Time Weight Collaborative Filtering

Yi Ding
School of Information Technology and Electrical Engineering
University of Queensland
ITEE, University of Queensland, QLD 4072, Australia
ding@itee.uq.edu.au

Xue Li
School of Information Technology and Electrical Engineering
University of Queensland
ITEE, University of Queensland, QLD 4072, Australia
xueli@itee.uq.edu.au

ABSTRACT

Collaborative filtering is regarded as one of the most promising recommendation algorithms. The item-based approaches for collaborative filtering identify the similarity between two items by comparing users' ratings on them. In these approaches, ratings produced at different times are weighted equally. That is to say, changes in user purchase interest are not taken into consideration. For example, an item that was rated recently by a user should have a bigger impact on the prediction of future user behaviour than an item that was rated a long time ago. In this paper, we present a novel algorithm to compute the time weights for different items in a manner that will assign a decreasing weight to old data. More specifically, the users' purchase habits vary. Even the same user has quite different attitudes towards different items. Our proposed algorithm uses clustering to discriminate between different kinds of items. To each item cluster, we trace each user's purchase interest change and introduce a personalized decay factor according to the user own purchase behaviour. Empirical studies have shown that our new algorithm substantially improves the precision of item-based collaborative filtering without introducing higher order computational complexity.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*Information Filtering*; I.2.6 [**Artificial Intelligence**]: Learning—*Knowledge Acquisition*

General Terms

Algorithms, Experimentation, Performance, Human Factors

Keywords

collaborative filtering, item-based approach, time weight

1. INTRODUCTION

Nowadays the amount of information in the world is increasing far more quickly than our ability to process it. Collaborative filtering and content-based filtering are two prevailing techniques that help users filter useless information. Content-based filtering analyses an item's attributes to make recommendation, while collaborative filtering uses historical data on user preferences to predict items that a user might like. To date, collaborative filtering is best known for its use on e-commerce web sites. It has also been widely used in other areas, for example, filtering Usenet News, recommending TV shows and web personalization.

Over the years, various approaches for collaborative filtering have been developed. Generally, there are categorized into two classes: memory-based algorithms and model-based algorithms [3]. Memory-based algorithms represent the classical trend in collaborative filtering. This type of algorithm includes user-based algorithms and item-based algorithms. User-based algorithms represent a user as a vector in the item-space [12],while item-based algorithms represent an item as a vector in the user-space [13]. On the other hand, model-based algorithms deploy the data to build a model that is then used for predictions. The aspect model(AM) [7] and the personality diagnosis model(PD) are two popular model-based algorithms. AM is a new probabilistic graphical model that combines a Poisson mixture with a latent aspect class model to model users' purchase behaviour. PD is a different case, as it treats each user in the training set as an individual model. It computes the likelihood for the test user to be in the model of each training user and then uses the training users as the estimators [8]. Contrary to model-based algorithms, memory-based approaches are usually simpler and require less offline computation,performing better when the training data is small [14]. Compared to user-based algorithms, item-based algorithms can dramatically improve the scalability of collaborative filtering and provide better quality [13]. Up until now, item-based algorithms have been widely used in many applications in the real world, such as at Amazon.com.

Item-based algorithms have achieved success both in research and practice. However, in these approaches data collection is regarded as static. Ratings produced at different times are weighted equally. That is to say, changes in user purchase interest are not taken into consideration. For example, a man previously liked thrillers. So he rated *The 39 Steps* , a thriller released in 1935, the highest score 5 three years ago. But he changes his interests as time goes by. Currently he dislikes thrillers. He rated *The Bourne Identity*, another thriller released in 2002, the score 2 a month ago. Classic collaborative filtering algorithms suppose these two scores have the same weights in predicting the user future preference for thrillers. However, we argue that the target user's recent ratings reflect his/her future preferences more than the old ratings. A good collaborative filtering algorithm should gradually decay the influence of old data and predict the user future preferences precisely. From the perspective of intuition, an item that was rated recently by a user should have a bigger impact on the user's prediction than an item that was rated a long time ago. To address this problem, a simple and popular approach is the use of sliding windows. The approach just uses new data in current sliding window and discards old data. However, this approach can aggravate sparsity. Sparsity is a well-known problem in collaborative filtering. It refers to the fact that most users do not rate many items and hence the user-item rating matrix is very sparse. Using only new data can make this sparsity factor more severe, thereby degrading the precision of recommendation system. In this paper, we present a novel time weight algorithm. We find appropriate time weights for different items that are rated at different times. The more recent the data is, the more it contributes to predicting the items. More specifically, the users' purchase habits vary. Some people's interests change with time quickly, while others persist in what they like for a long time. Even the same user has quite different attitudes towards different items. Purchase interest in some kinds of items last much longer than other kinds of items. We assume that the same user has the similar preferences and similar purchase interest changes for the similar items. So we use clustering to discriminate between different items. There exists strong relevancy between these items belonging to the same cluster. Then to each item cluster, we trace each user's purchase interest change and compute automatically a personalized decay factor for each user according to his/her own purchase behaviour.

The remainder of the paper is organized as follows. Section 2 briefly presents some of the research literature related to collaborative filtering. In section 3, we propose a novel time weight collaborative filtering algorithm and describe different sub-tasks of the algorithm in detail. Section 4 presents our experimental work,providing details of our data set, evaluation metrics, results of different experiments and discussion of the results. In the final section, we make a conclusion and point out directions for future work.

2. RELATED WORK

Recommendation systems started attracting major research interest during the early nineties [5]. Since that time, many techniques have been explored in collaborative filtering. Maritza L. *et al* made a comparison of a number of different algorithms, namely memory-based algorithm, dependency-networks algorithm, online learning algorithm and support vector machine and conducted many experiments in [10]. The results from these experiments showed that for a wide range of conditions, memory-based algorithm outperforms support vector machine, dependency-networks and online methods.

In collaborative filtering, the prediction accuracy is a key issue. It influences the prevalence of the recommendation systems to a great extent. So research on collaborative filtering is primarily focused on improving this element. Rong Jin *et al* presented an approach of normalizing ratings of different users to the same scale [9] and an optimization algorithm to automatically compute the weights for different items based on rating discrepancies among different users [8]. Chun Zeng *et al* adopted two techniques: a matrix conversion method for similarity measure and an instance selection method [19]. Cai-Nicolas Ziegler *et al* exploited taxonomic background knowledge for the computation of personalized recommendations [21]. Many model algorithms have also been developed to improve the accuracy of collaborative filtering. For example, Thomas Hofman presented a Gaussian probabilistic latent semantic model in [7]. The observed user ratings can be modelled as a mixture of user communities or interest groups. Users are able to participate probabilistically in one or more groups. The user community is denoted by a hidden variable.

Unfortunately, as far as we know, few papers have focused on the temporal feature of ratings in collaborative filtering. In [16], the authors proposed that a movie's production year, which reflects the situational environment in which the movie is filmed, might significantly affect target users' future preferences. Loren Terveen *et al* defined users' preferences using their personal history [17]. Kazunari Sugiyama *et al* explored a type of time-based collaborative filtering with detailed analysis of user's browsing history in one day [15]. Yanchang Zhao *et al* proposed using decaying function to tackle time series data [20]. However, the recency of ratings has not been studied so far. In fact, with the fast growth of e-commerce, many collaborative filtering applications have been fielded for a long time. For example, in [2], the authors described a TiVo television show collaborative recommendation system which started four years ago. It has currently accumulated approximately 100 million user ratings, some of which are very old. Since the value of these very old ratings is questionable, we should seek to develop an algorithm that will decay the influence of these.

Furthermore, recently mining concept-drifting data has received growing interest. Concept-drifting means the concept that we try to learn from the data is constantly evolving. This is very similar to the interest-drifting in user purchase history. There have been some efforts dedicated to data selection in the environment of concept-drifting [4][18]. In [4], the author pointed out that using old data blindly is not better than "gambling". In other words, using old data unselectively helps produce a more accurate hypothesis only if there is no concept-drifting and the amount of old data chosen arbitrarily just happen to be right. Our proposed algorithm can select data wisely and trace changes in user purchase interest. This algorithm alleviates the problem of concept-drifting in the situation of collaborative filtering.

3. TIME WEIGHT ALGORITHMS

The main idea of our time weight algorithm is to find appropriate time weights for items in order that the items rated recently are able to contribute more to the prediction of the recommendation items. Intuitively, we can surmise that recent data corresponds to the latest user purchase interest. More recent data should have higher value in the time weighting. In the following sections, we first describe the definition of collaborative filtering algorithms, and then we introduce the time function to item-based collaborative filtering algorithms. Finally, we propose the new algorithm.

3.1 Item-based Collaborative Filtering Algorithms

Collaborative filtering problem can be defined as follows: Given a database D as a tuple $< U_i, I_j, O_{ij}, T_{ij} >$, where U_i identifies the i-th user of the system, I_j identifies the j-th items of the system, O_{ij} represents the i-th user's opinion on the j-th item and T_{ij} represents producing time of the opinion, find a list of k recommended items for each user U.

The classic item-based collaborative filtering algorithms have two phases:

Phase 1 Similarity Computation. There are three main approaches to compute the similarity between two items:

Cosine similarity: An item is considered as a vector in the m dimensional user-space. The similarity between different items is measured by computing the cosine of the angle between different vectors as:

$$sim(I_a, I_b) = \cos(\vec{a}, \vec{b}) = \frac{\vec{a} \cdot \vec{b}}{|\vec{a}||\vec{b}|} = \frac{\Sigma_i^m O_{ia} \times O_{ib}}{\sqrt{\Sigma_i^m O_{ia}^2}\sqrt{\Sigma_i^m O_{ib}^2}} \tag{1}$$

where I_a identifies the a-th item of the system.

O_{ia} represents the i-th user opinion on the a-th item.

Pearson correlation coefficient: The similarity between different items is measured as:

$$sim(I_a, I_b) = \frac{\Sigma_i^m (O_{ia} - \overline{O_i}) \times (O_{ib} - \overline{O_i})}{\sqrt{\Sigma_i^m (O_{ia} - \overline{O_i})^2}\sqrt{\Sigma_i^m (O_{ib} - \overline{O_i})^2}} \tag{2}$$

where $\overline{O_i}$ is the average of the i-th user's ratings.

Conditional Probability-Based Similarity: An alternate way of computing the similarity between different items is to use a measure that is based on the conditional probability of purchasing one of the items given that the other has already been purchased. That is,

$$P(j|i) = \frac{Freq(ij)}{Freq(i)} \tag{3}$$

where $Freq(i)$ is the number of users that have purchased the i-th item

Phase 2 Preference prediction. The prediction of the preference for a given object can be computed by using the sum of the ratings of the user to items weighted by the similarity between different items as:

$$O_{ij} = \frac{\Sigma_{c=1}^k O_{ic} \cdot sim(I_j, I_c)}{\Sigma_{c=1}^k sim(I_j, I_c)} \tag{4}$$

where I_j identifies the j-th item,

I_c identifies the nearest neighbors of the j-th item,

O_{ij} represents the i-th user's opinion on the j-th item.

3.2 Time Function

As we assumed before, the user purchase interest is sensitive to time. The recommendation process should assign a greater level of importance to recent data. So we propose that in the phase of Preference Prediction, each rating is assigned a weight defined by a function f(t) to the time t. That is to say, in our proposed algorithm, the Equation (4) is modified as:

$$O_{ij} = \frac{\Sigma_{c=1}^k O_{ic} \cdot sim(I_j, I_c) \cdot f(t_{ic})}{\Sigma_{c=1}^k sim(I_j, I_c) \cdot f(t_{ic})} \tag{5}$$

where t_{ic} represents the time the user's opinion O_{ic} was produced.

Furthermore, we assume that the time function f(t) is a monotonic decreasing function, which reduces uniformly with time t and the value of the time weight lies in the range (0,1). In other words, all the data contribute to the recommendation items, while the most recent data contributes the most. The old data reflects users' previous preferences. It should have small weights in the prediction of recommendation. In our proposed algorithm, we choose an exponential form for the time function to achieve the goal. The exponential time function is widely used in many applications in which it is desirable to gradually discay the history of past behaviour as time goes by [1]. Firstly, we define a half-life parameter T_0 as:

$$F(T_0) = (1/2)f(0) \tag{6}$$

That is to say, the weight reduces by 1/2 in T_0 days.
Then we define the decay rate λ as:

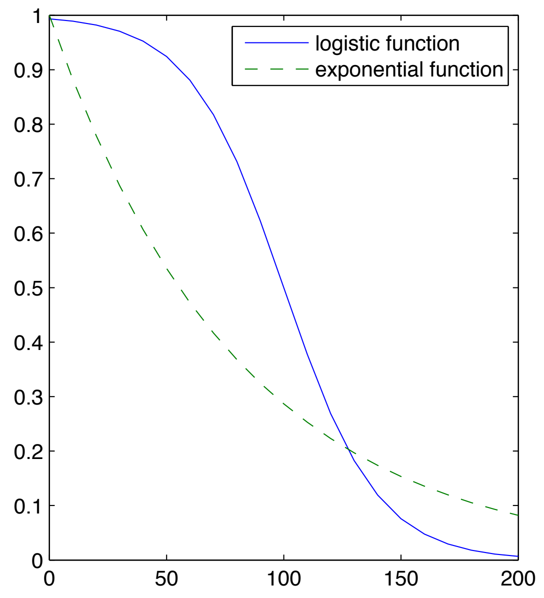
$$\lambda = \frac{1}{T_0} \tag{7}$$

The time function is as follows:

$$f(t) = e^{-\lambda \cdot t} \tag{8}$$

From the Equation (8), we can observe that the value of the time function is in the range $(0, 1)$, and it reduces with time. The more recent the data, the higher the value of the time function is. The exponential function satisfies our needs well. However, there are still other time functions, for example, logistic function. From the Figure (1), we can observe the difference between these two functions. The exponential function reduces at a reducing rate. The gradient of the curve at data point that is close to zero is steeper than data point that is far away from zero. Here, x=0 represents current data. The higher the value of x is, the older the data is. However, to the logistic function, the gradient of the curve at the middle data point is steepest. Obviously, we emphase the user's latest purchase interest and focus on the most recent data. In this case, the exponential function is more suitable. At the same time we analyse the half-life parameter T_0 in detail. Conceptually, the aim of designing a half-life parameter is to define the rate of decay of the weight assigned to each data point. From the Equation (6), we can see that T_0 is inveresely proportional to λ. The lower the value of T_0, the higher the value of λ. The higher the value of λ, the faster old data decays and the lower the importance of the historical information compared to more recent data. The value of parameter T_0 decides the decay rate of old data. From the Figure (2), we can see the different curves of time function when the value of parameter T_0 is different. Obviously, we should select different values of parameter T_0 under different circumstances and the selection of parameter T_0 is a key issue to the performance of our algorithm. In the real world, the decay rate of old data is decided by how frequently the user purchase interest changes. If the user preference for the type of items is consistent, old ratings related to the type of items can help improve accuracy of predicting future preference for the type of items. In this case, we should decay the old data slowly and assign a high value to T_0. The deviation between time weights is comparatively small. So in predicting future user preference for this type of items, similarity plays a more important role than time weight. Conversely, if the user preference for the type of items changes frequently and dramatically, we should assign a low value to T_0. This means that old ratings related to the type of items cannot predict the user future preference for the type of items precisely. These old data should decay as quick as possible. In other words, time weights would contribute more to predicting the future preference compared to when the user preference is consistent. So we should select the appropriate parameter T_0 to precisely predict the user future preference according to the user personalized purchase history.

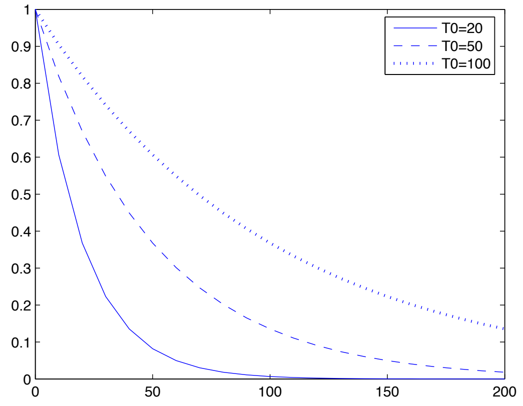
Figure 1: A Comparison of Two Functions



3.3 Learning Parameters

We need to find the appropriate value of parameter T_0 to precisely predict the user future preference, thereby improving the performance of our proposed algorithm. However, the users' purchase habits vary. Even the same user has quite different attitudes towards different kinds of items. Purchase interest in some kinds of items last much longer than other kinds of items. Furthermore, it is not feasible interactively providing a corresponding value of parameter T_0 for each user and each items. We assume that the same user has the similar preferences and similar purchase interest changes for the similar items. It means to one user, the similar items have the similar decay rates of old data. So we propose to compute the corresponding values of parameter T_0 for each user and each cluster of items according to the user personalized purchase history. To realize the goal, we first use simple K-Means clustering approach to summarize items into different clusters through the rating information [11]. There exists strong relevancy between these items belonging to the same cluster. Then to each item cluster, we take the leave-one-out approach to compute automatically the personalized value of parameter T_0. In other words, each time we leave out one item rated by a user from one item cluster. We are then able to use the omitted item to measure how well the user purchase behaviour on the item can be explained. To measure this, we introduce the mean absolute error (MAE). MAE is a popular metric in collaborative filtering. It computes the average absolute deviation of recommendations from their true user-specified values. For an item cluster, each user has many ratings for items belonging to this cluster. The MAE can be computed as:

Figure 2: the curves of time function using different T_0



$$MAE = \frac{\Sigma_{i=1}^{N} |p_i - q_i|}{N} \tag{9}$$

Here N identifies the number of the user's ratings in a cluster

p_i identifies the predicted rating for the i-th item

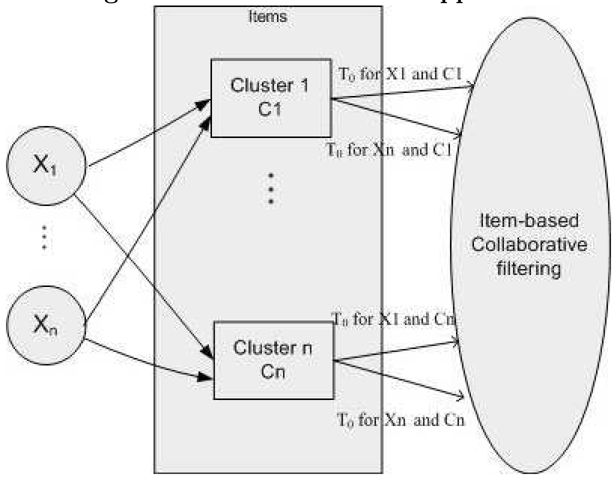
q_i identifies the user's true rating for the i-th item

In our proposed algorithm for each user, the prediction P_i for each item is parameterized by T_0. So we use MAE(T_0) to emphasize that the MAE is parameterized by T_0. By minimizing the value of MAE, we will find optimal parameter T_0. Formally, the optimization problem is stated as follows:

$$\arg\min MAE(T_0) \tag{10}$$

However, finding the optimal solution to Equation (10) is rather difficult due to the non-concave objective function. We derive an optimization strategy for Equation (10) that uses approximate value. First, we set interactively an upper bound and a lower bound of parameter T_0. We then

Figure 3: Overview of Our Approach



discretize the value of parameter T_0. At last, we scan all the possible values and select the optimal value assigned to parameter T_0. By this method, to each item cluster we can approximately compute the personalized optimal value of T_0 for each user. The Figure 3 is an overview of our approach.

3.4 Building the Model

Our new time weight collaborative filtering algorithm is based on item-based collaborative filtering. In our proposed algorithm, the prediction of the preference for a given object can be computed by using the sum of the ratings of the user to items weighted by the similarity between different items and the time weight. The input to this algorithm is the N $\times$ M user-item matrix C, N $\times$ M user-time matrix T that represents the time users' opinions on items were produced, a parameter n that specifies the number of item-to-item similarities that will be stored for each item, a parameter K that specifies the number of item clusters and a parameter l that denotes the number of recommendation items. If the i-th user has no rating on the j-th item in the system, the values of C_{ij} and T_{ij} are both equal to zero. The output is an N $\times$ l matrix N that stores the N $\times$ l recommendation items. In it, every row represents every user. For each user there are l recommendation items. The algorithm is shown in Figure 4.

```
TimeWeightCollaborativeFiltering (C,T,n,K,l)
1. Matrix M←ComputeItemSimilarity(C,n);
2. ItemSimpleKMeans(C,K);
3.     for (ClusterNo = 1; ClusterNo<=K;
4.      ClusterNo++){
5.       for (UserNo = 1; UserNo<=N; UserNo++){
6.        Array P←LearningParameters(C,M,T,n,K,l);}
7.       }
8.     Matrix N←
9.     PredictRecommendationItems(M,T,P,l);
```

Figure 4: Time Weight Collaborative Filtering Algorithm

4. EXPERIMENTS

We have conducted a set of experiments to examine the performance of our new time weight collaborative filtering algorithm. Particularly, we address the following two issues:

1 How does the parameter T_0 influence the prediction accuracy?

T_0 is an important parameter for our new algorithm, which defines the decay rate λ of old data. That is to say, the parameter T_0 decides the importance of the historical data. Under different circumstances, the value of T_0 should be changed to meet different requirements. Experiments change the value of parameter T_0 and are conducted to examine the impact of parameter T_0 on the final performance of the new algorithm.

2 How is our time weight algorithm compared to the existing collaborative filtering algorithm?

Our approach is compared to the classic item-based algorithm. These two algorithms are both incorporated into the Pearson Correlation Coefficient method.

4.1 Experiment Design

We use two datasets in our experiments: EachMovie [1] and GroupLens [2]. EachMovie and GroupLens have been the most widely used common datasets in collaborative filtering research projects. EachMovie was collected during 18 months where 72,916 users rated 1628 movies [6]. GroupLens consisted of 1,000,209 ratings for 3900 movies by 6040 users. The global statistics of these two datasets used in our experiments are showed in Table 1.

Table 1: characteristics of EachMovie and GroupLens

Name of database	EachMovie	GroupLens
Number of Users	1394	1558
Number of Items	1628	50
Avg. ♯ of rated Items/User	145.6	20
Number of Ratings	6	5

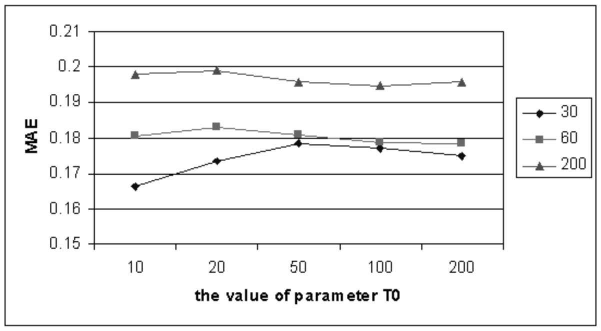
We alter the training size to be the first 30, 60 or 200 users for training. And we also utilize two protocols, *All But One* and *Given k*. In *All But One*, the newest rated items for each user are used for testing. But in the second protocol, *Given k*, we select *k* ratings from each user as the observed ratings and then predict the remaining ratings. By varying the number of training users and the number of given items, we can test our proposed algorithm for different configurations. In all the experiments the number of the nearest neighbours is set to 30. The evaluation metric used in our experiment is the mean absolute error (MAE).

4.2 Experiment(1): impact of parameter T_0

In the first experiment, we vary the parameter T_0 from 10, 20, 50, 100 to 200. The value of λ is 0.1, 0.05, 0.02, 0.01 and 0.005 respectively. The results of using different constant T_0 are demonstrated in Table 2, Table 3 and Figure 5. Since the parameter T_0 controls the decay rate of historical

[1]www.research.compaq.com/SRC/eachmovie
[2]www.cs.usyd.edu.au/Research/GroupLens/data/million

Figure 5: MAE using different T_0 on EachMovie in ***All But One***



information, we can see that the parameter T_0 dramatically influences the performance of the algorithm and under different configurations to obtain the best performance we should assign different values to the parameter T_0.

Table 3: MAE using different T_0 on GroupLens in ***All But One*****. A smaller value means a better performance**

Training Users Size	T_0	λ	All but One
30	10	0.1	0.8941
	20	0.05	0.8918
	50	0.02	0.8751
	100	0.01	0.8557
	200	0.005	0.8511
	auto	auto	0.8626
60	10	0.1	0.7449
	20	0.05	0.7638
	50	0.02	0.7721
	100	0.01	0.777
	200	0.005	0.7872
	auto	auto	0.7785
200	10	0.1	0.7166
	20	0.05	0.7316
	50	0.02	0.7552
	100	0.01	0.7844
	200	0.005	0.8161
	auto	auto	0.7889

Another observation is that the approach that the value of parameter T_0 is fixed does not guarantee the improvement of the performance of collaborative filtering and in some circumstances the performance is not good. The experimental results confirm that the users' purchase habits vary and assigning the same value of parameter T_0 to different users is inappropriate. In our proposed algorithm, the value of parameter T_0 can be automatically computed according to the user's history of past behaviour. The results show that our new algorithm is always close to the optimal performance for all configurations.

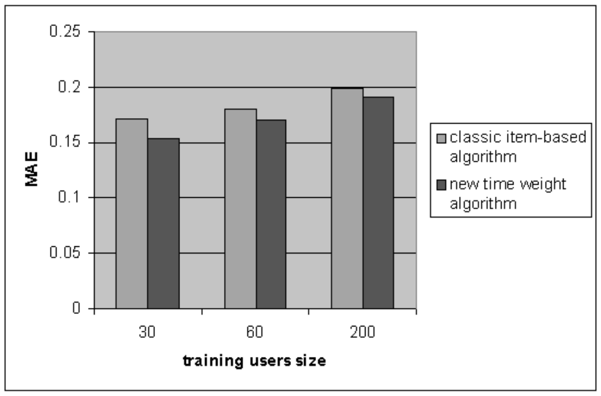
4.3 Experiment(2): comparison to the classic item-based algorithm

In this experiment, we compare our new time weight algorithm to the classic item-based algorithm. The parameter T_0 is automatically computed for different users and different item clusters. The results are presented in Table 4, Table 5 and Figure 6. Obviously, our new algorithm is able to boost the prediction accuracy for all configurations.

Table 5: MAE using different algorithms on GroupLens in ***All But One*****. A smaller value means a better performance**

Training Users Size	Methods	All but One
30	Classic	0.8637
	time weight	0.8626
60	Classic	0.8104
	time weight	0.7785
200	Classic	0.8508
	time weight	0.7889

Figure 6: MAE using different algorithms on EachMovie in ***All But One***



4.4 Complexity Analysis

Classic item-based collaborative filtering's scalability is that it can create the expensive similar items table offline. The offline computation of the similar table is extremely time intensive, with $O(N^2M)$ as worst case. Here, N represents the number of items. M represents the number of users. In our proposed algorithm, the offline computation is even more expensive. Our algorithm needs to compute similar items table and the personalized value of parameter T_0 offline. So the offline complexity is equal to $O(N^2M + NKl + MNKt)$. Here K means the number of item clusters. l means the iterative times in clustering items. t means the number of possible values of parameter T_0. Compared to computing similar items table, computing the personalized value of parameter T_0 has a cheaper computation.

Our proposed algorithm's online complexity is the same as classic item-based algorithm. The online component is just looking up similar items the user's ratings. So our algorithm scales independently of the number of users and items. The algorithm is fast even for extremely large data sets.

Table 2: MAE using different T_0 on EachMovie. A smaller value means a better performance

Training Users Size	T_0	λ	All but One	10 Items Given	20 Items Given	30 Items Given
30	10	0.1	0.1664	0.2056	0.1400	0.1300
	20	0.05	0.1736	0.2063	0.1635	0.1423
	50	0.02	0.1784	0.2063	0.1816	0.1633
	100	0.01	0.1771	0.2062	0.1866	0.1723
	200	0.005	0.1747	0.2061	0.1888	0.1469
	auto	auto	0.1538	0.2047	0.1381	0.1260
60	10	0.1	0.1807	0.1868	0.1313	0.1596
	20	0.05	0.1830	0.1871	0.1459	0.1605
	50	0.02	0.1809	0.1871	0.1636	0.1709
	100	0.01	0.1788	0.1871	0.1692	0.1784
	200	0.005	0.1785	0.1870	0.1717	0.1823
	auto	auto	0.1707	0.1862	0.1303	0.1596
200	10	0.1	0.1980	0.1938	0.1891	0.2057
	20	0.05	0.1991	0.1940	0.1825	0.1951
	50	0.02	0.1959	0.1943	0.1888	0.1900
	100	0.01	0.1947	0.1945	0.1912	0.1908
	200	0.005	0.1958	0.1945	0.1938	0.1915
	auto	auto	0.1912	0.1942	0.1890	0.1901

Table 4: MAE using different algorithms on EachMovie. A smaller value means a better performance

Training Users Size	Methods	All but One	10 Items Given	20 Items Given	30 Items Given
30	Classic	0.1715	0.2060	0.1908	0.1815
	time weight	0.1538	0.2047	0.1381	0.1260
60	Classic	0.1812	0.1870	0.1741	0.1863
	time weight	0.1707	0.1862	0.1303	0.1596
200	Classic	0.1985	0.1946	0.1955	0.1924
	time weight	0.1912	0.1942	0.1890	0.1901

5. CONCLUSION AND FUTURE WORK

In this paper, we present a new time weight collaborative filtering algorithm. Unlike the non-time weighted algorithms, we utilize a predefined time function for different items. The main idea is to predict prcisely user future purchase interest by deploying time weight. The item that was rated recently by a user should have a bigger impact on the user's prediction than an item that was rated long time ago. At the same time we learn users' rating behavior to find the appropriate personalized parameter for each item cluster. Empirical studies have shown that our new algorithm can definitely improve the precision of item-based collaborative filtering algorithms.

Our further work is to study collaborative filtering algorithms on streaming data. This issue brings us a new challenge that data are all contained not in a database ready for random access but are seen once only from online resources.

6. ACKNOWLEDGMENTS

The work reported in this paper was funded in part by the Australian Research Council - Discovery Project Grant (ARC DP0558879), and computational resources used in this work were provided by the Queensland Parallel Supercomputing Foundation (QPSF).

7. REFERENCES

[1] C. C. Aggarwal, J. Han, J. Wang, and P. S. Yu. A framework for projected clustering of high dimensional data streams. In *VLDB*, pages 852–863, 2004.

[2] K. Ali and W. v. Stam. Tivo: making show recommendations using a distributed collaborative filtering architecture. In *Conference on Knowledge Discovery in Data*, pages 394 – 401, Seattle, WA, USA, 04.

[3] J. S. Breese, D. Heekerman, and C. Kadic. Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence(UAI)*, 1998.

[4] W. Fan. Systematic data selection to mine concept-drifting data streams. In *Proceedings of the 2004 ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 128 – 137, Seattle, WA, USA, 2004.

[5] D. Goldberg, D. Nichols, B. M. Oki, and D. Terry. Using collaborative filtering to weave an information tapestry. *communications of the ACM*, 35(12):61–70, 1992.

[6] J. L. Herlocker, J. A. Konstan, L. G. Terveen, and J. T. Riedl. Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems (TOIS)*, Volume 22(Issue 1):5 – 53, 2004.

[7] T. Hofmann. Collaborative filtering via gaussian probabilistic latent semantic analysis. In *Proceedings of the 26th annual international ACM SIGIR*

conference on Research and development in informaion retrieval, pages 259 – 266, Toronto, Canada, 2003.

[8] R. Jin, J. Y. Chai, and L. Si. An automatic weighting scheme for collaborative filtering. In *Annual ACM Conference on Research and Development in Information Retrieval*, pages 337 – 344, 2004.

[9] R. Jin and L. Si. A study of methods for normalizing user ratings in collaborative filtering. In *Annual ACM Conference on Research and Development in Information Retrieval*, pages 568 – 569, 2004.

[10] M. L., C. N., and J. d. J. Perez-Alcazar. A comparison of several predictive algorithms for collaborative filtering on multi-valued ratings. In *ACM symposium on Applied computing*, pages 1033 – 1039, 2004.

[11] Q. Li and M. Zhou. Research and design of an efficient collaborative filtering predication algorithm. In *Parallel and Distributed Computing, Applications and Technologies, 2003. PDCAT'2003*, pages 171– 174, 2003.

[12] P. Resnick, N. Iacovou, M. Suchak, P. Bergstorm, and J. Riedl. Grouplens: an open architecture for collaborative filtering of netnews. In *ACM conference on Computer supported cooperative work*, pages 175 – 186, 1994.

[13] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Item-based collaborative filtering recommendation algorithms. In *International World Wide Web Conference*, pages 285 – 295, 2001.

[14] L. Si and R. Jin. flexible mixture model for collaborative filtering. In *the twentieth international conference on machine learning (ICML-2003)*, washington DC, 2003.

[15] K. Sugiyama, K. Hatano, and M. Yoshikawa. Adaptive web search based on user profile constructed without any effort from users. In *Proceedings of the 13th international conference on World Wide Web*, pages 675 – 684, New York, NY, USA, 2004.

[16] T. Y. Tang, P. Winoto, and K. C. C. Chan. On the temporal analysis for improved hybrid recommendations. In *Web Intelligence, 2003. WI 2003. Proceedings. IEEE/WIC International Conference on*, pages 214 – 220, 2003.

[17] L. Terveen, J. McMackin, B. Amento, and W. Hill. Specifying preferences based on user history. In *Conference on Human Factors in Computing Systems*, pages 315 – 322, Minneapolis, Minnesota, USA, 2002.

[18] H. Wang, W. Fan, P. S. Yu, and J. Han. Mining concept-drifting data streams using ensemble classifiers. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 226 – 235, Washington, D.C, 2003.

[19] C. Zeng, C.-X. Xing, and L.-Z. Zhou. Similarity measure and instance selection for collaborative filtering. In *International World Wide Web Conference*, pages 652 – 658, 2004.

[20] Y. Zhao, C. Zhang, and S. Zhang. A recent-biased dimension reduction technique for time series data. In *Advances in Knowledge Discovery and Data Mining: 9th Pacific-Asia Conference, PAKDD 2005*, volume 3518 / 2005. Lecture Notes in Computer Science, 2005.

[21] C.-N. Ziegler, G. Lausen, and L. S. Thieme. Taxonomy-driven computation of product recommendations. In *the Thirteenth ACM conference on Information and knowledge management*, pages 406 – 415, Washington, D.C., USA, 2004.