Simultaneous Selection Method of Query Items and Neighbors in a Recommender System

Koki Miura, Mitsuru Takeuchi, and Yoshifumi Okada

***Abstract*—User-based collaborative filtering (CF) is the most popular and basic recommendation approach. It selects *k*-nearest neighbors with similar preferences to an active user and recommends items that are rated highly by neighbors of the active user. In user-based CF, it is important to employ better query items (input items to user-based CF) and neighbors to provide high recommendation accuracy. We propose a simultaneous selection method for query items and *k*-nearest neighbors for each active user. The proposed method identifies adequate query items and *k*-nearest neighbors by repeatedly updating selection of query items and neighbors. In this study, recommendation accuracy in each update is evaluated using two benchmark datasets for recommender systems. Experimental results show that the proposed method enables more users to obtain items that are suited to their preferences compared to typical user-based CF.**

***Index Terms*—Recommender system, user-based collaborative filtering, item selection, neighbor selection**

I. Introduction

Recommender systems have enabled us to obtain a wide variety of products or information (hereafter item). To date, many recommendation algorithms have been proposed [1]-[7]. In terms of practical application, collaborative filtering (CF) is the most successful approach [8]-[12]. In particular, user-based CF is very popular owing to its effectiveness and easy implementation [13]. It selects *k*-nearest neighbors (hereafter neighbors) to the active user (i.e., the user asking for recommendations) based on preference similarities. This process is referred to as neighbor selection. Then, items that are rated highly by the selected neighbors are recommended to the active user. In neighbor selection, preference similarity between the active user and the other users is computed using the rating data of all items (hereafter query items) preferred by the active user. This is based on the assumption that there exist neighbors with ratings that are similar to those of the active user for all query items. However, the above assumption is not always true, because the number of items in a database is generally very large.

We consider that it is important to employ better query items and neighbors to provide high recommendation accuracy. In this study, we propose a method that enables simultaneous selection of adequate query items and neighbors. The proposed method is based on an algorithm that repeatedly updates the following selection processes.

- Selection of query items that are useful for discriminating between neighbors and non-neighbors
- Selection of neighbors by those query items

Recommendation for each active user is performed using the query items and neighbors obtained by the above repetitive updates. In this study, the proposed method is applied to two benchmark datasets for a recommender system, and its performance is evaluated for each update.

Manuscript received January 5, 2016.

K. Miura is with Division of Production and Information Systems Engineering, Muroran Institute of Technology, 27-1, Mizumoto-cho, Muroran, Hokkaido 050-8585, Japan (e-mail: miura@cbrl.csse.muroran-it.ac.jp).

M. Takeuchi is with Division of Production and Information Systems Engineering, Muroran Institute of Technology, 27-1, Mizumoto-cho, Muroran, Hokkaido 050-8585, Japan (e-mail: takeuchi@cbrl.csse.muroran-it.ac.jp).

Y. Okada is with College of Information and Systems, Muroran Institute of Technology, 27-1, Mizumoto-cho, Muroran, Hokkaido 050-8585, Japan (corresponding author; telephone: +81-143-5408; fax: +81-143-5408; e-mail: okada@csse.muroran-it.ac.jp)

II. Method

In the proposed method, neighbor selection is executed for each active user. Figure 1 shows the procedure of the proposed method. An input to the method is a rating matrix, in which each row is an item, each column is a user, and each element is rating data. The proposed method consists of the following four steps.

1) Classify into neighbors and non-neighbors
2) Select items that are useful for discriminating between neighbors and non-neighbors
3) Repeat Steps 1 and 2 until the termination conditions are satisfied
4) Output neighbors and items

A. Classification into neighbors and non-neighbors

The similarity between an active user and another user is measured by treating each user as a vector (hereafter rating vector) of the rating data and computing the cosine of the angle formed by the rating vectors. The cosine similarity between the active user *Au* and another user U_j is expressed as follows:

$$sim(Au, U_j) = \frac{\overrightarrow{Au} \cdot \overrightarrow{U_j}}{\left|\overrightarrow{Au}\right| \times \left|\overrightarrow{U_j}\right|}, \quad (1)$$

where · denotes the dot product of the two rating vectors. Note that the similarity is calculated using only items rated by both *Au* and U_j.

Subsequently, we sort the other users in descending order of similarity and calculate the median of the similarities. Neighbors and non-neighbors are discriminated by a

ISBN: 978-988-19253-8-1
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

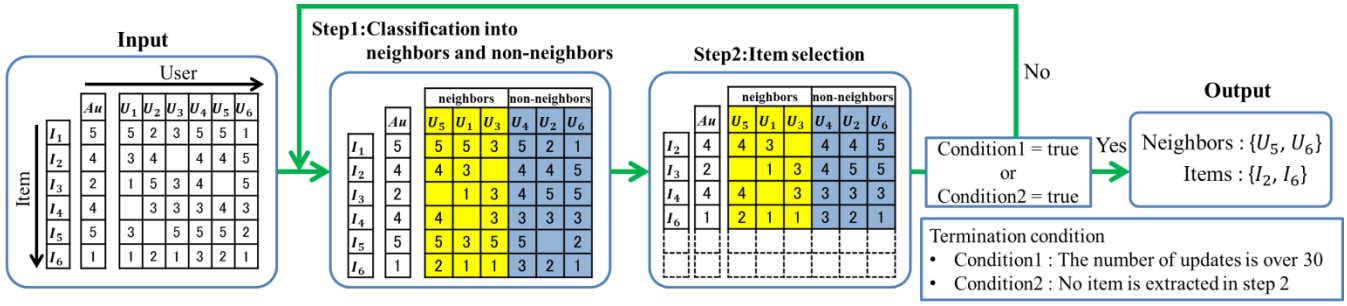


Fig. 1. Procedure of the proposed method

predefined threshold. In this study, the threshold is set to the greater of the median and $1/\sqrt{2}$.

B. Item selection

For each item, we calculate a correlation ratio between neighbors and non-neighbors. The correlation ratio is given as follows:

$$\eta_k^2 = \frac{Sb}{St}, \tag{2}$$

where η_k^2 is the correlation ratio for item k, St is the total variation, and Sb is the between-class variation. St and Sb are calculated as follows:

$$St = \sum_j (x_{jk} - \overline{x(k)})^2 \tag{3}$$

,

$$Sb = \sum_c \left((\overline{x_c(k)} - \overline{x(k)})^2 \cdot n_c(k) \right) \tag{4}$$

Here, c is a class label variable and takes 1 or 2, where 1 and 2 correspond to the neighbor user and non-neighbor user classes, respectively. x_{jk} is a rating value for item k of user i. $\overline{x_c(k)}$ is the mean value of ratings for item k in class c, and $\overline{x(k)}$ is the mean value of ratings for item k over the two classes. $n_c(k)$ is the number of users who rate item k in the two classes.

Subsequently, we compute the $F0$ value to estimate the difference between neighbors and non-neighbors. The $F0$ value for item k is calculated as follows:

$$F0 = \frac{(n-p)\eta_k^2}{(p-1)(1-\eta_k^2)}, \tag{5}$$

where n is the number of users rating item k, and p indicates the number of classes. In this study, an item with an $F0$ value of not less than 2.0 is considered a discriminative item between neighbors and non-neighbors.

C. Item/neighbor update and termination conditions

The above steps are repeated until the predefined termination conditions are satisfied. Then, the remaining items and neighbors are output. The termination conditions are as follows:

1) The number of repetitions exceeds 30;
2) There is no item to be selected (Section IIB).

III. Evaluation Experiments

A. Datasets

In this study, we used two datasets, MovieLens and Jester [14], which are benchmark datasets for recommender systems. MovieLens consists of 100,000 rating data (integer values of 1–5) from 900 users for 1682 movies. Jester consists of 300,000 rating data (real values in the range from −10.0 to +10.0) from 3,000 users for 100 jokes.

B. Experimental method

In this experiment, we investigated recommendation accuracies using items and neighbors obtained in each update. The experiment was conducted by 3-fold cross-validation. Each dataset was transformed into the rating matrix in advance. The procedure of the experimental method is described in the following.

First, for each dataset, the rating matrix was divided into three subsets with regard to users, where one subset was a test dataset, and the remaining two subsets were a training dataset. The test dataset was used as a set of active users, and the training dataset was used as a set of other users. For every active user, we divided the rated items into query and evaluation items. The query items were used for item updates, and the evaluation items were used to calculate recommendation accuracy.

Next, the recommendation accuracy in the respective updates for each active user was calculated by the following procedure. The neighbors of the active user were identified using query items (Section II). The score of item k (except for query items) was calculated as follows [13]:

$$score(k) = \overline{Au} + \frac{\sum_{Nu_j \in Z} \left(sim(Au, Nu_j) \times (Nu_{jk} - \overline{Nu_j}) \right)}{\sum_{Nu_j \in Z} |sim(Au, Nu_j)|}, \tag{6}$$

where z is a set of neighbors rating item k. $\overline{Au}$ and $\overline{Nu_j}$ are the mean ratings for the active user and a neighbor j, respectively, and Nu_{jk} is a rating value for item k by neighbor j. Next, X items were recommended in descending order of the above scores. In this experiment, we set X = 100 for MovieLens and X = 20 for Jester. Subsequently, we calculated the recommendation accuracy using three indexes, *precision*, *recall*, and *F1-measure* [15]. *Precision* is a ratio of items that were rated highly by the active user out of the total recommended items. *Recall* is a ratio of items that were rated highly by the active user out of the total highly rated evaluation items. *F1-measure* is the harmonic mean of *precision* and *recall*. These indexes all take values in the range of 0–1.

ISBN: 978-988-19253-8-1
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

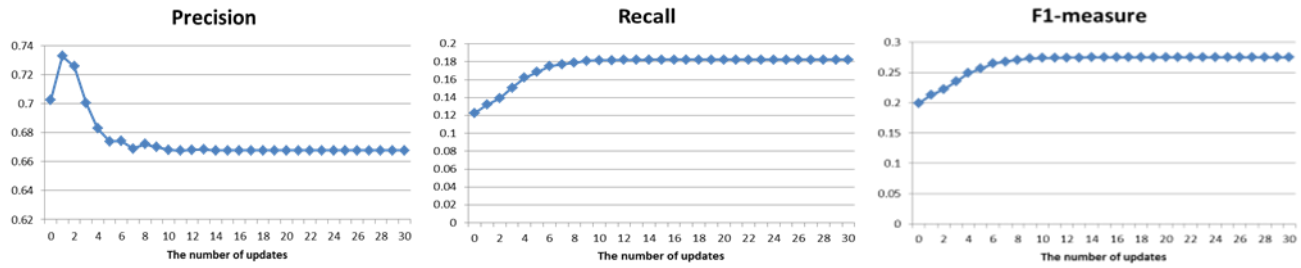


Fig. 2. Recommendation accuracy of MovieLens

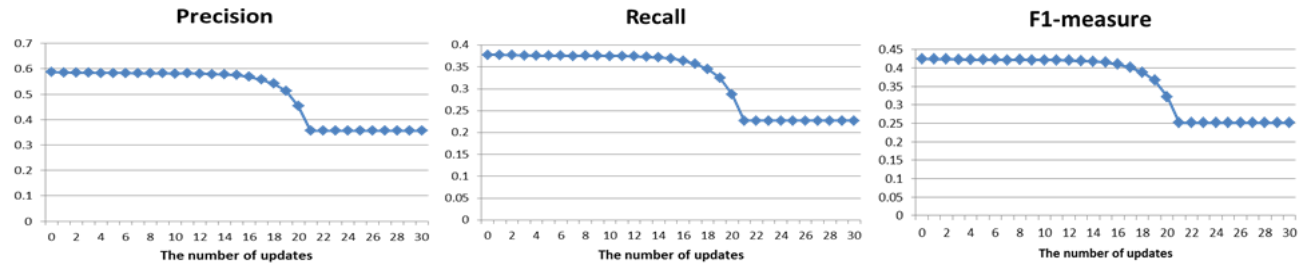


Fig. 3. Recommendation accuracy of Jester

The above operations were repeated three times by substituting the training dataset for the test dataset. Finally, for the three indexes, we calculated the mean values over all active users.

IV. Results and Discussion

Here, we discuss the results for the MovieLens and Jester datasets.

A. *Recommendation accuracy*

Figure 2 shows the *precision*, *recall*, and *F1-measure* results for MovieLens. The horizontal axis is the number of updates, and the vertical axis is the score of each index. Note that the score in each step shows the mean value of the scores for all users. When the number of item updates is 0, it indicates a result of the existing user-based CF that utilizes all query items for similarity calculations. The *precision* score shows an increase of approximately 3% by the first update and subsequently becomes constant via a rapid decrease. The *recall* score becomes constant after an increase of approximately 6% until the fourteenth update. *F1-measure* shows a result that is similar to *recall*. This indicates that this update method is effective for improving recommendation accuracy.

Figure 3 shows the *precision*, *recall*, and *F1-measure* results for Jester. Similar to the results for MovieLens, these graphs also show the recommendation accuracy for each update. As can be seen, all indexes show a similar tendency; the scores decrease drastically from approximately the eighteenth update after maintaining stable scores. This is due to fact that recommendation items are not scored adequately because the number of neighbors decreases significantly as the number of updates is increased.

B. *Number of active users who are recommended highly rated items*

Figure 4 shows the number of active users for whom one or more highly rated evaluation items are recommended in each update.

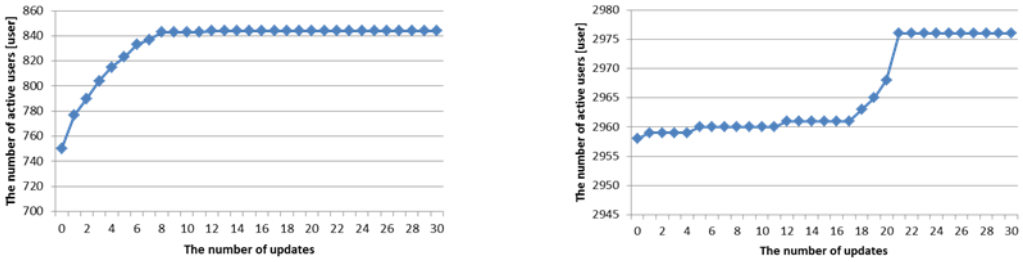
Figure 4(a) shows the results for MovieLens. As can be seen, the number of active users increases monotonically until the eleventh update. In addition, there is a difference of approximately 100 users between the no update result and the eleventh update result. This indicates that more users can obtain items that are suitable to their preferences by this update method.

Figure 4(b) shows the results for Jester. As can be seen, the number of active users increases gradually until the seventeenth update and subsequently shows a sharp increase followed by constant scores. This also indicates that more users can obtain items that are suitable to their preferences by this update method, although the rate of increase is substantially lesser than that of MovieLens.

V. Conclusion

In this study, we have proposed a method that enables simultaneous selection of adequate query items and neighbors for each active user. In an experiment, the proposed method was applied to two datasets, MovieLens and Jester, and the recommendation accuracy was evaluated for each update. The following two findings were obtained. First, recommendation accuracy depends on the types of datasets. And second, for both datasets, more users can obtain items that are suitable to their preferences by updating query items and neighbors. In future, we will introduce a technique that automatically determines a termination condition for each user and evaluate the recommendation accuracies of the improved method.

ISBN: 978-988-19253-8-1
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)



(a) MovieLens (b) Jester

Fig. 4. Number of active users who are recommended highly-rated items

REFERENCES

[1] D. Bridge, M. Göker, L. McGinty, and B. Smyth, "Case-based recommender systems," *Knowledge Engineering Review*, vol. 20. no. 3, Sep. 2005, pp. 315-320.

[2] R. Burke, P. Brusilovsky, A. Kobsa and W. Nejdl, "Hybrid web recommender systems," *The Adaptive Web: Methods and Strategies of Web Personalization*, Germany, Heidelberg: Springer, 2007, pp. 377-408.

[3] K. Goldberg, T. Roeder, D. Gupta, and C. Perkins, "Eigentaste: A constant time collaborative filtering algorithm," *Information Retrieval*, vol. 4, no. 2, 2001, pp. 133-151.

[4] M. J. Pazzani. "A framework for collaborative, content-based and demographic filtering," *Artificial Intelligence Review*, vol. 13. no. 5-6, 1999, pp. 393-408.

[5] C. Thompson, M. Göker, and P. Langley. "A personalized system for conversational recommendations," *Journal of Artificial Intelligence Research*, vol. 21, 2004, pp. 393–428.

[6] K.H.L.Tso-sutter, L.B.Marinho, and L.Schmidt-Thieme, "Tag-aware recommender systems by fusion of collaborative filtering algorithms," in 2008 *Proc. ACM Symposium on Applied Computing* Fortaleza, Ceara, Brazil, pp. 1995-1999.

[7] M. Zanker, "A collaborative constraint-based meta-level recommender," in 2008 *Proc. ACM Conference on Recommender Systems*, Lausanne, Switzerland, ACM Press, pp. 139-146.

[8] B. Sarwar, G. Karypis, J. A. Konstan, and J. Riedl, "Item-based collaborative filtering recommendation algorithms," in 2001 *Proc. 10th International Conference on World Wide Web*, Hong Kong, ACM, pp. 285-295.

[9] K. Yu, A. Schwaighofer, V. Tresp, X. Xu, and H.-P.Kriegel, "Probabilistic memory-based collaborative filtering," *IEEE Transactions on Knowledge and Data Engineering*, vol. 16, no. 1, 2004, pp. 56-69.

[10] G. Linden, B. Smith, and J. York, "Amazon.com recommendations: item-to-item collaborative filtering," *Internet Computing*, IEEE Volume 7, no. 1, 2003, pp. 76-80.

[11] J. Herlocker, J. A. Konstan, and J. Riedl, "An empirical analysis of design choices in neighborhood-based collaborative filtering algorithms," *Information Retrieval*, vol. 5, no. 4, 2002, pp. 287-310.

[12] J. B. Schafer, D. Frankowski, J. Herlocker, and S. Sen, "Collaborative filtering recommender system," *The Adaptive Web*, Lecture Notes in Computer Science, vol. 4321, 2007, pp. 291-324.

[13] X. Su, Taghi M. Khoshgogtaar, "A Survey of Collaborative Filtering Techniques," *Advances in Artificial Intelligence*, vol. 2009, 2009.

[14] GroupLens_Research: http://www.grouplens.org

[15] X. Wang, T. Nagashima, K. Fukuta, Y. Okada, M. Sawai, H. Tanaka And T. Uozumi, 2010. "Statistical method for classifying cries of baby based on pattern recognition of power spectrum," *International Journal of Biometrics*, vol. 2, no. 2, 2010, pp. 113-123.

ISBN: 978-988-19253-8-1
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)