Regular Paper

Improvement of Collaborative Filtering with the Simple Bayesian Classifier

Koji Miyahara† and Michael J. Pazzani††

Collaborative-filtering-enabled Web sites that recommend books, CDs, movies, and so on, have become very popular on the Internet. Such sites recommend items to a user on the basis of the opinions of other users with similar tastes. In this paper, we discuss an approach to collaborative filtering based on the Simple Bayesian Classifier, and apply our model to two variants of the collaborative filtering. One is user-based collaborative filtering, which makes predictions based on the users' similarities. The other is item-based collaborative filtering, which makes predictions based on the items' similarities. In our approach, the similarity between users or items is calculated from negative ratings and positive ratings separately. To evaluate our algorithms, we used a database of movie recommendations. Our empirical results show that our proposed Bayesian approaches outperform typical correlation-based collaborative filtering algorithms. We also discuss an approach that combines user-based and item-based collaborative filtering with the Simple Bayesian Classifier to improve the performance of the predictions. After the user-item rating matrix has been filled out with pseudo-scores generated by the item-based filter, the user-based recommendation is applied to the matrix. We show that the combined method performs better than the single collaborative recommendation method.

1. Introduction

The growth of the Internet has resulted in the availability of a tremendous amount of information and a vast array of choices for consumers. Recommender systems are designed to help a user cope with this situation by selecting a small number of options to present to the user [17]. They filter and recommend items on the basis of a user preference model. Among the various types of recommender systems that have been proposed, their filtering techniques fall into two categories: content-based filtering [15] and collaborative filtering or social filtering [20].

In content-based filtering, a user preference model is constructed for the individual on the basis of the user's ratings and descriptions (usually, textual expression) of the rated items. Such systems try to find regularities in the descriptions that can be used to distinguish highly rated items from others. On the other hand, collaborative filtering tries to find desired items on the basis of the preferences of a set of similar users. In order to find like-minded users, it compares other users' ratings with the target user's ratings. Since it is not necessary to analyze the contents of items, collaborative filtering can be applied to many kinds of domains where a textual description is not available or where regularities in the words used in the textual description are not informative [5].

In this paper, we focus on collaborative filtering techniques. A variety of algorithms have been reported and evaluated empirically [2],[17],[20]. Some of the most popular algorithms in collaborative filtering use a correlation-based approach. We report experimental results comparing collaborating filtering with the Simple Bayesian Classifier as an alternative approach. We also report the experimental results of an approach that combines user-based collaborative recommendation, which is based on users' similarities, and item-based collaborative recommendation, which is based on items' similarities.

This paper is organized as follows. We explain our problem space and outline related work, including the central ideas of a current typical collaborative filtering algorithm. Next, we define a formulation of the Simple Bayesian Classifier for collaborative filtering. The proposed model is applied to user-based collaborative recommendation and item-based collaborative recommendation tasks. We then evaluate our algorithms on a database of user ratings for movies, and show that our approach outperforms a typical correlation-based collaborative

† Information Technology R&D Center, Mitsubishi Electric Corporation
†† Department of Information and Computer Science, University of California, Irvine

Table 1 Example of a user-item rating matrix.

User \ Item	I_1	I_2	I_3	I_4	I_5
U_1	*Like*	*Dislike*	*Dislike*		*Like*
U_2	*Dislike*			*Like*	*Dislike*
U_3		*Like*	*Dislike*		*Like*
Class Label	*Like*	*Like*	*Like*	*Dislike*	*?*

filtering algorithm. Then, we propose an approach that combines the item-based collaborative filter with the Simple Bayesian Classifier and the user-based collaborative filter with the Simple Bayesian Classifier to improve the performance, and show that the combined method performs better than the single collaborative recommendation method.

2. Problem Space

The problem of collaborative filtering is to predict how well a user will like an item that he/she has not rated. This problem space can be expressed as a user-item rating matrix. Each entry of the rating matrix shows a user's rating for a specific item.

Table 1 is an example of a rating matrix, where rows correspond to users, columns correspond to items, and the matrix entries are ratings. Typically, the rating matrix is *sparse*. *Sparse* in this context means that some entries in the matrix are empty, because each user typically rates only a very small subset of all possible items. The last row represents the ratings of a user for whom the system will make predictions. The prediction task can be seen as filling the target user's empty elements of the matrix.

Most collaborating filtering systems adopt numerical ratings and try to predict the exact numerical ratings. Generally, they predict user ratings on a continuous scale. In contrast with these systems, we will treat collaborative filtering as a classification task that classifies unseen items into two or more separate classes. By treating it as a classification task, we expect that we can apply well-founded classification algorithms. In addition, we are not interested in the prediction of the exact rating a user would have given to a target item. We would much rather have a system that can accurately distinguish between recommendable items and others. Therefore, we defined two classes, *Like* and *Dislike*, that were used as class labels. Our problem space is to predict a class label ("*Like*" or "*Dislike*") on I_5 in Table 1.

3. Related Work

The main idea of collaborative filtering is to recommend new items of interest for a particular user on the basis of other users' opinions. A variety of collaborative filtering algorithms have been reported and their performance has been evaluated empirically [2),18),20)]. These algorithms are based on a simple intuition: predictions for a user should be based on the preference patterns of other people who have similar interests. Therefore, the first step of these algorithms is to find similarities between user ratings. Suppose we have a database of user ratings for items, where users indicate their interest in an item on a numeric scale. Resnick et al. use the Pearson correlation coefficient as a measure of preference similarity [18)]. The correlation between users j and k is

$$w_{jk} = \frac{\sum_i (R_{ij} - \overline{R_j})(R_{ik} - \overline{R_k})}{\sqrt{\sum_i (R_{ij} - \overline{R_j})^2 \cdot \sum_i (R_{ik} - \overline{R_k})^2}},$$

where $R_{ij}(R_{ik})$ is the rating of user $j(k)$ for item i, $\overline{R_j}(\overline{R_k})$ is the mean rating value of user $j(k)$, and all summations over i are over the items that have been rated by both j and k. The predicted rating of user j for item i is computed as a weighted sum of other users' ratings:

$$\hat{R}_{ij} = \overline{R_j} + \sum_k (R_{ik} - \overline{R_k}) w_{jk} \Big/ \sum_k |w_{jk}|.$$

If no ratings for item i are available, the predicted value is equivalent to the mean value of ratings by user j.

This correlation-based prediction scheme has been shown to perform well. However, it might be valuable to think of other approaches. Breese, et al. [2)] report a variety of modifications to the above typical collaborative filtering technique and the use of Bayesian clustering and a Bayesian network. A primary difference between what we propose below and the work of Breese, et al. is that we construct a separate Bayesian model for each user. This is practical only for the Simple Bayesian Classifier, which is linear in the number of examples and the number of features.

One of the problems of collaborative filtering is the sparsity of a typical rating matrix. Since most users generally do not rate most items,

similarities between users calculated from a sparse rating matrix may be inaccu rate. As a result, the accuracy of predictions may be poor. To tackle this problem, many techniques have been proposed. Billsus and Pazzani [1], and Sarwar, et al. [19] have proposed dimensionality reduction of a rating matrix with Singular Value Decomposition (SVD) to capture the similarity in a reduced dimensional matrix. Melville, et al. [12] use content-based filtering to fill out un-rated entries in a sparse rating matrix. Our combination approach, which we propose below, is different from the above approaches. We do not convert the rating matrix, nor do we use contents descriptions.

4. Simple Bayesian Model

4.1 Simple Bayesian Classifier

We will now propose an application of the Simple Bayesian Classifier to the collaborative filtering problem space. The Simple Bayesian Classifier is one of the most successful machine learning algorithms in many classification domains. Despite its simplicity, it is shown to be competitive with other complex approaches, especially in text categorization tasks [10]. Making the "naive" assumption that features are independent given the class label, the probability of an item belonging to class j given its n feature values, $p(class_j|f_1, f_2, \ldots f_n)$ is proportional to:

$$p(class_j)\prod_i^n p(f_i|class_j),$$

where both $p(class_j)$ and $p(f_i|class_j)$ can be estimated from training data. To determine the most likely class of an example, the probability of each class is computed, and the example is assigned to the class with the highest probability. Although the assumption that features are independent once we know the class label of an item is not realistic in this domain, the Simple Bayesian Classifier has been shown to be optimal in many situations where this assumptions does not hold [3] and has been empirically shown to be competitive with more complex approaches in many others [10]. Moreover, the Simple Bayesian Classifier is fast because its learning time is linear in the number of examples in the training data.

Here, we define the Simple Bayesian Model for collaborative filtering. In our model, other users correspond to features and the matrix entries correspond to feature values. To determine the most likely class of the target item, the following formula is calculated:

$$\begin{aligned} Class = & \mathop{\arg\max}_{class_j \in \{Like, Dislike\}} \\ & p(class_j|U_1 = Like, U_3 = Dislike, \\ & \ldots, U_n = Like) \\ = & \mathop{\arg\max}_{class_j \in \{Like, Dislike\}} \\ & p(U_1 = Like, U_3 = Dislike, \ldots, \\ & U_n = Like|class_j)p(class_j) \\ = & \mathop{\arg\max}_{class_j \in \{Like, Dislike\}} \\ & p(class_j)\prod_i p(U_i = class_k|class_j). \end{aligned}$$

Moreover, we make use only of the data that both users rated when estimating conditional probabilities. In this representation, the following condition holds:

$$p(U_i = Like|class_j) + p(U_i = Dislike|class_j) = 1.$$

For example, in the rating matrix shown in Table 1, the estimated conditional probability of $p(U_1 = Like|Like)$ is *0.33*. However, this direct calculation may distort the probability when the number of commonly rated items is very small. We use a Laplacian prior in the actual calculation of conditional probabilities to smooth the probability estimates with few ratings and to avoid estimating a probability to be 0. Therefore, the value of $p(U_1 = Like|Like)$ is $(1 + 1)/(3 + 2) = 0.4$ in our model. By using the Simple Bayesian Classifier to make predictions, we expect to avoid a problem with typical correlation-based collaborative filtering algorithms. The correlation-based algorithms make a global model for similarity between users, rather than separate models for classes of ratings (e.g., positive rating vs. negative rating). For example, it might happen that a set of one user's positive ratings is a good predictor for other users' positive ratings but the negative ratings of one user may not be a good predictor for other users' negative ratings. Since the proposed model treats each class of ratings separately, we expect that the Bayesian model will capture predictiveness between users more precisely.

4.2 Item-Based Collaborative Filtering

So far, we have discussed a collaborative filtering technique based on users' similarities. This technique is called *User-Based Collaborative Filtering*. In a user-item rating matrix like Table 1, calculation of users' similarity is

equivalent to calculation of the similarity between rows. It is also possible to make predictions based on the similarity between columns, that is, the similarity between items. This technique is called *Item-Based Collaborative Filtering*. The main idea of item-based collaborative filtering is that the user will like an item that is similar to items he/she liked earlier, and will dislike an item that is similar to items he/she disliked earlier.

In our model's item-based collaborative filtering, items correspond to features and the ratings of the target user correspond to feature values. Therefore, the following formula will be calculated to determine the class label to which the target item belongs:

$$Class = \arg\max_{class_j \in \{Like, Dislike\}} p(class_j | I_1 = Like, I_3 = Like, \ldots, I_n = Dislike),$$

where I_i is an item that the target user has already rated.

Conditional probabilities, like $p(I_i = Like | class_j)$, are calculated by looking into the ratings of the target item, which is now going to be predicted, and the item I_i, which the target user has already rated. We make only use of the commonly rated pairs obtained from different users.

5. Experiments

5.1 Dataset

We used experimental data from the EachMovie collaborative filtering service. The EachMovie service was part of a research project at the DEC Systems Research Center [11]. The service was available for an 18-month period until it was shut down in September 1997. During that time 72,916 users entered numeric ratings for 1,628 movies. User ratings were recorded on a numeric six-point scale, ranging from 0 to $1(0, 0.2, 0.4, 0.6, 0.8, 1.0)$. In our experiments, we use an experimental protocol similar to the one first used in Billsus and Pazzani [1]. We restricted the number of users to the first 2,000 users in the database. These 2,000 users provided ratings for 1,366 different movies.

5.2 Evaluation Criteria

As we have described in Section 2, we are interested in discriminating between liked items and disliked items. To distinguish items, we transformed numerical ratings into two labels. In the EachMovie dataset, we labeled items as *Like* if the numerical rating for the item was 0.8 or 1.0, or *Dislike* otherwise. Not only does assigning class labels allow us to measure classification accuracy, but we can also apply additional performance measures, *precision* and *recall*, commonly used for information retrieval tasks. Precision and recall are defined as follows:

$$Precision = \frac{Number\ of\ liked\ items\ assigned\ to\ \text{“}Like\text{”}\ class}{Number\ of\ items\ assigned\ to\ \text{“}Like\text{”}\ class}.$$

$$Recall = \frac{Number\ of\ liked\ items\ assigned\ to\ \text{“}Like\text{”}\ class}{Number\ of\ liked\ items}.$$

However, it might be easy to optimize either one separately. To avoid this problem, we use F-Measure [9], which combines *precision* and *recall*:

$$F\text{-}Measure = \frac{2 \cdot precision \cdot recall}{precision + recall}.$$

In our experiments, we measure the performance of the algorithms by using classification accuracy and F-Measure.

5.3 Experimental Methodology

In our first experiment, we evaluated the performance of the user-based collaborative filter with the Simple Bayesian Classifier. We also evaluated a typical correlation approach, which is described in Resnick, et al. [18]. We consider all correlations in the correlation-based approach; that is, we do not restrict correlation values to those above a certain threshold. Since the correlation-based approach will predict numerical ratings, we labeled ratings *Like* or *Dislike* according to whether they were above or below a threshold value of 0.7 (the midpoint between the two possible user ratings *0.6* and *0.8*).

In our second experiment, we evaluated the performance of the item-based collaborative filter with the Simple Bayesian Classifier. As in the first experiment, we also evaluated a correlation-based approach. Instead of a correlation between users, we calculated a correlation between items in the correlation-based approach.

In both experiments, we report learning curves where we vary the number of rated items in training data. We randomly selected 20 test users who had rated at least 80 movies each. For each test user, we ran a total of 20 paired trials for each algorithm. For each trial, we randomly selected 50 rated items as a training set and 30 rated items as a test set. Then, we started training with 10 rated items out of

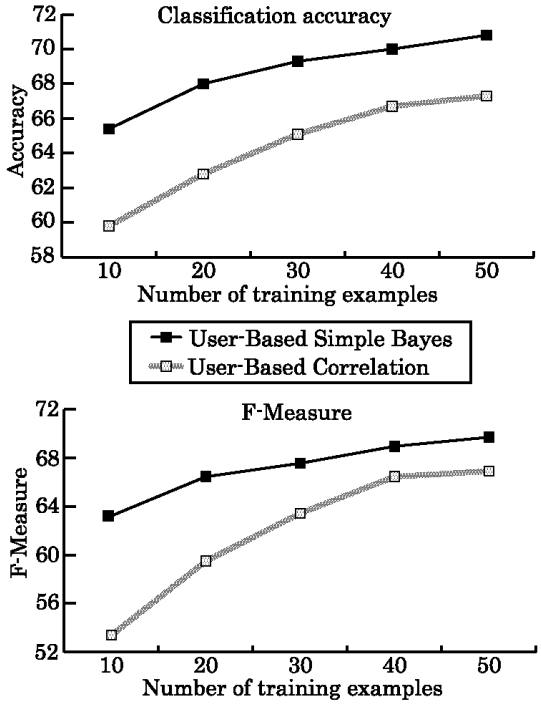

Fig. 1 Learning curves of user-based collaborative filtering.

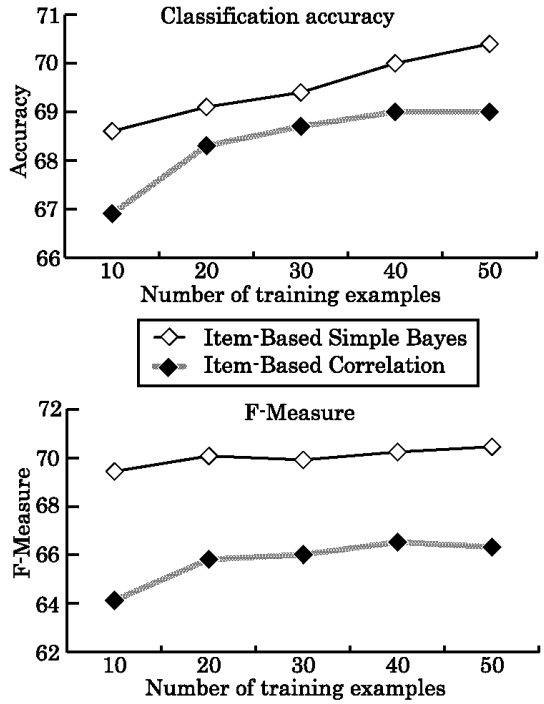

Fig. 2 Learning curves of item-based collaborative filtering.

50 training examples and increased the training examples incrementally in steps of 10 up to 50 items, measuring the algorithms' performance on the test examples. We repeated this for all test users and the final results reported here are averaged over 20 test users. Note that we used the same test users, the same training examples, and the same test examples in both experiments.

5.4 Results and Discussion

Figure 1 shows the learning curves of two different user-based collaborative filtering algorithms in our first experiment with the EachMovie dataset. The algorithm labeled *User-Based Simple Bayes* is user-based collaborative filtering with the Simple Bayesian Classifier, and the algorithm labeled *User-Based Correlation* is user-based collaborative filtering with correlation. These results show that the Simple Bayes performs better than the correlation-based algorithm. For 50 training examples, *Simple Bayes* reaches a classification accuracy of 70.8%, while *Correlation* is 67.3%. The F-Measure of *Simple Bayes* is 69.7%, while that of *Correlation* is 66.9%.

Figure 2 shows the learning curves of two different item-based algorithms in our second experiment. The algorithm labeled *Item-Based Simple Bayes* is item-based collaborative filtering with the Simple Bayes, and the algorithm labeled *Item-Based Correlation* is user-based collaborative filtering with correlation. As in the first experiment, the Simple Bayes outperforms the correlation-based algorithm. With 50 training examples, the classification accuracy is 70.4% for *Simple Bayes* and 69.0% for *Correlation*, while the F-Measure is 70.5% for *Simple Bayes* and 66.3% for *Correlation*.

Our experimental results show that the proposed collaborative recommendation with the Simple Bayesian Model significantly outperforms a typical correlation-based algorithm. We think that calculating the probability by separating positive ratings and negative ratings captures a more precise similarity between users or items, and leads to better predictions, as we described in Section 4.1. We also think that probability smoothing by means of a Laplacian prior in the Simple Bayesian Model might be effective, especially when the number of items rated in common between users is small.

Although item-based collaborative filters and user-based collaborative filters show similar performance for larger training examples, item-based filters significantly outperform user-based filters for 10 or 20 training examples. With

10 training examples, *User-Based Simple Bayes* reaches a classification accuracy of 65.4%. On the other hand, *Item-Based Simple Bayes* is 68.6% accurate. The F-Measure of *User-Based Simple Bayes* is 63.2%, whereas that of *Item-Based Simple Bayes* is 69.4%.

In user-based collaborative filters, the number of training examples has a strong impact on calculation of the similarity between users. For example, with 10 training examples, the number of commonly rated items is 10 or less. It would be very difficult to get 10 items rated in common between users in a sparse rating matrix. This might lead to inaccurate similarity estimates. On the other hand, the number of training examples does not have any impact on the calculation of similarity between items in our second experiment on item-based collaborative filters. We believe that this is one reason why item-based collaborative filters perform better than user-based collaborative filters for smaller training examples.

Although we used a simple binary rating scheme ("*Like*" and "*Dislike*"), it might be possible to apply the Bayesian model to the numerical ratings (e.g., the n-point scale of ratings) by constructing a separate Bayesian model that corresponds to each of the numerical ratings. However, it may happen that some ratings rarely appear in the data. As a result, the model might be inaccurate and the performance might be poor. To avoid this problem, we can transform numerical ratings into a smaller number of classes, e.g., *Like* and *Dislike*, which we adopted in our experiments. Since the Bayesian model predicts probabilities of class membership, we can use these probabilities to prioritize unseen items. This characteristic will have a similar impact to that of more detailed numbered ratings, and will be useful in cases where there are many items to be recommended, even though our goal is to have a system that can accurately distinguish between recommendable items and others.

6. Improvement of the Performance

6.1 Combining a User-Based Filter and an Item-Based Filter

Although the proposed collaborative filters with Simple Bayesian Classifier outperform typical correlation-based approaches, we still have a problem of a sparse rating matrix. If we apply a collaborative recommendation to a fully rated matrix, it would be expected that we might get better results. On the other hand, it is expected that the performance will be improved by combining a user-based collaborative filter and an item-based collaborative filter, because each makes predictions from a different point of view.

Here, we will propose an approach that combines the item-based collaborative filter and the user-based collaborative filter. This approach consists of the following two steps. First, we fill out un-rated entries in a sparse rating matrix with pseudo-scores generated by the item-based collaborative filter with the Simple Bayes. This step gives a dense rating matrix. Next, we apply the user-based collaborative filter with the Simple Bayes to the dense matrix that is generated in the first step. Since the similarities between users will be calculated from a larger number of commonly rated items than in the case of a sparse matrix, we can capture the similarities more accurately. As a result, it is expected that the performance will be better.

By using the combination method, we also avoid a problem with collaborative filtering. In collaborative filtering, a predicted score is based on other users' similarities and other users' ratings for the target item. If one user has not rated the target item, his/her similarity to the target user is not considered for a prediction of the target item, even if his/her preference is very similar to the target user. On the other hand, the item-based collaborative filter generates other users' pseudo-scores for the target item based on item similarity in the combination approach. Therefore, the pseudo-scores of other users are available for making a prediction, if they have not rated the target item.

6.2 Neighborhood Selection

In collaborative filtering, it is usual to select a subset of like-minded users (the neighborhood) for making a prediction, instead of the entire set of users. It is known that the size of the neighborhood has an impact on the prediction quality [6].

Likewise, in many supervised learning algorithms, feature selection is a common preprocessing technique. By restricting the number of features, it might be expected that one could increase the accuracy of the learner by ignoring irrelevant features, or reduce the computation time [16].

We apply a feature selection method to find a set of the N most informative users. Since our goal is to discriminate between classes, we de-

fine "most informative" as being equivalent to providing the most information about an item's class membership. We use an information-theory-based approach to determine the N most informative users. This is accomplished by computing the expected information gain [16] that the feature value ("*Like*" or "*Dislike*") of a user U contributes to the classification of a set of labeled items S that have been rated by the target user:

$$E(U,S) = I(S) - [p(U = Like) \cdot I(S_U = Like) + p(U = Dislike) \cdot I(S_U = Dislike)],$$

where $p(U = Like)$ is the probability that user U likes an item, $S_U = Like(Dislike)$ is the subset of labeled items S for which $U = Like(Dislike)$ holds, and $I(x)$ is the entropy of a set of labeled items, defined as:

$$I(S) = \sum_{c \in classes} -p(S_c) \cdot \log_2(p(S_c)),$$

where S_c is the set of all items rated by the target user that belong to class c ($c = \{Like, Dislike\}$).

6.3 Experiment

First, we evaluated the effectiveness of neighborhood selection for the combination approach using the same data (test users, training examples, and test examples) as in the experiments described in Section 5.3. For each test user, we applied the item-based collaborative filter with the Simple Bayes to the dataset to fill out all entries in the rating matrix except those of the test user. After that, we applied the user-based collaborative filter with the Simple Bayes to the dense rating matrix. We report the classification accuracy when the size of neighborhood in the user-based filter was varied between 10 users and 1,999 users. Note that 1,999 users corresponds to no neighborhood selection. For each test user, we ran a total of 20 paired trials. For each trial, we used 50 items as a training set and 30 items as a test set. Then we started predictions, varying the size of neighborhood according to the method mentioned in the previous section. The final results reported here are averaged for 20 test users.

Next, we evaluated the performance of the combination approach, using a similar methodology to that described in Section 5.3, and the same data. When making a prediction by using the user-based collaborative filter, we made use only of the ratings obtained from the selected neighborhood. In this experiment, we set the neighborhood size to 30, because we obtained the best classification accuracy in the first experiment.

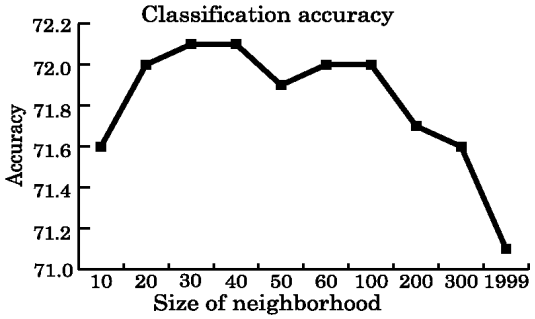

Fig. 3 Effectiveness of neighborhood selection in the combination approach.

6.4 Results and Discussion

Figure 3 shows the effectiveness of neighborhood selection. For neighborhood sizes of 30 and 40, the performance reaches the maximum accuracy of 72.1%. The accuracy decreases in proportion to the size of the neighborhood. For a neighborhood size of 1,999, which is the case in which we do not apply neighborhood selection, it reaches a classification accuracy of 71.1%. The results show that neighborhood selection is one way to improve the performance.

Figure 4 shows the learning curves of the combination approach in the second experiment described in the previous section. For reference, we show the learning curves of the user-based collaborative filter with the Simple Bayes and the item-based collaborative filter with the Simple Bayes. They are the results of the experiments shown in Fig. 1 and 2. The algorithm labeled *Item-User Combination* is the proposed combination collaborative filter with the Simple Bayesian Classifier. Note that we used 30 neighborhoods when we applied the user-based collaborative filter. The combination approach significantly outperforms the user-based collaborative filter and the item-based collaborative filter with the Simple Bayes. *Item-User Combination* reaches an accuracy of 72.1% with 50 training examples, while the accuracy of *Item-Based Simple Bayes* is 70.4% and that of *User-Based Simple Bayes* is 70.8%. The F-Measure of *Item-User Combination* reaches 72.6%, while that of *Item-Based Simple Bayes* is 70.5% and that of *User-Based Simple Bayes* is 69.7%. The results show that the combination method is effective for improving the performance. We think one reason for the improvement is that

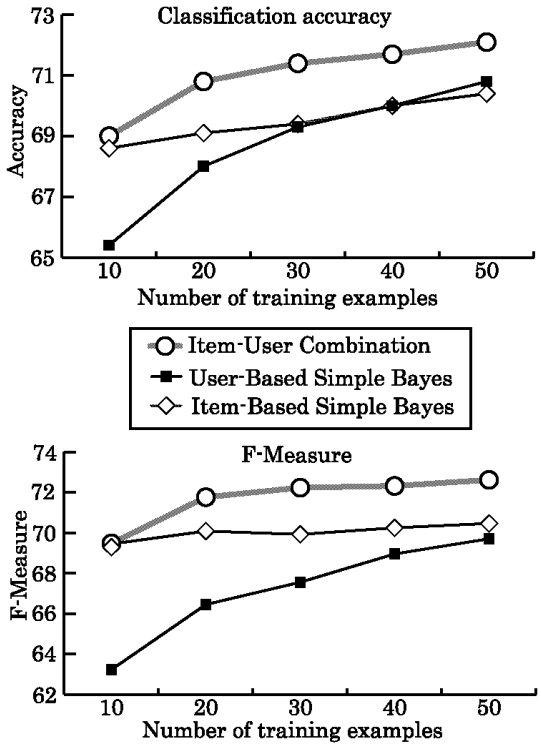

Fig. 4 Learning curves of item-user combination collaborative filtering.

the combination approach will capture similarities between users more accurately because the number of commonly rated items in a dense rating matrix will be larger than in the case of a sparse matrix.

With 10 training examples, the performance of the combination method is very similar to that of the item-based collaborative filter with the Simple Bayes. The classification accuracy of *Item-User Combination* is 69.0%, while that of *Item-Based Simple Bayes* is 68.6%. The F-Measure of *Item-User Combination* is 69.5%, while that of *Item-Based Simple Bayes* is 69.4%. This implies that 10 commonly rated items are insufficient to calculate the similarity between users. One way to avoid this problem will be to extend the number of items rated by the target user by making predictions with the item-based collaborative filter.

7. Conclusion

In this paper, we have reported on collaborative filtering with the Simple Bayesian Classifier. We proposed a user-based collaborative filter and an item-based collaborative filter with the Simple Bayesian Classifier. We found that the proposed methods perform better than typical correlation-based approaches. We also proposed a method that combines the user-based collaborative filter and the item-based collaborative filter, and showed that the combination method works well. Since the combination method uses the user-based collaborative filter and the item-based collaborative filter independently, we believe that improving the performance of the individual filter will lead to better performance of the whole system.

It is important to adopt other types of datasets to verify our methodology, because our experiments used only one dataset. We will also investigate a combination of content-based filtering and collaborative filtering. As a first step, we plan to integrate keyword features of items with a collaborative filtering framework, using the Simple Bayesian Classifier.

References

1) Billsus, D. and Pazzani, M.: Learning Collaborative Filters, *Proc. 15th International Conference on Machine Learning*, San Francisco, CA., Morgan Kaufmann Publishers (1998).
2) Breese, J., Heckerman, D. and Kadie, C.: Empirical Analysis of Predictive Algorithms for Collaborative Filtering, *Proc. 14th Conference on Uncertainty in Artificial Intelligence*, Madison, WI, Morgan Kaufmann Publisher (1998).
3) Domingos, P. and Pazzani, M.: On the Optimality of the Simple Bayesian Classifier under Zero-One Loss, *Machine Learning*, Vol.29, pp.103–130 (1997).
4) Good, N., Scafer, J., Konstan, J., Borchers, A., Sarwar, B., Herlocaker, J. and Riedl, J.: Combining Collaborative Filtering with Personal Agents for Better Recommendations, *Proc. AAAI*, pp.439–446, AAAI Press (1999).
5) Gupta, D., Digiovanni, M., Narita, H. and Goldberg, K.: Jester 2.0: A New Linear-Time Collaborative Filtering Algorithm Applied to Jokes, *Workshop on Recommender Systems Algorithms and Evaluation, 22nd International Conference on Research and Development in Information Retrieval*, Berkeley, CA (1999).
6) Herlocker, J., Konstan, J., Borchers, A. and Riedl, J.: An Algorithmic Framework for Performing Collaborative Filtering, *Proc. 22nd International Conference on Research and Development in Information Retrieval*, pp.230–237, Berkeley, CA, ACM Press (1999).
7) Hill, W., Stead, L., Rosenstein, M. and Furnas, G.: Recommending and Evaluating Choices in a Virtual Community of Use, *Proc. Conference on Human Factors in Computing*

Systems, pp.194–201, Denver, CO, ACM Press (1995).
8) Lewis, D.: Naive (Bayes) at Forty: The independence assumption in information retrieval, *Proc. 10th European Conference on Machine Learning* (1998).
9) Lewis, D. and Gale, W.A.: A Sequential Algorithm for Training Text Classifiers, *Proc. 17th International Conference on Research and Development in Information Retrieval*, pp.3–12, London, Springer-Verlag (1994).
10) McCallum, A. and Nigam, K.: A Comparison of Event Models for Naive Bayes Text Classification, *American Association for Artificial Intelligence* (*AAAI*) *Workshop on Learning for Text Categorization* (1998).
11) McJonese, P.: EachMovie Collaborative Filtering Data Set, DEC Systems Research Center (1997).
12) Melville, P., Mooney, R. and Nagarajan, R.: Content-Boosted Collaborative Filtering, *Workshop on Recommender Systems, 2001 International Conference on Research and Development in Information Retrieval*, New Orleans, LA (2001).
13) Mitchell, T.: *Machine Learning*, MacGraw-Hill, New York (1997).
14) Miyahara, K. and Pazzani, M.: Collaborative Filtering with the Simple Bayesian Classifier, *Proc. 6th Pacific Rim International Conference on Artificial Intelligence*, pp.679–689, Melbourne, Australia (2000).
15) Pazzani, M. and Billsus, D.: Learning and Revising User Profiles: The identification of interesting Web sites, *Machine Learning*, Vol.27, pp.313–331 (1997).
16) Quinlan, J.R.: Induction of Decision Trees, *Machine Learning*, Vol.1, pp.81–106 (1986).
17) Resnick, P. and Varian, H.: Recommender Systems, *Comm. ACM*, Vol.40, No.3, pp.56–58 (1997).
18) Resnick, P., Neophytos, I., Mitesh, S., Bergstrom, P. and Riedl, J.: GroupLens: An open architecture for collaborative filtering of netnews, *Proc. CSCW94: Conference on Computer Supported Cooperative Work*, pp.175–186, Chapel Hill, Addison-Wesley (1994).
19) Sarwar, B.M., Karypis, G., Konstan, J. and Riedl, J.: Application of Dimensionality Reduction in a Recommender System: A Case Study, *ACM WebKDD 2000 Web Mining for E-Commerce Workshop* (2000).
20) Shardanand, U. and Maes, P.: Social Information Filtering: Algorithms for automating "word of mouth", *Proc. Conference on Human Factors in Computing Systems*, pp.210–217, Denver, CO, ACM Press (1995).

(Received April 6, 2002)
(Accepted June 4, 2002)

Koji Miyahara received his B.E. and M.E. degrees in Information Systems Engineering from Kyushu University in 1986 and 1988, respectively. He joined Mitsubishi Electric Corp. in 1988 and currently works at the Information Technology R&D Center of Mitsubishi Electric Corp. He was a visiting researcher at the University of California, Irvine, from 1999 to 2000. His current research interests include user interface, intelligent agents and information filtering. He is a member of the IPSJ and IEEE.

Michael J. Pazzani is CEO of AdaptiveInfo, which builds personalization software for the wireless Web that automatically prioritizes information displayed for each individual user. Dr. Pazzani received his Ph.D. in Computer Science from UCLA in 1987 and is on leave from a full professor's position at the University of California, Irvine. Dr. Pazzani is the author of two books and over 100 technical papers on personalization, machine learning, and recommendation systems.