)} + b_{i,period(t)} \\ p_u(t) &= p_u + \alpha_u dev_u(t) + p_{u,t} \end{aligned}$$

$b_{u,period(t)}$ and $b_{i,period(t)}$ indicate the periodic changes of users and items. p_u captures the stationary portion of the factor, $\alpha_u \cdot dev_u(t)$ approximates a possible portion that changes linearly over time, and $p_{u,t}$ absorbs the very local, day-specific variability.

In three data sets, we will compare our algorithm with the mainstream algorithms, the comparison of our results are show in table 2 and table 3.

After the experiment, we found that our algorithm is better than some mainstream algorithms in these data sets. But at the same time, we also found that SVD++ model achieve better results than mainstream dynamic algorithms in MovieLens 1M and MovieLens 10M. In our research, we found that some

users who rate less than 20 movies have not been deleted in MovieLens 1M and MovieLens 10M. All of the dynamic algorithms are dependent on the time variation. If the personal data set is very small, the dynamic algorithm may not show better results than the static algorithm.

VII. CONCLUSIONS

In this paper, we propose a new dynamic recommendation algorithm DTDM. This algorithm classifies all items and predicts unknown ratings on each class in order to better rating prediction results. we use the pseudo inverse matrix method to optimize DTDM algorithm and use the nearest neighbor method to prevent over-fitting.

In the experiment, we did some comparison of algorithm details. At last we adjust the parameters, and compare with some mainstream algorithms on three big data sets. The experimental results show that the DTDM algorithm has excellent performance.

In the next step, we need to find more effective cycle impact factors and decay function fitting algorithms, in order to enhance the performance of the algorithm again.

ACKNOWLEDGEMENTS

The authors would like to sincerely thank the reviewers for their insightful comments and very valuable suggestions.

REFERENCES

[1] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl, "GroupLens: An open architecture for collaborative filtering of netnews," in *Proc. ACM Conf. Comput. Supported Cooperat. Work*, 1994, pp. 175–186.

[2] Y. Zhang, J. Callan, and T. Minka, "Novelty and redundancy detection in adaptive filtering," in *Proc. 25th Annu. Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, 2002, pp. 81–88.

[3] D. Goldberg, D. Nichols, B. M. Oki, and D. Terry, "Using collaborative filtering to weave an information tapestry," *Commun. ACM*, vol. 35, no. 12, pp. 61–70, 1992.

[4] Y. Koren, "Factorization meets the neighborhood: A multifaceted collaborative filtering model," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Las Vegas, NV, USA, Aug. 2008, pp. 426–434.

[5] G. Widmer and M. Kubat, "Learning in the presence of concept drift and hidden contexts," *Mach. Learn.*, vol. 23, no. 1, pp. 69–101, 1996.

[6] Y. Koren, "Collaborative filtering with temporal dynamics," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2009, pp. 89–97.

[7] S. Wei, N. Ye, and Q. Zhang, "Time-aware collaborative filtering for recommender systems," in *Pattern Recognition*. Berlin, Germany: Springer, 2012.

[8] K. Lin and D. Liu, "Category-based dynamic recommendations adaptive to user interest drifts," in *Proc. 6th Int. Conf. Wireless Commun. Signal Process.*, 2014, pp. 1–6.

[9] Y. Koren, "Factor in the neighbors: Scalable and accurate collaborative filtering," *ACM Trans. Knowl. Discovery Data*, vol. 4, no. 1, 2010, Art. no. 1.

[10] H. Ebbinghaus, "Memory: A contribution to experimental psychology," *Ann. Neurosci.*, vol. 20, no. 4, pp. 155–156, 2013.

[11] F. M. Harper and J. A. Konstan, "The MovieLens datasets: History and context," *ACM Trans. Interact. Intell. Syst.*, vol. 5, no. 4, 2016, Art. no. 19.

[12] L. Banda and K. K. Bharadwaj, "Evaluation of collaborative filtering based on tagging with diffusion similarity using gradual decay approach," in *Advanced Computing, Networking and Informatics*, vol. 1. Basel, Switzerland: Springer International, 2014, pp. 421–428.

[13] J. Gaillard and J. M. Renders, "Time-sensitive collaborative filtering through adaptive matrix completion," in *Proc. Eur. Conf. Inf. Retr.*, Springer, 2015, pp. 327–332.

[14] N. Koenigstein, G. Dror, and Y. Koren, "Yahoo! Music recommendations: Modeling music ratings with temporal dynamics and item taxonomy," in *Proc. ACM Conf. Recommender Syst. (Recsys)*, Chicago, IL, USA, Oct. 2011, pp. 165–172.

[15] C. Luo, X. Cai, and N. Chowdhury, "Probabilistic temporal bilinear model for temporal dynamic recommender systems," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2015, pp. 1–8.

[16] W. Sen, Z. Xiaonan, D. Yannan, "A collaborative filtering recommender system integrated with interest drift based on forgetting function," *Int. J. u-and e-Serv. Sci. Technol.*, vol. 8, no. 4, pp. 247–264, 2015.

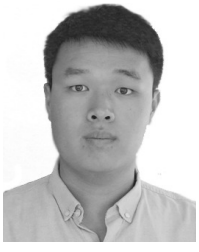
[17] M. Yedla, S. R. Pathakota, and T. M. Srinivasa, "Enhancing K-means clustering algorithm with improved initial center," *Int. J. Comput. Sci. Inf. Technol.*, vol. 1, no. 2, pp. 121–125, 2010.

[18] Y. Zhang and Y. Liu, "A collaborative filtering algorithm based on time period partition," in *Proc. 3rd Int. Symp. Intell. Inf. Technol. Secur. Inform. (IITSI)*, 2010, pp. 1–4.

[19] Y. Zhang *et al.*, "Daily-aware personalized recommendation based on feature-level time series analysis," in *Proc. 24th Int. Conf. World Wide Web ACM*, 2015, pp. 1373–1383.

[20] F. Zhao, Y. Xiong, X. Liang, X. Gong, and Q. Lu, "Privacy-preserving collaborative filtering based on time-drifting characteristic," *Chin. J. Electron.*, vol. 25, no. 1, pp. 20–25, 2016.

BAOSHAN SUN received the Ph.D. degree from Tianjin Polytechnic University, China. He joined the School of Computer Science and Software Engineering, Tianjin Polytechnic University, as an Associate Professor. He has accomplished two scientific research projects at national, three projects at ministerial level, and three transverse projects. He has published more than ten papers in major academic journals, participated in the publishing of three textbooks of computer science, and won three patents for his inventions. He has undertaken three national research projects and two provincial and ministerial projects. His current research interests include the studies of machine learning, national language processing, and computer networks.



LINGYU DONG is currently pursuing the M.Eng. degree with the School of Computer Science and Software Engineering, Tianjin Polytechnic University. He is currently involved in machine learning algorithms for recommendation systems. His research interests include machine learning, neural network, data mining, and recommendation systems.