Reconstruct Dynamic Systems from Large-Scale Open Data

Kun-Hung Tsai*, Chia-Yu Lin*, Li-Chun Wang*, and Jian-Ren Chen†
*Department of Electrical and Computer Engineering, National Chiao Tung University, Taiwan
E-mail: moonape1226.cm02g@nctu.edu.tw, sallylin0121@gmail.com, lichun@g2.nctu.edu.tw
†Industrial Technology Research Institute, Taiwan
E-mail: cjr@itri.org.tw

***Abstract*—With the prosperity of e-commerce, on-line vendors use recommendation systems in different fields. Classic recommendation algorithms are designed assuming that data is stationary and will not change over time. However, since the scale and variability of data are growing gradually, these methods will encounter the issues of the memory deficient and the out-of-date model, which degrade the recommendation accuracy intensively. In addition, retraining the whole model for every new arrival record results in high complexity. In this paper we propose a light-weight adaptive updating method to overcome these issues. Comparing with the explicit feedback recommendation, which asks the customers to express their opinions on the recommended items, the implicit feedback recommendation is easier to collect and non-intrusive way. However, the dynamic time-variant system with implicit feedback has not been seen in the literature. In this paper, we propose a real-time incremental updating algorithm (RI-SGD) to deal with time-variant systems based on the implicit feedback. We compare our method with methods that retraining the whole model and show that our method costs less than 1% of the retraining time with a competitive accuracy.**

***Index Terms*—Recommendation System, Implicit feedback, Adaptive Algorithm, Concept Drift**

I. Introduction

Recommendation systems analyze data collected from users such as users' rating on items, web-browsing data and music play-lists to reflect history personal preference of users and build a model to predict whether users will buy or like items. Several methods including content-based recommendation, user and item-based collaborative filtering and matrix factorization are proposed to predict ratings to make accurate recommendation.

A. Matrix Factorization

Matrix factorization is proven to be much more memory-efficient than similarity-based method and achieves better accuracy in rating problem [1]. It costs $f*N+f*M$ rather than $N^2/2$ in user similarity-based method. f is the size of latent factor. N, M are number of users and items, respectively. With the size of data growing dramatically, the memory-efficient property of matrix factorization can save much more memory space. [2] uses matrix factorization to deal with rating prediction problem in Netflix competition [3]. Matrix factorization decomposes the user-item record matrix into multiplication of two low-rank matrices and only stores the latent factor of users and items. On the other hand, similarity-based method stores the similarity of two elements, which occupies much memory when size of users or items is large. Moreover, due to the characteristic of matrix factorization, it can help to discover similarity items and users.

As shown in Fig. 1, the record matrix M is decomposed into X and Y, which are the latent factor matrices of users and items respectively. The latent factor vectors represents attributes obviously possess by the items, such as music types, movie genre, etc. Moreover, the factors can help us discover attributes which are uninterpretable by the content itself. For customers, the higher values in the factors, the more interest they have in the attributes. Fig. 2 is an example of two dimensions latent factor model. If the user is in favor of the specific artist, they would have similar location in the 2 dimension space.

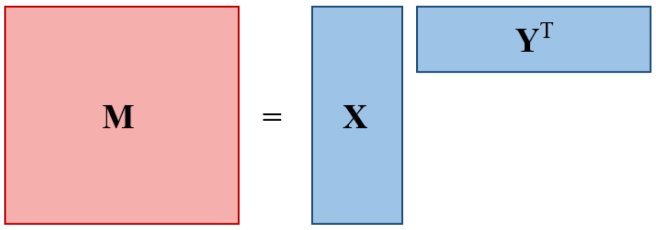


Fig. 1. The model of matrix factorization.

In matrix factorization, each user u is given a vector $x_u \in \mathbf{R^f}$ and each item i is given a vector $y_i \in \mathbf{R^f}$. The values in record matrix M are the preference of users to the items, which can be predicted by $\widehat{r}_{ui} = x_u^T * y_i$. According to [2], we use the values in rating matrix as training data and obtain the model by minimize the error function below.

$$\underset{x^*,y^*}{\text{minimize}} \quad \sum (r_{u,i} - x_u^T y_i)^2 + \lambda(\|x_u\|^2 + \|y_i\|^2) \tag{1}$$

B. Implicit Feedback

Comparing to implicit feedback, explicit feedback such as rating is easy to analyze, but service providers have to ask users to provide their preference, which is a hard and time-consuming process. On the contrary, users generate implicit feedback such as transaction record and web-browsing data all the time. Service providers are able to collect implicit feedback without disturbing users. However, according to [4], there are two challenges when using implicit feedback. Since the value

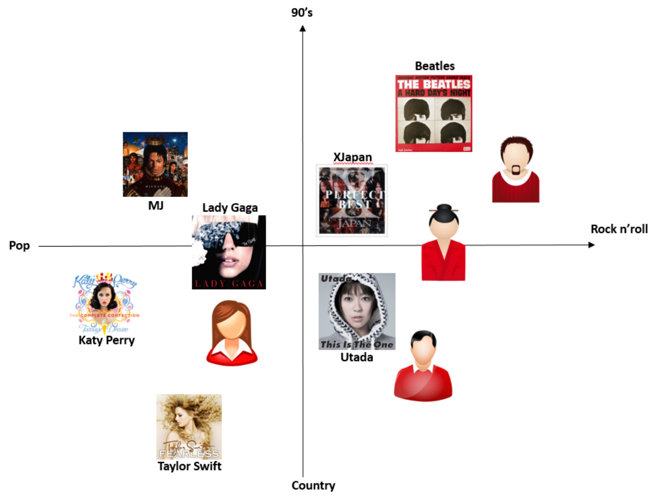


Fig. 2. The latent feature of users and items.

range of implicit feedback is wide, the first challenge is how to transfer these non-negative values into useful preference values. Besides, implicit feedback value will increase overtime. How to update model incrementally becomes the second challenge. To solve these problems, we propose a light-weight adaptive updating method for matrix factorization that deals with implicit feedback at the same time.

C. Real-time Updating Problem

Most of the matrix factorization algorithms assume that the whole data is stored in database and the system is executed in batch mode. That is, the models are trained in a single pass and do not update any more. In the real-time recommendation system, new users and items are added in the system, users' record increases and users' tastes evolve over time. Therefore, the old models may become out of date and make poor recommendation after new data comes into the systems.

[4] revised the original cost function to overcome implicit feedback problem. Although the revised method achieved better accuracy, the author didn't consider how to update the model in real-time. [5] proposed a real-time updating method for explicit feedback recommendation using stochastic gradient descent (SGD). In [5], implicit feedback model had not been considered. [6] solved the real-time updating problem with binary value of implicit feedback. However, the authors replaced the implicit feedback with binary value. Since [6] didn't use the real numerical implicit feedback value, the accuracy in [6] was not high.

Above researches cover part of the issues we discuss, but none of them shows a complete scope of work. Our proposed algorithm not only considers the implicit feedback and real-time updating problem at the same time but is able to make accurate recommendation in short updating time. Besides, we compare our method with models that never updates to show how out-of-date model affects the performance in recommendation system. In addition, we conduct experiments on IBM Infosphere Streams platform [7] to evaluate the performance of our method and other different updating algorithms in streaming data environment. The result shows that our method achieves more than 300 times speedup with competitive accuracy among all updating methods.

The paper is organized as follows. Section II is literal survey. Section III discusses the details of our proposed real-time incremental updating algorithm. The experiment and numerical results are shown in section IV. Finally, we conclude the paper in Section V.

II. Related Work

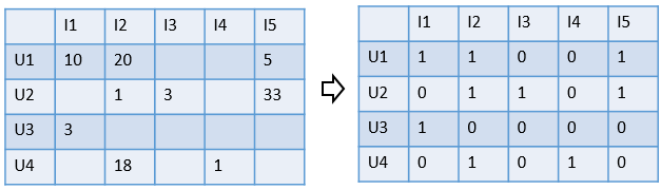
[4] explained the difference between implicit feedback and explicit feedback. The numerical value of implicit feedback represented confidence while the numerical value of explicit feedback showed the preference of users. Since the value of implicit feedback could not be directly mapped to users preference, the methods used to train the model must be revised as well.

[4] revised the cost function in matrix factorization to adapt to implicit feedback in (2). The authors replaced the original rating with binary value p_{ui}, that only indicated whether a user had confidence in the item. The numerical value of implicit feedback r_{ui} was then turned into confidence weighting $c_{ui} = (1 + \alpha r_{ui})$ in the cost function. α is the attenuate parameter for implicit feedback. [4] also proposed alternative least square with weight regularization (ALSWR) to solve the revised cost function with implicit feedback (2).

$$\underset{x^*,y^*}{\text{minimize}} \sum c_{ui}(p_{ui} - x_u^T y_i)^2 + \lambda(\|x_u\|^2 + \|y_i\|^2) \qquad (2)$$

[5] proposed a real-time updating algorithm based on rating-oriented matrix factorization (PMF) and ranking-oriented matrix factorization (RMF) with SGD. The evaluation with real world dataset showed that the algorithm could solve batch-trained problem in recommendation system and the complexity as well as memory space scaled linearly with number of records. However, the methods were designed based on explicit feedback and the author didn't discuss how to deal with the implicit feedback problem.

When new rating records kept flowing into the system, the cost of retraining the whole model would be too large. Therefore, [6] proposed a real-time incrementally updating method using implicit feedback. The authors replaced the elements in the rating matrix with binary values because they thought implicit feedback was difficult to measure preference. Fig. 3 was an example of replacing the rating matrix elements with binary values. No matter what the numerical value was, the authors replaced the element with one if the record existed and replaced with zero if there is no record in the record matrix. The method was based on SGD and the updating time was 2 to 25 times faster than retraining the whole model. However, the algorithm didn't take the numerical value of implicit feedback into account, which decreased model performance intensively.



	I1	I2	I3	I4	I5
U1	10	20			5
U2		1	3		33
U3	3				
U4		18		1	

	I1	I2	I3	I4	I5
U1	1	1	0	0	1
U2	0	1	1	0	1
U3	1	0	0	0	0
U4	0	1	0	1	0

Fig. 3. Replace the rating matrix element with binary value.

None of the related work considered the implicit feedback and real-time updating problems together. Therefore, we proposed a real-time incremental matrix factorization updating method to deal with implicit feedback and real-time update problem.

III. Incremental Matrix Factorization for Implicit Feedback

In this section, we propose a real-time incremental updating algorithm to analyze implicit feedback and update model dynamically. The proposed method is composed of two parts.

The first part is building recommendation model. According to the memory-efficient and high accuracy property, we choose matrix factorization to build model. In order to make matrix factorization deal with implicit feedback effectively, we adopt the cost function (2) and ALSWR in [4] to solve matrix factorization. It's proven in [4] that ALSWR can solve the transformed cost function and generate latent factor vectors x_u and y_i efficiently. The equation is as shown below:

$$\begin{aligned} x_u &= (Y^T C^u Y + \lambda I)^{-1} Y^T C^u p(u) \\ y_i &= (X^T C^i X + \lambda I)^{-1} X^T C^i p(i) \end{aligned} \tag{3}$$

In (3), $X \in \mathbf{N} * \mathbf{f}$ and $Y \in \mathbf{M} * \mathbf{f}$ are latent factor matrices for users and items respectively. f is the size of latent factor and N, M are size of users and items, respectively. $C^u \in \mathbf{N} * \mathbf{N}$ is a diagonal matrix with $C^u_{ii} = c_{ui}$ for each user u and $C^i \in \mathbf{M} * \mathbf{M}$ is a diagonal matrix with $C^i_{uu} = c_{ui}$ for each item i. $p(u) \in \mathbf{N} * \mathbf{1}$ is binary confidence vector for user u and $p(i) \in \mathbf{M} * \mathbf{1}$ is binary confidence vector for item i. The algorithm is applied several time to get an optimized model. However, in real-time system, uninterrupted data flows into system over time so the system has to update the model in minutes or even in seconds. The recalculation cost of ALSWR when updating model update is too expensive. Thus, we find another way to update model.

The second part of the real-time incremental updating algorithm is updating model. Fig. 4 is an example of implicit feedback record matrix. We can find that it is unnecessary to recalculate the whole model every time when the rating matrix changes. Since the element only is affected by the corresponding two latent factor vectors, we adopt the original SGD technique to create a new incrementally updating method. There are two cases for incoming records $r_{u,i}$ from user u on item i entering the system. If the user and the item of the incoming record are already in the system, we add one to corresponding confidence weighting $c_{u,i}$. If the user and the item of the incoming record are new to the system, we not only add one to corresponding confidence weighting $c_{u,i}$ but also set the binary confidence $p_{u,i}$ to one. After modifying the corresponding confidence weighting and binary confidence, both cases calculate the first order gradient of the cost function (2) as presented in (4) and update the corresponding latent factor vectors of user u and item i as shown in (5). Our method only updates the two latent factor vectors related to the new records and retains most of the original model. The updating cost of our method is $O(f)$, which is much less than the cost to retrain the whole model with ALSWR($O(|\Omega|f^2+(m+n)f^3)$).

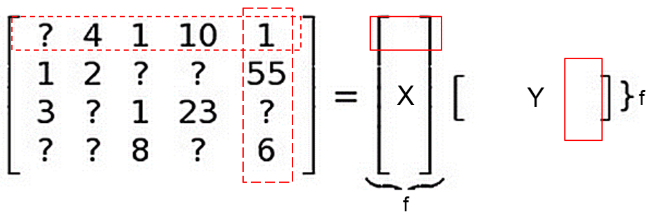


Fig. 4. An example of record changing dynamic system.

$$\begin{aligned} \frac{\partial C(x,y)}{\partial x_u} &= c_{u,i}(p_{u,i} - x_u^T y_i) y_i) + 2\lambda x_u \\ \frac{\partial C(x,y)}{\partial y_i} &= c_{u,i}(p_{u,i} - x_u^T y_i) x_u) + 2\lambda y_i \end{aligned} \tag{4}$$

$$x_u \leftarrow x_u - \frac{\partial C(x,y)}{\partial x_u} \quad y_i \leftarrow y_i - \frac{\partial C(x,y)}{\partial y_i} \tag{5}$$

Fig.5 is our model updating flow. When a new record enters the system, our algorithm measures the total number of records of the user. If the record number are large enough, latent factor model is used to recommend items to the user. Otherwise, if the record number is too small or it is a new user, latent factor model recommends popular items to the user. After making recommendation, we use the modified evaluation method in [8] to evaluate the accuracy of the model. Unlike the method in [8], which measures the accuracy every record, we modify the method and measure the accuracy after 2.5% of the whole data passing through the system to decrease the calculation cost. The model accuracy decrease intensively when concept drift happens. Thus, we also set an accuracy threshold θ in our algorithm to check whether the performance goes down sharply. If the accuracy goes below the threshold, our algorithm updates the model with SVD. Besides, in order to prevent over-fitting, we only update the model with the latest implicit value for users who produce several records on the same item within a certain period of time.

IV. Evaluation and Discussion

A. Datasets

Our dataset is Last.fm Dataset - 1K users [9] from online music provider Last.fm. The dataset is collected from Last.Fm API in [10] and it contains the listening habits of nearly $1,000$ last.fm users till May, 5th 2009. Each record represents a

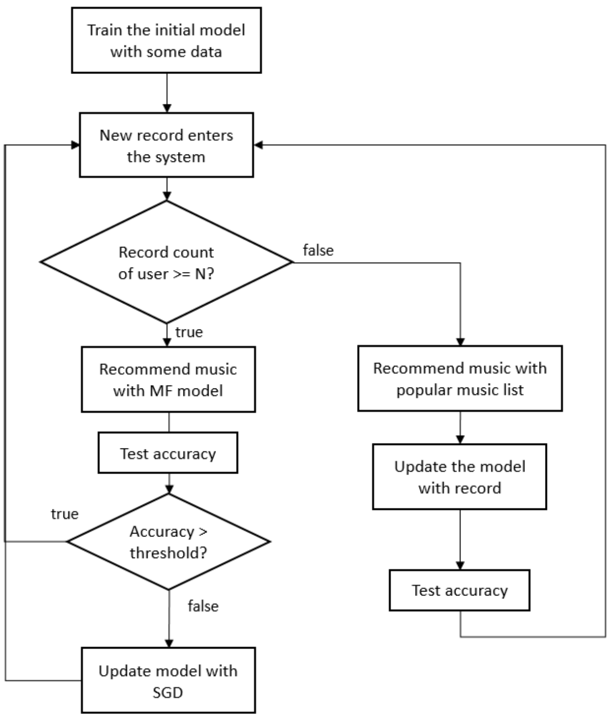

Train the initial model with some data
New record enters the system
Record count of user >= N?
false
true
Recommend music with MF model
Recommend music with popular music list
Test accuracy
Update the model with record
true
Accuracy > threshold?
false
Test accuracy
Update model with SGD

Fig. 5. Model updating flow.

listening event from a user to a song. There are 19 million records in the dataset with songs from 170 thousands of artists. The dataset contains user id, timestamps, artist id, artist name, track id and track name. Since artist id and track id are not recorded in all of the records, we first delete these two columns from the dataset.

Since there are over 1 million songs in the dataset, on average we only have a song repeated 17 times in the dataset. It does not apply to the accumulating characteristic of implicit and and there are many songs are listened only once in the dataset. The two reasons making it too hard to evaluate accuracy based on songs, so we regard the songs as same item if the artist (singer) is the same. To observe the characteristic of dataset, we plot the dataset in log scale as shown in Fig.6. We can discover that still about 30% of the artists have been played only one time in the whole dataset. These records will severely affect the evaluation process of the system. Thus, we delete about $50,000$ one time records from the original dataset to avoid biased performance result.

B. *Methodology and Evaluation*

In the experiments, we will show our proposed incremental updating algorithm can intensively reduce the updating time and keep high accuracy. We revise the source code of Mahout [11] and implement the experiment on IBM Infosphere Streams Platform [7] to evaluate our algorithm. Mahout is a project sponsored by Apache and it implements several classical recommendation algorithms including neighborhood-based method and matrix factorization using ALSWR. Infosphere

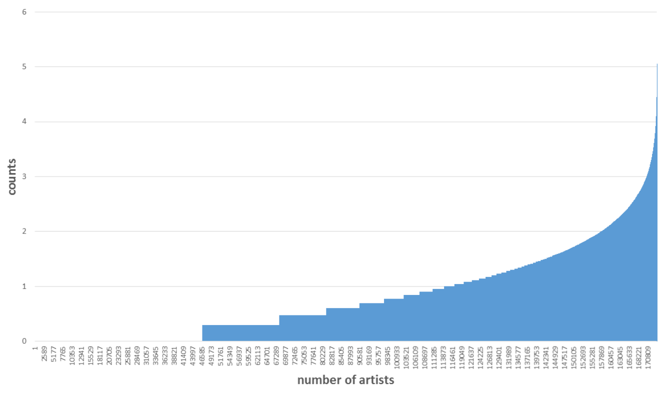


Fig. 6. Histogram of artist count (log) to number of users.

Stream Platform is developed by IBM and it provides an integrated solution for big streaming data application.

At first, we sort the data by timestamps and use ALSWR algorithm to train an initial matrix factorization model by 5% and 10% of the data. The rest records will be sent to the system one by one. The system will update the model with different methodologies: (1) no update, (2) uses ALSWR algorithm to retrain the whole model when every record, (3) applies windowing method to retrain the model by latest n% of data with ALSWR, (4) updates model with our proposed algorithm (RI-SGD), (5) updates with proposed algorithm using kernel function, (6) applies popular item method and (7) the proposed method in [8]. In the fifth methodology, we revise the original cost function with logistic kernel function (6) to evaluate the performance of limiting range of predicted value.

$$C(x,y) : \frac{(p_{u,i} - x_u^T y_i)^2}{(1 + exp(-(1 + r_{ui}))} + \lambda(\|x_u\|^2 + \|y_i\|^2))^{-1} \quad (6)$$

We set the number of latent factor vectors f to 20 for all mechanisms since it is able to preserve most of the information we need in the model. According to a pretest experiment, the threshold θ of our proposed algorithm is set to 0.1 and all other parameters in the experiment have been tuned to get best performance.

To evaluate the performance, each method recommends 10 artists every new record and check whether any of the recommended artists overlaps with the real record. We evaluate the average accuracy when 2.5% (about 500,000 records) of data have been analyzed in the system and plot the performance figures. The method we use to evaluate the accuracy is based on [8] and the equation is showed below. We plus 1 to hit count when recommended artists overlaps with the real record.

$$Accuracy = hitcount/number\ of\ records \quad (7)$$

C. *Numerical result and observation*

1) Time: Fig.7 and Fig.8 show the updating time of using 5% and 10% data to train an initial matrix factorization model. We define 2.5% of data as one block in our figure. In the retraining whole model using ALSWR, the updating time grows linearly with the size of data. On the other hand, our proposed RI-SGD method only update the corresponding two latent factors, which is unrelated to the data size of initial latent factor model. The same phenomenon can also be observed in RISGD log and ISGD. In windowing method, if we set window size to 5% of the data, we not only use 5% of the data to train an initial matrix factorization model but 5% of the data to retrain the following model. Therefore, we can find out that the updating time of 10% of the data is twice the training time of using 10% of the data.

On average, ALSWR spends 90 seconds to retrain the whole model and windowing method spends about 38 seconds. We can find that methods with SVD updating mechanism cost less than 100 milliseconds to update model, which is about 0.1% of the updating time using ALSWR and 0.2% of the updating time using windowing method from Table I and II. The popular item method spends about 2 seconds every update. Even though the updating time of ALSWR and windowing method are acceptable in the experiment, it is unable to deal with larger system with larger user and item size in real-time applications. In contrast, our proposed method is able to handle real-time update problem.

2) Accuracy: The accuracy results are shown in Fig.9 and Fig.10. From the two figures, it is easy to see that model accuracy attenuates with time if no updating mechanism is applied. Our proposed method and ALSWR have a relatively stable and accurate result. Windowing method provides the best accuracy among all method because it rebuilds the model with latest user behavior. RI-SGD log do not have good accuracy since limiting the range of predicted value may eliminate the effect of items with large implicit value. ISGD [12], which ignore the numerical value of implicit feedback does not perform well, either. We also can find out that merely recommending popular artists is a bad strategy from the accuracy result in the figures.

Although adopting the latest data makes windowing method the best method in accuracy, it suffers from long training time and limit user problem. If the new user is not included in the training model, it is unable to make any recommendation based on the model. Our proposed method can have a competitive accuracy(about 12%) over other mechanisms and retain extremely low updating time.

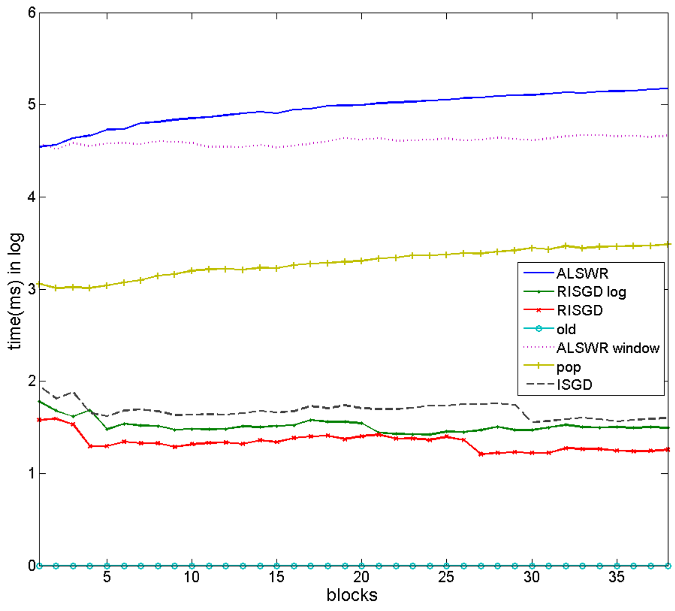


Fig. 7. The updating time of using 5% of data in initial training model.

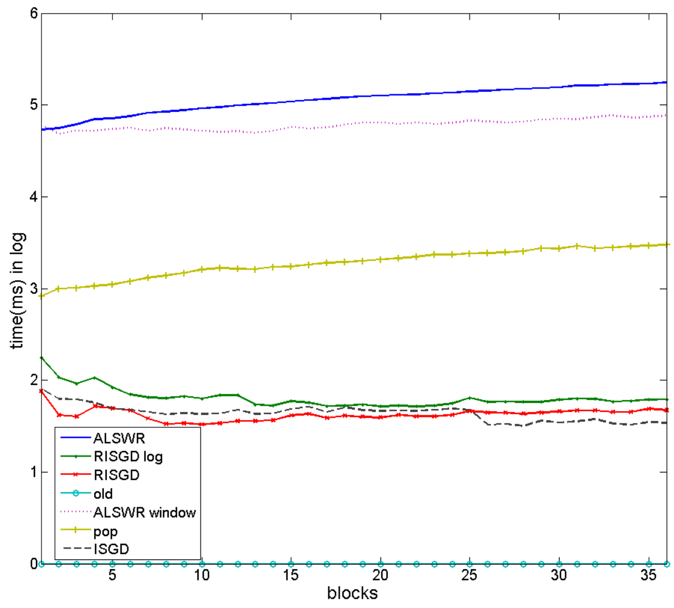


Fig. 8. The updating time of using 10% of data in initial training model.

TABLE I
UPDATING TIME OF USING 5% OF DATA IN INITIAL TRAINING MODEL.

method	avg. update time
No update	0
Proposed RI-SGD method	26.85ms
Proposed RI-SGD method with logistic	42.99ms
binary ISGD	48.83ms
Retrain method	90170.19ms
Windowing method	38040.24ms
Pop item method	1926.71ms

TABLE II
UPDATING TIME OF USING 10% OF DATA IN INITIAL TRAINING MODEL.

method	avg. update time
No update	0
Proposed RI-SGD method	29.74ms
Proposed RI-SGD method with logistic	47.92ms
binary ISGD	41.37ms
Retrain method	120583.20ms
Windowing method	75077.52ms
Pop item method	2578.28ms

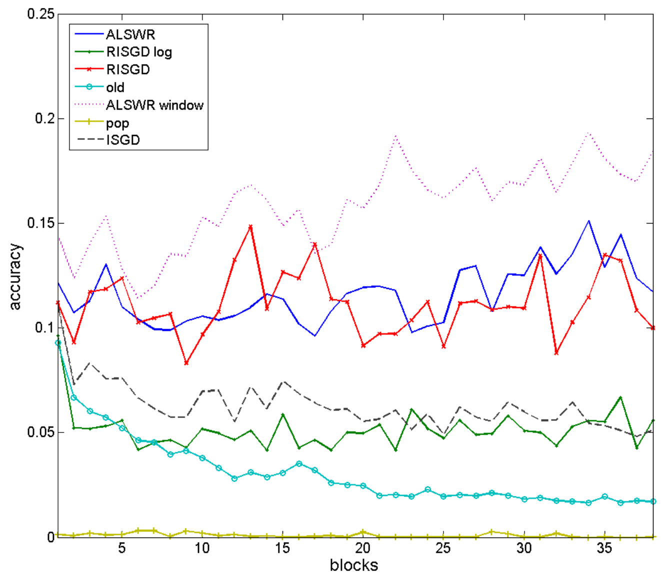


Fig. 9. The accuracy of using 5% of data in initial training model.

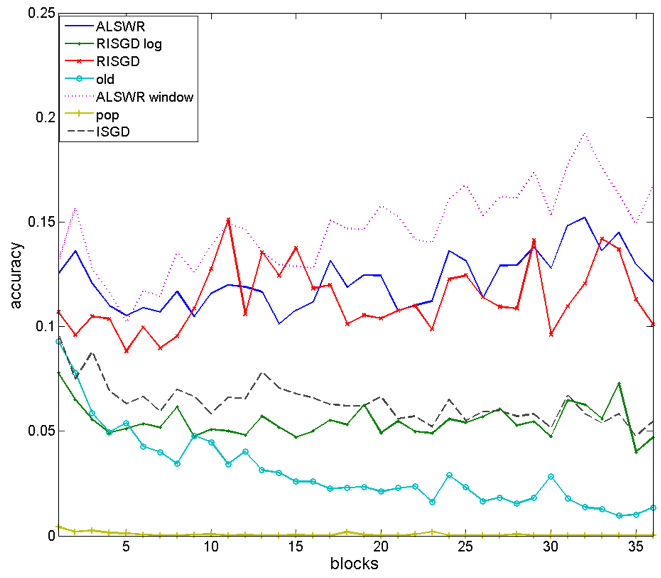


Fig. 10. The accuracy of using 10% of data in initial training model.

D. Discussion

Since the data percentage used in the initial models is considered small (5% and 10%) relative to the rest of the data, it is not surprising to see that the accuracy results seem similar no matter how much data we use in the initial models. However, the experiment of using more percentage of data as initial model should be conducted in the future.

In the experiment, we use music dataset to evaluate the performance of our proposed algorithm. However, our algorithm can be applied to all kind of datasets with the property of implicit feedback: the accumulating characteristic and large-range numerical value.

Although our algorithm is providing enough accuracy with less updating time, there are several methods for us to improve our algorithm in the future. First, since the effect of large record values will dominate gradient descent process, we will use forgetting factor to reduce the impact of old records. Second, we will investigate the effect of user information on the performance of prediction models. If enough information is provided, how we can use them to improve the accuracy becomes an important issue. Finally, we will try to implement the on-line updating version of non-negative matrix factorization. Since it considers the non-negative characteristic of implicit feedback, we believe NMF can obtain a better accuracy than original matrix factorization.

V. Conclusion

In this paper, we investigated how the uninterrupted streaming data with time affected the performance of model-based recommendation algorithm in a dynamic system and discussed the accumulating characteristic of implicit feedback and the deficiency of traditional model-based method in streaming data environment.

We proposed a robust real-time incremental updating algorithm using SGD to overcome the time wasting real-time updating problem for solving latent factors in matrix factorization. We implemented the proposed algorithm on IBM Infoshpere Streams Platform to compare with retraining, windowing-based using ALSWR and other methods. The experiment result showed that the accuracy of our proposed method and retraining with ALSWR was almost the same but we spend less only than 1% of the retraining time of ALSWR to update the model. All in all, the proposed method not only dealt with implicit feedback but also updated model in a short time and keep high accuracy.

References

[1] G. Adomavicius and A. Tuzhilin, "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.*, vol. 17, no. 6, pp. 734–749, 2005.

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

[3] J. Bennett, C. Elkan, B. Liu, P. Smyth, and D. Tikk, "Kdd cup and workshop 2007," *ACM SIGKDD Explorations Newsletter*, 2007.

[4] Y. Hu, Y. Koren, and C. Volinsky, "Collaborative filtering for implicit feedback datasets," 2008.

[5] G. Ling, H. Yang, I. King, and M. Lyu, "Online learning for collaborative filtering," in *The International Joint Conference on Neural Networks (IJCNN)*, 2012.

[6] J. Vinagre, A. M. Jorge, and J. Gama, "Fast incremental matrix factorization for recommendation with positive-only feedback," in *User Modeling, Adaptation, and Personalization*, Springer, 2014.

[7] C. Ballard, D. M. Farrell, M. Lee, P. D. Stone, S. Thibault, S. Tucker, *et al.*, *IBM InfoSphere Streams Harnessing Data in Motion*. IBM Redbooks, 2010.

[8] J. Vinagre, A. M. Jorge, and J. Gama, "Evaluation of recommender systems in streaming environments," in *User Modeling, Adaptation, and Personalization*, Springer, 2014.

[9] Ò. Celma, *Music Recommendation and Discovery in the Long Tail.* PhD thesis, Universitat Pompeu Fabra, 2008.

[10] Last.fm, "Last.fm api." http://cn.last.fm/api.

[11] Apache Software Foundation, "Apache mahout:: Scalable machine-learning and data-mining library." http://mahout.apache.org.

[12] S. Rendle and L. Schmidt-Thieme, "Online-updating regularized kernel matrix factorization models for large-scale recommender systems," in *Proceedings of the ACM Conference on Recommender Systems*, 2008.