Chapter 14
Collaborative Filtering

Abstract Collaborative filtering is a relatively new technique to the database marketing field, gaining popularity with the advent of the Internet and the need for "recommendation engines." We discuss the two major forms of collaborative filtering: memory-based and model-based. The classic memory-based method is "nearest neighbor," where predictions of a target customer's preferences for a target product are based on customers who appear to have similar tastes to the target customer. A more recently used method is item-based collaborative filtering, which is model-based. In item-based collaborative filtering predictions of a target customer's preferences are based on whether customers who like the same products the target customer likes tend to like the target product. We discuss these and several other methods of collaborative filtering, as well as current issues and extensions.

14.1 Introduction

One day you rent a movie, "Independence Day," at a video rental store. You are surprised at the cash register to find that there are ten movie titles recommended on your receipt that happen to match your interests pretty well. An automatic collaborative filtering system in the store has a database storing the movie tastes of many other customers. It identifies customers in the database who like "Independence Day," finds out what other movies they liked, and recommends the ten most liked movie titles to you. It automatically recommends a set of movie titles as an expert, or more to the point, a *friend* would.

Collaborative filtering is a recently developed data mining technique. Its main concept originated from work in the area of information filtering and first introduced by Goldberg et al. (1992). Their email filtering system, called Tapestry, innovated the field of recommendation system even though it required users to evaluate items explicitly and respond to complex queries. Since then, the system has been improved to become automatic (Resnick

Table 14.1 Input data for collaborative filtering

	Movie 1	**Movie 2**	**Movie 3**	**Movie 4**	**Movie 5**
Amy	5	2	–	4	1
Joseph	1	–	1	2	–
Michael	–	4	3	–	5
Jim	3	1	–	1	2
Laura	5	3	4	–	1

et al. 1994) and the algorithm has been fine-tuned (Shardanand and Maes 1995). More recently, a number of websites including Amazon.com, CD-Now.com and MovieFinder.com have adopted collaborative filtering systems to provide personalized recommendations to their customers. Collaborative filtering is broadening its base of applications to include financial services, travel agencies, and so on.

Let us take an example of movie selection to understand the task applied to collaborative filtering. The objective of collaborative filtering is to select (and then recommend) a set of movies that each customer will like. The typical input data takes the form of preference ratings on each product/item evaluated by users. As shown on Table 14.1, it is the $n \times m$ user-item matrix (n users; m items) with each cell representing a user/consumer's preference rating on a specific item/product. Our main task is to predict the preference ratings for missing cells, based on other observed preference ratings. For example, Amy has rated Movies 1, 2, 4 and 5. Then what is Amy's predicted preference rating for Movie 3? Similarly, we wish to predict the missing preference ratings for all other customers. Once we have all the predicted movie ratings, we are ready to provide movie recommendations for each customer (e.g., suggest three highly rated movies for each customer).

14.2 Memory-Based Methods

There are a number of algorithms used in collaborative filtering, but they can be divided into two main categories, memory-based and model-based (Sarwar et al. 2001). Memory-based methods, also called neighborhood-based, user-based or heuristic methods, attempt to find a set of users that have similar preferences to the target user. Once a neighborhood of users is identified, memory-based methods combine their preferences to predict the preference of the target user. On the other hand, model-based methods first develop a model of user ratings. They usually take a probabilistic approach and calculate the expected preference for the target user. We first study memory-based methods in this section and describe model-based methods in the next section.

Memory-based methods predict the unobserved preferences of an active or target user based on the (observed) preferences of other users. Let $r_{i,j}$

be the preference rating of user i on item j (e.g., the value of cell in row i and column j in Table 14.1). The neighborhood-based method predicts the preference rating of an active user a for item j ($\hat{r}_{a,j}$) from the following equation.

$$\hat{r}_{a,j} = \overline{r}_a + \tau \sum_{i=1}^{n} s_{a,i}(r_{i,j} - \overline{r}_i) \tag{14.1}$$

where $\overline{r}_i$ is the mean preference rating of user i, $\overline{r}_a$ is the mean preference rating of the active user, $s_{a,i}$ is the similarity between user a and user i and τ is the normalizing constant such that the $\sum_{i=1}^{n} |s_{a,i}|$ becomes one. The mean preference rating is computed over the set of items that the user has evaluated. Note that the summation will only be applied to users whose preference ratings on item j are observed. For example, for predicting the preference of user a on "Independence Day" we incorporate only the ratings of users who have evaluated "Independence Day."

The predicted rating in Equation 14.1 consists of two components: the active user's own mean preference rating and the observed ratings of other users in the database. Without ratings from other users, the best guess for $\hat{r}_{a,j}$ is the mean of the user's previous preference ratings over other items. The beauty of collaborative filtering is the capability of improving the prediction accuracy by incorporating other users' opinions. The user i who experienced item j evaluates her preference rating for item j as $r_{i,j}$. So her relative preference is $(r_{i,j} - \overline{r}_i)$. This might suggest that the active user will also prefer movie j higher than her average. However, this depends on whether the active user and user i have similar tastes. This is measured by the similarity in tastes between the two users, $s_{a,i}$, which can be positive (the active user and user i tend to like the same movies) or negative (the active user and user i tend to have opposite tastes in movies).

The relative ratings of users who have rated on item j will be combined to predict $\hat{r}_{a,j}$, but the contribution of each user ($s_{a,i}$) will be different. If the other user is "similar" to the active user, a larger weight is assigned. If the other user is less similar to the active user, his or her opinions are not reflected heavily on the predicted preference rating of the active user. In sum, the predicted rating is her mean preference rating plus the weighted sum of other users' relative preferences. And the weights are determined by the similarity between each user and the active user.

The use of relative preference rather than absolute preference is based on the recognition that rating distributions are centered at different points for different users (Herlocker et al. 1999). Herlocker et al. show that the method using relative preference provides significantly better prediction accuracy than the method of using absolute preference. Some users may tend to use categories 3 to 5 on a five point rating scale while others may tend to use categories 1 to 3. If a user gives the same ratings for all items, her rating distribution will not provide any information for predicting the rating of the active user.

Going one step further, Herlocker et al. (1999) have also tried to incorporate the difference in spread between users' rating distributions. The original ratings are converted to z-scores with mean zero and variance one. However, this did not perform significantly better than relative preference approach. The authors conclude that the difference in variance among users' rating distributions does not influence the prediction performance, but the mean difference does.

14.2.1 Computing Similarity Between Users

Similarity between users, $s_{a,i}$, is the most important concept in the neighborhood-based collaborative filtering. Mainly based on the work of Breese et al. (1998) and Herlocker et al. (1999), this section reviews various similarity measures and their modifications used in collaborative filtering.

14.2.1.1 Similarity Measures

Researchers have proposed a number of different metrics that measure the similarity or the distance between users. We describe three similarity measures: Pearson correlation, Spearman rank correlation, and cosine vector similarity. Other similarity measures including the entropy-based uncertainty measure and mean-squared difference have been applied but found not to perform as well as Pearson correlation (Herlocker et al. 1999)

Pearson Correlation Coefficient

The most popular similarity measure may be Pearson correlation coefficient that GroupLens first introduced in its neighborhood-based algorithm (Resnick et al. 1994). The Pearson correlation coefficient between the active user a and the user i is given by

$$s_{a,i} = \frac{\sum_j (r_{a,j} - \overline{r}_a)(r_{i,j} - \overline{r}_i)}{\sqrt{\sum_j (r_{a,j} - \overline{r}_a)^2 \sum_j (r_{i,j} - \overline{r}_i)^2}} \tag{14.2}$$

Note that the above correlation is computed over the items that both users have evaluated. Hence, if there are only a few numbers of commonly rated items, these correlations may be unreliable.

Claiming its better performance, Shardanand and Maes (1995) have proposed the following constrained Pearson correlation coefficient as the

similarity measure.

$$s_{a,i} = \frac{\sum_j (r_{a,j} - 4)(r_{i,j} - 4)}{\sqrt{\sum_j (r_{a,j} - \overline{r}_a)^2 \sum_j (r_{i,j} - \overline{r}_i)^2}} \tag{14.3}$$

They use 4 because it is the midpoint of their seven-point rating scale. However, there are no theoretically convincing reasons why 4 is better than the mean ratings.

Spearman Rank Correlation Coefficient

Herlocker et al. (1999) criticized the rather restrictive data assumptions required for the Pearson correlation coefficient and proposed Spearman rank correlation coefficient as a similarity measure. The Pearson correlation coefficient can be cast in a regression framework that relies on several assumptions, including that the relationship is linear and the error distribution has a mean of 0 and constant variance. These assumptions are frequently violated in collaborative filtering data. Spearman rank correlation coefficient does not rely on these model assumptions. It is identical to Pearson except that it computes a measure of correlation between rank orders instead of rating values. There has not been any significant performance difference observed between Pearson and Spearman (Herlocker et al. 1999). However, these authors suggest using Spearman rank correlation for a rating scale with a small number of discrete values and Pearson correlation for the rating scale with continuous values.

(Cosine) Vector Similarity

Adopting the idea from information retrieval literature, Breese et al. (1998) have proposed the cosine vector similarity measure. In information retrieval, the vector of word frequencies characterizes each document and the angle between two vectors of word frequencies measures the similarity between two documents. Likewise, the similarity between the active user a and the other user i is defined as

$$s_{a,i} = \cos(\mathbf{r}_a, \mathbf{r}_i) = \frac{\mathbf{r}_a \cdot \mathbf{r}_i}{\|\mathbf{r}_a\| \, \|\mathbf{r}_i\|} \tag{14.4}$$

where $\mathbf{r}_a = (r_{a,1}, r_{a,2}, \ldots, r_{a,J})$, $\mathbf{r}_i = (r_{i,1}, r_{i,2}, \ldots, r_{i,J})$, $\mathbf{r}_a \cdot \mathbf{r}_i = \sum_{j=1}^{J} r_{a,j} r_{i,j}$, $\|\mathbf{r}_a\| = \sqrt{\sum_{j=1}^{J} r_{a,j}^2}$, and $\|\mathbf{r}_i\| = \sqrt{\sum_{j=1}^{J} r_{i,j}^2}$. Note that the angle is calculated over items that both users have evaluated. Because of its origin in the field of information retrieval, ratings below zero as well as the unrated items receive zero rating values.

The cosine vector similarity is mathematically similar to Pearson correlation coefficient. However, Pearson correlation takes into account the differences in rating scale between different users (i.e., $\overline{r}_i$ and $\overline{r}_a$, see Equation 14.2).

The cosine vector does not. The cosine vector similarity has been shown to be successful in the field of information retrieval, but it does not perform well compared to the Pearson correlation coefficient in the area of collaborative filtering (Breese et al. 1998).

14.2.1.2 Significance Weighting

One critical problem that all similarity measures have in common is that they do not consider the significance (or confidence) of the similarity measure. Does the correlation of 0.9 between two users tell you that two users are very similar? It depends on the confidence attached to the correlation. If 0.9 is calculated based on 100 co-rated items, we will trust the correlation and conclude that preference structures of two users are very similar. However, if the correlation of 0.9 is based on five co-rated items, we may reserve our conclusion because we are not sure whether 0.9 is the true correlation or results from chance. That is, we need to recognize that the true similarity between users is an unknown parameter. Since the calculated correlation is the estimate of the true similarity, it is important to know the confidence of our estimate.

Herlocker et al. (1999) have found that some users who correlated highly with the active user did not perform well in predicting the preference ratings of the active user. Those high correlations (with terrible predicting performance) were frequently based on tiny sample sizes (often three to five co-rated items). In order to improve the prediction accuracy, a weighting factor has been proposed to add to the correlation such that the algorithm can discount the correlations based on small samples. More specifically, if two users had fewer than 50 co-rated items, Herlocker et al. multiplied the original correlation by a significance weight of $n/50$ where n is the number of co-rated items. For more than 50 co-rated items, a significance weight of 1 was uniformly used. The authors showed that applying significance weighting improved the prediction accuracy whether it is applied to Pearson or Spearman correlation (Herlocker et al. 1999). However, they did not provide any theoretically convincing reasons why they selected 50 as opposed to 10 or 100 as a cutoff.

14.2.1.3 Variance Weighting

Similarity measures discussed so far assume that all item ratings have the identical informational value in predicting the preference of the active user. However, researchers are beginning to recognize that preference ratings on certain items are more important than others in identifying similarities among users (Breese et al. 1998; Herlocker et al. 1999). For example, the item that all users rate highly or badly is not very a useful piece of information in differentiating users. On the other hand, the item that 50% of users rate very positively and the rest rate very negatively tells us a lot about similarities among users.

For treating the informational value of each item rating differently, Herlocker et al. (1999) have proposed to incorporate the item-variance weight factor into the calculation of the Pearson correlation coefficient. More specifically, first standardize each rating so that its mean is zero and its variance is one. Then Equation 14.2 can be written as

$$s_{a,i} = \sum_{j=1}^{J} z_{a,j} z_{i,j} / J \tag{14.5}$$

where $z_{a,j} = (r_{a,j} - \overline{r}_a)/\sigma_a$, $z_{i,j} = (r_{i,j} - \overline{r}_i)/\sigma_i$ and J is the number of co-rated items. We now incorporate the item-variance weight factor into Equation 14.5, the standardized Pearson correlation.

$$s_{a,i} = \sum_{j=1}^{J} v_j z_{a,j} z_{i,j} / \sum_{j=1}^{J} v_j \tag{14.6}$$

where $v_j = (\sigma_j^2 - \sigma_{\min}^2)/\sigma_{\max}^2$, $\sigma_j^2 = \sum_{i=1}^{n} (r_{i,j} - \bar{r}_j)^2/(n-1)$, $\sigma_{\min}^2$ and $\sigma_{\max}^2$ are the minimum and maximum variances over all items.

It intuitively makes sense to incorporate the variance weighting in the similarity computation. Unfortunately, however, employing the item-variance weighting factor did not show any significant gain in the prediction accuracy (Herlocker et al. 1999).

14.2.1.4 Selecting Neighborhoods

Shardanand and Maes (1995) have observed that selecting a subset of users instead of all possible users improved the prediction performance of collaborative filtering. Adding a large number of users with correlations (similarity) that were very low in magnitude seemed to increase the noise rather than providing additional information in predicting the ratings of the active user. Hence, they set an absolute correlation threshold so that only users with absolute correlations (with the active user) greater than the threshold are selected in computing Equation 14.1. An alternative way of selecting users with high informational values is to pick the n users that correlate most highly with the active user, where n is selected by the analyst (Herlocker et al. 1999).

How do you choose a specific value for the correlation threshold or the number n for the best n neighbors? Setting a high correlation threshold or a small n will allow you to limit users to those with high correlations. However, setting too high correlation threshold or too small n makes your predictions less accurate because you are not drawing on the opinions of enough neighbors. The magic number may have to be determined empirically for the given study. Herlocker et al. (1999) have compared the performance of various combinations of the correlation threshold and the best n neighborhood. They found the best n method (with n = 20) provided the best performance overall. Adding the feature of correlation threshold to the best n method does not improve the performance of the best n method.

14.2.2 Evaluation Metrics

Research in collaborative filtering evaluates the performance of each algorithm in terms of three metrics: coverage, statistical accuracy, and decision-support accuracy (Sarwar et al. 1998).

14.2.2.1 Coverage

Coverage measures the percentage of items for which a collaborative filtering algorithm can provide predictions. Since we evaluate algorithms of collaborative filtering, item predictions of an active user should be made using other users' ratings to be included in coverage counting. We do not expect 100% coverage because none of the users may rate certain items or similarities cannot be computed for some users who do not rate any items in common with the active user. As mentioned, applying the method of correlation thresholds or the best n correlates will reduce coverage.

14.2.2.2 Statistical Accuracy

Statistical accuracy measures the closeness between the predicted and the actual rating. Various metrics including mean absolute error, root mean squared error and Pearson correlation coefficient can be employed in measuring statistical accuracy. For more discussions on statistical accuracy, see Chapter 11.

14.2.2.3 Decision-Support Accuracy

Decision-support accuracy measures how effectively predictions enhance a user's ability to find items they like among the many choices in the full item set. The user's decision is a binary process in many instances. For example, a moviegoer will or will not watch the movie, *Matrix*. An Internet shopper will or will not purchase a book from Amazon. However, as shown in Table 14.1, the input data and the corresponding predictions are rating scores from 1 to 5. Hence, we need to convert the predicted rating scores into the binary variable (0/1). Suppose a moviegoer watches the movies scored greater than or equal to 4. Items with predicted scores of 4 or 5 are converted into the recommendation (1). Otherwise, items are converted into the recommendation (0). We study two important metrics for decision-support accuracy.

ROC Sensitivity

The concept of ROC (Receiver Operating Characteristic) curve originated in the field of signal detection to measure the diagnostic power of a model (Swets

Table 14.2 True event versus diagnosis – the basis for ROC curves (From Swets 1988)

		Event		
		Positive	**Negative**	
Diagnosis	Positive	True positive (a)	False positive (b)	$a + b$
	Negative	False negative (c)	True negative (d)	$c + d$
		$a + c$	$b + d$	$a + b + c + d - N$

1988; see also Chapter 11). In order to understand this concept, let us take a look at a two-by-two contingency table shown in Table 14.2. A diagnostic system (or model) looks for a particular "signal" and ignores other events called "noise." The event is considered to be "positive" or "negative," and the diagnosis made is correspondingly positive or negative. There are two ways in which diagnosis can be correct: "true-positive" and "true-negative" in Table 14.2. And there are two cases that diagnosis can be wrong: "false-positive" and "false-negative."

The true-positive proportion, $a/(a + c)$, and the false-positive proportion, $b/(b + d)$, captures all of the relevant information on accuracy of the model. These two proportions are often called the proportion of "hits" and "false alarms."[1] A good diagnostic model will provide many hits with few false alarms.

The ROC curve plots the proportion of hits versus false alarms for various settings of the decision criterion (see Fig. 14.1). Going back to the movie recommendation example, assume we set the rating threshold very high, say 4.9. We recommend the movie if the predicted preference rating is higher than 4.9. We do not recommend otherwise. Given the rating threshold, we can prepare the two-by-two contingency table after going through the filtering algorithm for the entire database. The proportions of hits and false alarms from the table will become a point of the ROC curve for the model. Now we set the rating threshold a bit lower, say 4.8. And plot a point of the ROC curve. Changing the value of the rating threshold to 0 will complete the ROC curve.

Note an ROC curve is generated for a particular model as a function of a critical decision criterion or parameter in the model, such as a cut-off. The performance (or value) of the model is measured to be the area under the ROC curve. The area varies from 0.5 to 1. The major diagonal in Fig. 14.1 represents the case of the area equal to 0.5 when the proportions of hits and false alarms are the same. Random assignment will lead to the area of 0.5. On the other hand, a perfect model when the curve follows the left and upper axes has the area of 1. There are no false alarms with 100% hits. The realistic model lies in between. The area under the curve increases as the model can increase more hits while reducing the number of false alarms.

[1] The true positive proportion is also called "sensitivity" that is the probability of a randomly selected positive event being evaluated as positive by the model. In addition, the true negative proportion is often called specificity that is the probability of a randomly selected negative event being evaluated as negative by the model. Note that the false positive proportion is (1 – specificity).

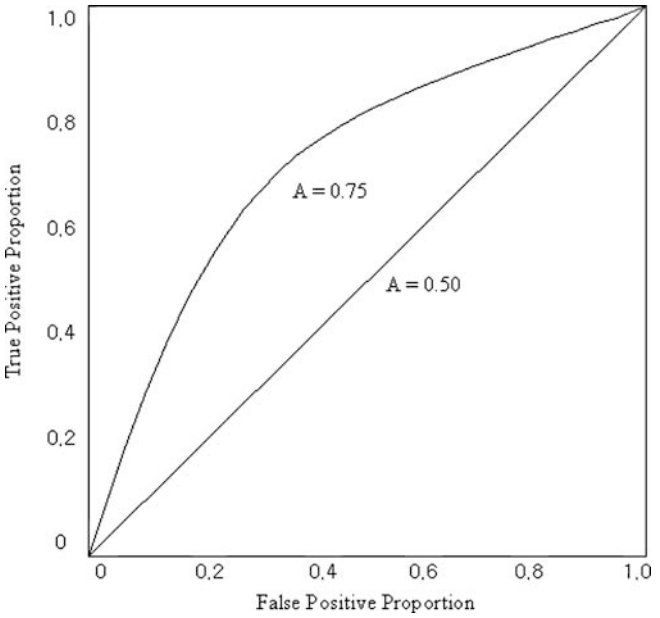

Fig. 14.1 The ROC curve (From Swets 1988).

PRC Sensitivity

Suppose that a movie prediction model recommends 100 titles and you like 50 of them, and another prediction model recommends 200 titles and you like 80 of them. Which movie prediction model is better? Researchers in information retrieval have developed decision-support accuracy measures called precision and recall to evaluate the model performance for cases as in this example.

Recall is the same as the true-positive proportion or the proportion of hits in the ROC (e.g., $a/(a+c)$ in Table 14.2). Precision is the number of true-positives divided by the total number of positives diagnosed by the model (e.g., $a/(a+b)$ in Table 14.2). Hence, precision indicates how selective the system is, and recall indicates how thorough it is in finding valuable information (Salton and McGill 1983). For example, suppose that the total number of movies in movie database a user will like is 500 (e.g., $a+c=500$). Suppose a model recommends 100 movies and the user likes 50 of them (e.g., $a+b=100$ and $a=50$). Then $b=50$ and $c=450$. Hence, the precision is 0.5 ($=50/100$) and the recall is 0.1 ($=50/500$).

The PRC (precision-recall curve) plots recall versus precision for various settings of the decision criterion. Similar to drawing the ROC curve, we start with a very high rating threshold. Given the rating threshold, say 4.5, we create the resulting two-by-two contingency table, calculate recall and

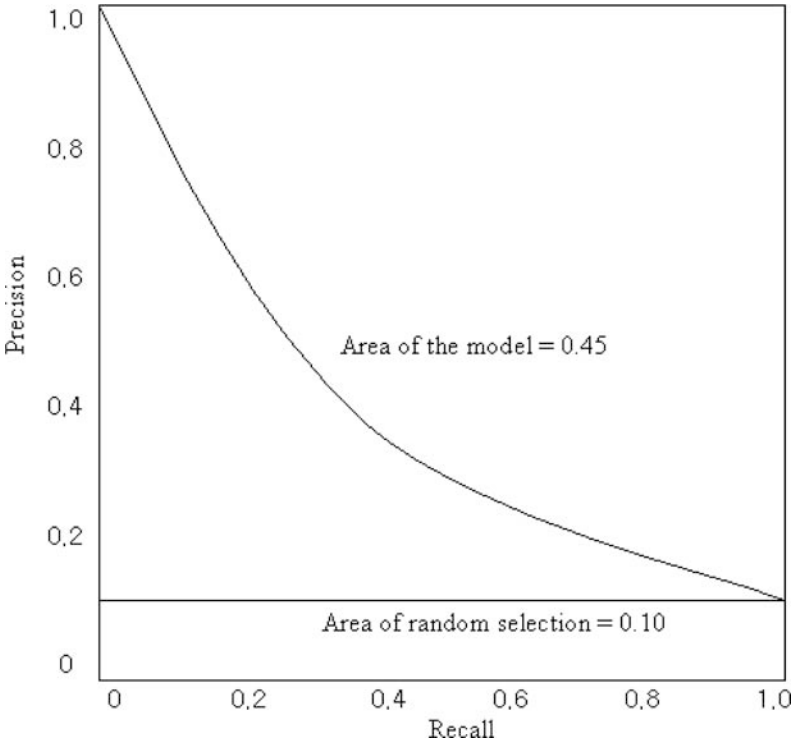

Fig. 14.2 The PRC (precision-recall curve).

precision, and plot it on the PRC graph. Decreasing the value of rating threshold gradually to zero, we can complete the locus of the PRC curve.

For a given level of recall, we would like the precision to be as high as possible. Therefore, the performance (or value) of the model is measured as the area under the PRC curve. Its shape is different from the ROC curve because of the different x-axis, as shown in Fig. 14.2. The area varies from 0 to 1. A perfect model with the area of one will have 100% precision for all recall values. The random selection will produce the horizontal line with its height determined by the proportion of actual positives in the entire samples. The area under the curve increases as the model can increase the precision for a given recall.

14.3 Model-Based Methods

Memory-based collaborative filtering has been widely used in practice because it is easy to implement. However, memory-based methods have several limitations (Ansari et al. 2000; Sarwar et al. 2001). First, when data are sparse, predictive accuracy becomes very poor. Many e-commerce sites such

as Amazon.com carry a large number of products. Even heavy users in these sites may have purchased well under 1% of the products. Because of low coverage, similarities (or correlations) between users are unreliable, and sometimes cannot be computed. Second, memory-based methods require computations that grow with both the number of customers and the number of products. They will suffer serious scalability problem with millions of customers and/or products. Model-based methods attempt to overcome these limitations of memory-based methods.

14.3.1 The Cluster Model

Cluster models treat the recommendation problem as a classification task, and identify groups consisting of users who appear to have similar preferences (Iacobucci et al. 2000). The segments are created using a clustering algorithm in which the number of clustering variables is equal to the number of items. When the number of items is too large, it is often recommended to use only items that are rated (or purchased) by a minimum number of users (or buyers). Once the segments are determined, cluster models assign a target user to the segment containing the most similar users. Some clustering algorithms classify the target user into multiple segments and calculate the strength of each segment membership. Then the predicted preference for the target user can be made by averaging the preferences of the other users in the segment.

Cluster models overcome the scalability problem of memory-based methods because they compare the target user to a small number of segments rather than the entire customer base. However, cluster models provide less personal recommendations than memory-based methods. Hence, the quality of their predictions is often poor (Sarwar et al. 2001; Linden et al. 2003). Cluster models overcome the scalability problem by grouping numerous users into a small number of segments and treating all customers in the given segment as the same in predictions. Increasing the number of segments may improve the quality of prediction, but then the scalability problem comes in.

14.3.2 Item-Based Collaborative Filtering

Unlike the memory-based collaborative filtering method that matches the target user to similar users, item-based methods consider a set of items that the target user has rated, calculate how similar they are to the target item, and then combine those similar items into the prediction (Sarwar et al. 2001; Linden et al. 2003). That is, item-based methods proceed in two steps: item similarity computation and prediction computation.

The most critical step in item-based methods is to compute the similarity between items and select the most similar items to the target item. The similarity between item i and j ($s_{i,j}$) is calculated over the users who have rated both items. For the data provided in Table 14.1, item-based methods compute the similarity between movies (or columns) while memory-based methods compute the similarity between users (or rows). There are several ways to measure the similarity between items. Here we present two most popular metrics: the Pearson correlation coefficient and the cosine vector similarity.

$$s_{i,j} = \frac{\sum_{u \in U} (r_{u,i} - \bar{r}_i)(r_{u,j} - \bar{r}_j)}{\sqrt{\sum_{u \in U} (r_{u,i} - \bar{r}_i)^2 \sum_{u \in U} (r_{u,j} - \bar{r}_j)^2}} \tag{14.6a}$$

$$s_{i,j} = \cos(\mathbf{r}_i, \mathbf{r}_j) = \frac{\mathbf{r}_i \cdot \mathbf{r}_j}{\|\mathbf{r}_i\| \, \|\mathbf{r}_j\|} \tag{14.6b}$$

In Equation 14.6a, U indicates the set of users who both rated item i and j, $r_{u,i}$ is the rating of user u on item i and $\bar{r}_i$ is the mean rating of item i over users $u \in U$. In Equation 14.6b,

$$\mathbf{r}_i = (r_{1,i}, r_{2,i}, \ldots, r_{U,i}), \mathbf{r}_j = (r_{1,j}, r_{2,j}, \ldots, r_{U,j}), \mathbf{r}_i \cdot \mathbf{r}_j$$
$$= \sum_{u=1}^{U} r_{u,i} r_{u,j}, \|\mathbf{r}_i\| = \sqrt{\sum_{u=1}^{U} r_{u,i}^2}, \text{and } \|\mathbf{r}_j\| = \sqrt{\sum_{u=1}^{U} r_{u,j}^2}.$$

Once the similarity computations between items are completed, then the predicted rating of item j for a target user u is given by the weighted sum of ratings provided by the user on the items similar to the item j. And similarities between items i and j ($s_{i,j}$) will measured by the Pearson correlation coefficient or the cosine vector similarity. That is, the predicted rating of item j for a target user u ($\hat{r}_{u,i}$) can be written as

$$\hat{r}_{u,j} = \bar{r}_u + \sum_{i=1}^{I} w_{i,j} (r_{u,i} - \bar{r}_u) \tag{14.7}$$

In Equation 14.7, $\bar{r}_u$ is a target user u's average rating across all I products he or she has rated, and $w_{i,j} = s_{i,j} / |\sum_i s_{i,j}|$ is the weighted similarity of item i and item j. As a result, the user will receive a high prediction for item j if the other items the user has rated tend to have high predictions and be positively correlated with item j. In addition, similarities are scaled (e.g., $w_{i,j}$) by the absolute sum of the similarities to make sure that the predicted rating is within the predefined range.

Sarwar et al. (2001) showed that item-based methods provided better quality prediction than the memory-based algorithm across all sparsity levels even though the difference was not significantly large. In addition, since the item-similarity matrix is fairly static and can be created offline, their online recommendation is fast even for extremely large data sets. Their

online speed of prediction does not depend on the total number of users, but depends only on how many items the target user has rated. In sum, item-based methods partially overcome the two challenging problems (data sparsity and scalability) that memory-based methods often face in online applications.

14.3.3 A Bayesian Mixture Model by Chien and George (1999)

Chien and George (1999) were the first to propose a model-based approach based on a Bayesian mixture model. They pointed out a limitation of memory-based methods that occurs when the number of co-rated items from a pair of users is very small (e.g., problem of sparse data). This can lead to high (but unreliable) similarity scores based on an extremely small number of co-rated items. Significance weighting schemes discussed earlier may alleviate this problem somewhat, but it is still heuristic. In addition, memory-based methods are not based on a statistical model so that we cannot statistically evaluate the uncertainty associated with the predicted values.

A Bayesian mixture model assumes that users who tend to give similar ratings can be modeled as having the same ratings probability distribution. That is, users can be partitioned into subgroups which are identified by common probability structure for the ratings. The prediction of a missing rating is based on the posterior distribution of the groupings and associated ratings probabilities. Markov Chain Monte Carlo (MCMC) methods with a hybrid search algorithm are used to estimate parameters and obtain predictions of the missing ratings. Chien and George (1999) show that their model outperforms memory-based methods both on two simulated data sets and a real data. We do not describe their model in detail here because a hierarchical Bayesian model in the next section overlaps with their model and is more practical.

14.3.4 A Hierarchical Bayesian Approach by Ansari et al. (2000)

Ansari et al. (2000) propose an effective hierarchical Bayesian model to predict missing ratings. They adopt a regression-based approach and model customer (or user) ratings as a function of product (or item) attributes, customer characteristics, and expert evaluations. Their model also accounts for unobserved sources of heterogeneity in customer preferences and product appeal structures. Specifically, customer i's rating on product j can be

written as

$$\begin{aligned} r_{ij} &= \mathbf{x}'_{ij}\mu + \mathbf{z}'_i\gamma_j + \mathbf{w}'_j\lambda_i + e_{ij} \\ e_{ij} &\sim N(0,\sigma^2), \lambda_i \sim N(\mathbf{0},\mathbf{\Lambda}),\ \gamma_j \sim N(\mathbf{0},\mathbf{\Gamma}) \end{aligned} \tag{14.8}$$

In Equation 14.8, the vector $\mathbf{x}_{ij}$ contains all observed product attributes, customer characteristics, and their interactions. The vector $\mathbf{z}_i$ contains customer characteristics for customer i and the vector $\mathbf{w}_j$ represents product attributes for product j. The random effects λ_i account for unobserved sources of customer heterogeneity and appear in the model interactively with the observed product attributes. Similarly, the random effects γ_j represent unobserved sources of product attributes and appear in the model interactively with the observed customer characteristics. The variance-covariance matrices $\mathbf{\Lambda}$ and $\mathbf{\Gamma}$ provide the information about the extent of unobserved heterogeneity in customer characteristics and product attributes, respectively.

Ansari et al. (2000) applied their model to the EachMovie data, which is a popular movie rating database frequently used by collaborative filtering researchers. Model parameters are estimated by Markov Chain Monte Carlo. Their estimation results provided several interesting findings. First, they compare the full model in Equation 14.8, the model with customer heterogeneity only, the model with product heterogeneity, and the model without any heterogeneity at all. The full model outperforms all the other models on various comparison criteria. They conclude that it is important to consider both customer and product heterogeneity. In addition, accounting for customer heterogeneity is more important than accounting for product heterogeneity. Second, they compared their hierarchical Bayesian model with memory-based methods and showed that their model is substantially better in rating predictions.

The main contribution of Ansari et al.'s method in collaborative filtering research is to adopt a statistically formal approach. Unlike the memory-based approach, their model can provide information on how accurate their rating predictions are from the corresponding posterior distribution. In addition, their model can be used even when rating or preference data for an item does not exist. For example, it can provide the predicted ratings for a new movie, given data on its attributes and the characteristics of customers (x, z, and w in Equation 14.8). Their model also has some advantages over Chien and George's Bayesian mixture approach. Ansari et al.'s model explicitly incorporates explanatory variables (e.g., customer characteristics and item attributes) so that it can explain why customers like or dislike a product. Finally, Chien and George's Bayesian mixture assumes that all customers in a given segment have the same preference structure. This assumption can be restrictive in practice, and a large number of parameters are required to be estimated. On the other hand, Ansari et al.'s model employs continuous heterogeneity so that each customer has his/her unique set of preferences.

14.4 Current Issues in Collaborative Filtering

For the last 10 years, a number of researchers have improved the algorithm of collaborative filtering significantly. In addition, it has been commercially incorporated in several web sites for many years now. However, we still have a number of practical and theoretical issues to be resolved.

14.4.1 Combining Content-Based Information Filtering with Collaborative Filtering

As shown, collaborative filtering is designed to solve the problem of making accurate and efficient recommendations, that is, it is a recommendation "engine." There is an alternative method to approach the same problem: content-based information filtering. Each method has its advantages and disadvantages. This section will briefly study the content-based information filtering and introduce recently developed techniques combining it with collaborative filtering.

14.4.1.1 Content-Based Information Filtering

Content-based information filtering recommends items for users by analyzing the content of items that they liked in the past (Balabanovic and Shoham 1997). Its underlying assumption is that the content of an item is what determines the user's preference (Balabanovic 1997). We predict the item preferences of an active user from the observed preferences from other users in collaborative filtering. In contrast, content-based filtering does not use the preferences of other users. Instead, its prediction is solely based on the content of an item and the historical preference of the active user.[2]

Content-based methods have been widely used. E-mail filtering software sorts e-mail into categories according to the sender and content of the title. New-product notification services advertise a new book or album by the user's favorite author or artist (Schafer et al. 1999). Search engines such as Yahoo recommend relevant documents on the basis of user-supplied keywords (Ansari et al. 2000).

To show the main idea of the content-based filtering, let us define an item as a vector $X = (x_1, \ldots, x_K)$ where each element x_i is the content or the attribute of the item. For example, an item can be a movie *Matrix* and its

[2] The hierarchical Bayesian approach by Ansari et al. (2000) models customer ratings as a function of product (or item) attributes. Hence, it can be considered as content-based methods. However, customer ratings in their model not only depend on product attributes but also customer characteristics and expert evaluations. It is a hybrid method to combine content-based methods and collaborative filtering.

contents may be its genre, its main actor/actress, its director, etc. An active user has evaluated a set of items so that we observe their preference ratings with their content information. For each user, the set of evaluated items will be used as the estimation sample for the content-based filtering. That is, we run a regression model $r = f(x_1, \ldots, x_K)$ for each user where r is the item ratings. Once the model is estimated, we predict the preference ratings of missing items for the active user.

14.4.1.2 Combining Techniques

Content-based filtering is an effective recommendation tool for new items where rating information from other users does not exist. However, it also has several limitations (Sarwar et al. 1998; Good et al. 1999). First, it often provides bad recommendations since it only considers the pre-specified content of items. If two items have the same content, it will predict them to have the identical ratings. Second, it tends to restrict the scope of the recommendation to the similar items that consumers have already rated (Balabanovic and Shoham 1997).

In contrast, collaborative filtering overcomes the limitations of the content-based filtering by enabling consumers to share their opinions and experiences on items (Herlocker et al. 1999). It recommends items that similar consumers have liked. It automates the process of word-of-mouth communication among consumers. However, it also has its own limitations. First, collaborative filtering does not work very well when the number of evaluators/users is small relative to the volume of information in the system. That is, it is difficult to find similar users in predicting ratings for some unpopular items. Second, it has an early rater problem that occurs when a new product/item appears in the database. Note that the collaborative filtering cannot provide the predictive ratings for a new product until some consumers have evaluated it.

Recognizing that the content-based and the collaborative filtering systems both have their advantages and disadvantages in recommending products, researchers have recently attempted to develop a hybrid model to combine these two approaches (Balabanovic 1997; Balabanovic and Shoham 1997; Basu et al. 1998; Sarwar et al. 1998; Good et al. 1999; Herlocker et al. 1999; Kim and Kim 2001). Claiming that their models take advantage of the collaborative filtering approach without losing the benefit of the content-based approach, they have shown that their models performed better than the individual approach. We describe the hybrid model by Kim and Kim (2001).

14.4.1.3 Hybrid Model by Kim and Kim (2001)

Taking a statistically more formal approach than the other combining methods, Kim and Kim (2001) have developed a hybrid recommender system that

combines the content-based and the collaborative filtering systems. Their point of departure is first to extract the content component of products/items by employing a regression and then to apply the collaborative filtering to the consumer's preference unexplained by this (content-based) regression. Specifically, the authors apply a simple regression to extract item attributes and then adjust upward or downward based on whether similar users have positive or negative errors associated with the regression-based prediction.

The Algorithm

The algorithm consists of six major steps. First, the system needs to determine a set of content features to characterize products/items. For the moviegoer example, key features may include the genre of the movie, the director, the producer, the main actors/actresses and so on.

The second step is to identify the relationship between item features and preference ratings. They use a simple regression applied to each user.

$$r_{aj} = \beta_{0a} + \beta_{1a} X_{1aj} + \ldots + \beta_{ai} X_{Kaj} + \varepsilon_{aj} \tag{14.9}$$

where r_{aj} is the preference rating of the active user a for item j, K is the number of specified features and X_{1aj} is the value of the 1st feature for product j evaluated by the active user a. The parameters to be estimated (or β's) in the Equation 14.9 measure how important each feature is in determining the preference of the user. Upon estimation, the model can predict the active user a's preference on items not yet evaluated. For the moviegoer example in Table 14.1, we would apply the regression with Amy's observed movie preferences as a dependent variable and the corresponding movie features as independent variables. Given the features of Movie 3, the estimation model would be used to predict Amy's rating for Movie 3 that is not rated.

Steps 1 and 2 are nothing but a version of content-based filtering. Only the contents of items are utilized to predict the preferences of a user. Content-based filtering cannot explain anything beyond item features. For example, a user may provide different ratings for the two movies with identical features. Many other factors than the specified item features will influence the preference ratings. The unexplained portion of user ratings will be modeled in the following steps.

Steps 3 and 4 are required to derive the matrix of prediction errors. The prediction error ($\varepsilon_{aj} = r_{aj} - \hat{r}_{aj}$) is the difference between the actual preference and the predicted preference for user a, movie j. In other words, the prediction errors are the residuals in regression model or the preferences unexplained by the model. It is required to calculate the predicted ratings for both observed and missing items. But the prediction errors for missing items cannot be calculated. Hence, the data matrix of prediction errors consists of a series of prediction errors with a set of missing values.

Step 5 is to apply the collaborative filtering technique to the prediction error matrix. Kim and Kim (2001) have employed a typical neighborhood-based algorithm to calculate the values for missing cells. Hence, the predicted rating of an active user t on product j ($e_{t,j}$) can be calculated as

$$e_{a,j} = \bar{\varepsilon}_a + \tau \sum_{i=1}^{n} s_{a,i}(\varepsilon_{i,j} - \bar{\varepsilon}_i) \tag{14.10}$$

where and n is the number of users in the prediction error matrix who have evaluated the item j. The weight $s_{a,i}$ is the (error) similarity between user i and the active user a. And τ is a normalizing factor such that the absolute values of the weights sum to one.

The final step is to sum the outputs from the third step and the fifth step. For the missing cells, the content-based filtering in step 3 provides $\hat{r}_{aj}$ while the collaborative filtering in step 5 produces e_{aj}. The predicted rating of item j for the active user a is the sum of these two numbers. Summarizing the algorithm,[3]

STEP 1: Determine a set of content features characterizing items.
STEP 2: Fit the (features) regression for each user.
STEP 3: Calculate the fitted preferences for all users and all items.
STEP 4: Derive a matrix of prediction errors.
STEP 5: Apply the collaborative filtering into the error matrix.
STEP 6: Sum the output from STEP 3 and STEP 5.

Performance of the Model

Kim and Kim's hybrid model is applied to the movie rating data. Four other competing models are also applied to the data. The baseline model that predicts the rating for each movie as the mean rating across users will benchmark the performance of the other personalized recommender systems. The second model is a content-based filtering method where the genres of the movie are used as the contents of the movie/item. A dummy variable is created for each of the ten genre variables including comedy, drama, action, art/foreign, classic, animation, family, romance, horror and thriller. A movie can be simultaneously classified into more than one of these genres. For each user, actual movie ratings are regressed on these ten genre dummies. The third is a collaborative filtering model using the neighborhood-based algorithm with similarities between users measured by Pearson correlation coefficients. In addition, twenty co-rated items are used as the cutoff for significance weighting, and the users with less than 0.01 correlations are not included as a set of neighborhood.

[3] A hybrid model by Kim and Kim (2001) has a same objective as a hierarchical Bayesian approach by Ansari et al. (2000) in combining content-based methods and collaborative filtering. Ansari et al.'s approach is statistically more rigorous. However, it is much easier to implement Kim and Kim's method.

Table 14.3 Predictive accuracy of various recommender models (From Kim and Kim 2001)

Type of Model	MAE	ROC
Baseline model	0.2238	0.7398
Content-based filtering	0.2103	0.7640
Collaborative filtering	0.1955	0.8058
Model by Kim and Kim (2001)	0.1832	0.8328

Table 14.3 shows the prediction accuracy for five models in the validation sample. Two performance measures are employed: the MAE and the ROC sensitivity measure. As expected, models incorporating some personalized components outperform the (aggregate) baseline model with respect to both MAE and ROC. In addition, the hybrid model is shown to outperform all other models for both prediction measures. With respect to the ROC, the hybrid model improves the predictive performance of the content-based and the collaborative filtering by 6.8% and 2.6%, respectively.

14.4.2 Implicit Ratings

So far we have limited our attention to case where we have preference rating data explicitly expressed by users on a discrete numerical scale. In the real world, however, explicit rating information is not always available. Often the data available are behavioral measures such as the user's web-browsing pattern, purchase history and so on. These data provide *implicit* ratings of products. GroupLens research shows that predictions based on reading time are nearly as accurate as predictions based on explicit numerical ratings in the news article recommendation (Konstan et al. 1997). However, Mild and Reutterer (2001) applied various versions of memory-based methods to actual purchase choices and found poor predictive performance. Considering its practical importance, the research on implicit ratings is relatively rare.

It is not obvious how to develop an implicit score from customers' purchase histories. One cannot simply assign one for an item purchased and zero for an item not purchased. A purchase of an item may imply that customer likes it.[4] But if an item has not been purchased, this might mean the customer dislikes it or the customer does not know its availability, or something else. Moreover, technically speaking, we cannot apply collaborative filtering to the data filled with either 1 or 0. You need 1's and 0's with missing cells that will be predicted.[5]

[4] Even this is debatable because purchase does not always imply the consumer liked the product. CDNOW allows customers later to go back and say "*own it but dislike it*" (Schafer et al. 1999).

[5] Sarwar et al. (2000) suggest that the frequency of purchases instead of purchase indicator (e.g., one) can be used for repeat purchase products.

An easy way of overcoming the problem is to use the default voting (Breese et al. 1998). The idea has been developed out of the observation that, when users only rate a small number of items, the correlation algorithm and neighbor-based collaborative filtering will not work well because it uses only ratings of the items both users have rated. The authors suggest that some default rating value is assigned for not-rated items such that the correlation is calculated over the union of the rated items. They also suggest that the same default value is assigned for some additional items k that neither user has rated. However, Breese et al. (1998) did not provide any justification on why some not-rated items should get default ratings and others should not. More research is definitely required.

Alternatively, Mild and Reutterer (2003) propose using the Jaccard or Tanimoto coefficient as the similarity measure to overcome the limited variance in similarities constructed for very sparse binary purchase data. The Tanimoto similarity between two users a and i ($s_{a,i}$) is defined as

$$s_{a,i} = \frac{\text{the number of items that both users purchased}}{\text{the number of items that either user } a \text{ or user } i \text{ purchased}}$$

That is, the Tanimoto similarity ignores the number of coinciding non-chosen items. Mild and Reutterer (2003) show that the Tanimoto similarity outperforms traditional similarity measures in the case of extremely asymmetric distributed or sparse data vectors (e.g., many zeros).

Another approach would be to calculate simple correlations between 0 and 1 data for the similarity measure for item-based collaborative filtering as in Sect. 14.3.2, Equation 14.7. That is, for each pair of items, we would calculate the correlation across users between 0 and 1 data of whether or not they purchased the product. That would provide the correlation needed in Equation 14.7. We would interpret a high correlation as meaning that customers who buy one product tend to buy the other. So if the active customer has purchased products that correlate positively with Product A, we would predict that the customer would be interested in purchasing Product A. This of course is a simple, brute force approach that avoids the fact that we do not know how to interpret the 0's and 1's (see Iacobucci et al. (2000) for a criticism of the use of correlations between 0 and 1 variables), but is worthy of future research because in all likelihood, the customer who has purchased a specific product tends to like it. Using this theme in an item-based collaborative filtering system would essentially just be an extension of market basket analysis (see Chapter 13), which looks at conditional probabilities of purchasing one product, given another product has been purchased. This is similar conceptually to the correlation that would be used in the item-based collaborative filtering system based on 0–1 data.

We may be able to increase the quality of data considerably by collecting implicit data that imply negative user preference. For example, information on returned products may indicate negative preference (Schafer et al. 1999). Or customer complaints on products/services can be incorporated. However,

a really complex question on implicit negative preference may be how to code/rate them. If 1 is for purchase and 0 for non-purchase, then what number should be assigned to the implicit negative preference such as the product return? Without any theoretical development, researchers have empirically searched for the optimal value leading to the best prediction performance. The better approach may be to construct the unobserved distribution of customer preference on the product.

Another issue on implicit ratings is how to combine the explicit and the implicit rating. We might first derive the implicit preference ratings from purchase histories. The rating database will be revised once the explicit rating information is available. For example, Amazon.com users sometimes provide explicit ratings on books they bought. It is an interesting future research agenda to develop a concrete method of initializing ratings with implicit rating information and updating them with explicit rating information.

14.4.3 Selection Bias

Most collaborative filtering algorithms assume that the missing data is generated randomly. However, customers can evaluate only products that they have purchased. Or the set of movies rated by a moviegoer may be movies that she tends to like. We do not know all possible causes of the missing data pattern, but we know that most data employed in collaborative filtering research has non-ignorable missing data pattern.

If the missing evaluations are missing at random, we can safely assume that the missing data does not have any informational value in rating predictions. That is, the fact that the evaluation is missing does not depend on the value of that evaluation, when the missing evaluations are missing at random (Ying et al. 2006). But this assumption may be too restrictive in practice. Some customers do not provide ratings simply because they have not purchased the product in question. And the reason why they did not purchase the product may be that they did not like it. Failing to incorporate missing-data mechanism can lead to biased estimates unless the missing data is generated completely at random (Little and Rubin 1987). That is, our rating predictions can be suboptimal if we ignore missing-data mechanism.

Ying et al. (2006) account for non-random missing data by proposing a joint model for the latent processes underlying selection and prediction. In their model, the fact that a product is "selected" for evaluation influences the predicted rating of the product. More specifically, their selection and prediction model can be written as

$$\begin{aligned} U_s &= \beta_s X_s + \varepsilon_s \\ U_p &= \beta_p X_p + \varepsilon_p \\ (\varepsilon_s, \varepsilon_p) &\sim N(0, 0, 1, 1, \rho) \end{aligned} \tag{14.11}$$

Similar to Ansari et al. (2000), they view product evaluations (or recommendations) as a target customer's latent consumption utility. In Equation 14.11, the subscript s indicates the selection component while the subscript p indicates the prediction component. And X represents all covariates such as product or customer characteristics.

The selection equation in Equation 14.11 is the key to incorporate nonrandom missing data mechanism. The latent value U_s can be translated into the observed quantity by assuming that $P(Y_s = 1) = P(0 < U_s)$ where Y_s is the 0/1 indicator representing whether the product is evaluated (see Chapter 11 for more discussion of selection models). The correlation ρ, representing the correlation between the error terms, is allowed to vary across customers captures the interrelation between the selection and the prediction processes.

We can interpret Equation 14.11 as a two-step rating process even though Ying et al. did not make any assumptions on temporal ordering of these two processes. First, a customer will make a decision on whether the item is evaluated (or purchased) according to the selection model. And then if the selection decision is affirmative, s/he will make the rating decision. If we ignore the selection model, the expected value for the prediction utility should be $E(U_p) = \beta_p X_p$. When the selection process is considered, it will change. For example, suppose that a customer has decided to evaluate (e.g., rating is not missing). Then the expected value for the prediction utility should be

$$E(U_p|U_s > 0) = \beta_p X_p + \rho\phi(\beta_s X_s)/\Phi(\beta_s X_s) \tag{14.12}$$

where $\phi(\cdot)$ and $\Phi(\cdot)$ are the density function and distribution function of the standard normal evaluated at $\beta_s X_s$. That is, incorporating the selection process changes the expected value for the prediction utility. It is also interesting to note the role of the correlation ρ. If the selection process is not correlated with the prediction process (e.g., $\rho = 0$), the expected value for the prediction utility in Equation 14.12 becomes $\beta_p X_p$ that is the same as the expected value without the selection model.

Applying their model into the EachMovie data, Ying et al. (2006) found that the correlation ρ is significantly different from zero. Hence, the missing-data generation process is not random. In addition, the inclusion of the selection model clearly improves the prediction accuracy for the item ratings.

14.4.4 Recommendations Across Categories

Current recommendation systems are designed to recommend a set of items from a single product category, whether their algorithms are based on collaborative filter, content-based filtering, or combining approach. For example, the recommendation system may select ten movies among thousands of movie titles. An interesting marketing issue may be the possibility of predicting item

preference in one category based on the preference information from other product categories. For example, consider an Internet store selling books and CDs that attempts to develop a recommender system. One can construct two separate recommendation engines, one of book customers and the other for CD customers. This approach is somewhat limited in utilizing cross-category information. Alternatively, ignoring category differences and treating all items from books and CDs homogeneously, one may develop one recommendation system. Even though this approach maximizes the use of customer purchase information, it may lead to the prediction bias resulting from mixing apples and oranges. Finally, we can think of a method somewhere between these two extreme approaches. Acknowledging category differences, we can still use cross-category purchase information in predicting customer behavior.

One way of incorporating the category effect is to weight ratings differently in calculating similarities between customers and mean ratings. For example, suppose there are two categories, books and CDs. When we predict the preference ratings of books, we multiply the ratings of CDs by a factor α and apply the collaborative filtering algorithm. If α is equal to zero, then the approach becomes two separate recommendation engines. On the other hand, if α is equal to one, it becomes the approach of ignoring cross-category effects. If α is somewhere between, we are assuming that the importance of CD purchase information is α times as much as that of book purchase information in predicting the preference rating of books. Similarly, when we predict the preference ratings of CDs, we multiply the ratings of books by a factor β and apply the collaborative filtering algorithm. That is, we can allow for any asymmetric effect in providing information. That is, the purchase information on books may be valuable in predicting purchase preferences of the same customer for CDs. At the same time, the purchase information on CDs may not be very useful in predicting purchase preferences for books.

How do we determine the values of α and β? A simple approach may be to determine empirically. That is, try every possible value from 0 to 1, and choose the ones that would provide the best prediction performance.