Model-based Approach for Collaborative Filtering

Minh-Phung Thi Do
University of Information Technology, Ho Chi Minh city, Vietnam

Dung Van Nguyen
University of Information Technology, Ho Chi Minh city, Vietnam

Loc Nguyen
Faculty of Information Technology, University of Natural Science, Ho Chi Minh city, Vietnam

Abstract
Collaborative filtering (CF) is popular algorithm for recommender systems. Therefore items which are recommended to users are determined by surveying their communities. CF has good perspective because it can cast off limitation of recommendation by discovering more potential items hidden under communities. Such items are likely to be suitable to users and they should be recommended to users. There are two main approaches for CF: memory-based and model-based. Memory-based algorithm loads entire database into system memory and make prediction for recommendation based on such in-line memory database. It is simple but encounters the problem of huge data. Model-based algorithm tries to compress huge database into a model and performs recommendation task by applying reference mechanism into this model. Model-based CF can response user's request instantly. This paper surveys common techniques for implementing model-based algorithms. We also give a new idea for model-based approach so as to gain high accuracy and solve the problem of sparse matrix by applying evidence-based inference techniques.

Keywords: collaborative filtering, memory-based approach, model-based approach, expectation maximization, Bayesian network.

1. Introduction
Recommendation system is a system which recommends items to users among a large number of existing items in database. Item is anything which users consider, such as product, book, and newspaper. There is expectation that recommended item are items that user will like most; in other words, such items are in accordance with user's interest.

There are two common trends of recommendation systems: content-based filtering (CBF) and collaborative filtering (CF) as follows [1, pp. 3-13]:

- CBF recommends an item to a user if such item is similar to other items that she/he likes much in the past (her/his rating for such item is high). Note that each item has contents which are properties and so all items compose a so-called item content matrix.
- CF recommends an item to a user if her/his neighbors (other users similar to her/him) are interested in such item. Note that user's rating on an item expresses her/his interest. All users' ratings on items compose a so-called rating matrix.

Both of them (CBF and CF) have their own strong points and weak points. Namely CBF focuses on content of item and user's own interest; it recommends different items to different users. Each user can receive unique recommendation; so this is the strong point of CBF. However CBF doesn't tend towards community like CF. As items that user may like "are hidden under" user community, CBF has no ability to discover such implicit items. This is the most common weak point of CBF.

If there are a lot of content associating with item (for example, items has many properties) then, CF consumes much system resource and time in order to analyze items whereas CF doesn't regard to content of items. That CF only works on users' ratings on items is strong point because CF doesn't encounter how to analyze rich content items. However it is also weak point because CF can do unexpected recommendation in some situations that items are considered to be suitable to user but they don't relate to user profile in fact. The problem gets more serious when there are many items that aren't rated and so rating matrix becomes spares matrix containing many missing values. In order to alleviate such weak point of CF, there are two approaches that improve CF:

- Combination of CF and CBF [2]. This technique is divided into two stages. Firstly, it applies CBF into setting up the complete rating matrix. Secondly, CF is used to make prediction for recommendation. This technique improves the precision of prediction but it takes much time when the first stage plays the role of filter step or pre-processing step. The content of item must be fully represented. It means that this technique requires both item content matrix and rating matrix.

- Compressing rating matrix into representative model which is used to predict missing values for recommendation. This is model-based approach for CF. Note that CF has two common approaches such as memory-based and model-based. The model-based approach applies statistical and machine learning methods to mining rating matrix. The result of mining task is the mentioned model.

Although the model-based approach doesn't give result which is as precise as the combination approach, it can solve the problem of huge database and sparse matrix. Moreover it can responds user's request immediately by making prediction on representative model through instant inference mechanism. So this paper focuses on model-based approach for CF. In section 2 we skim over the memory-based CF. Model-based CF is discussed carefully in section 3. We propose an idea for a model-based CF algorithm in section 4. Section 5 is the conclusion.

2. Memory-based collaborative filtering

Memory-based CF [1, pp. 5-8] algorithms use the entire or a sample of the user-item database to generate a prediction. Every user is part of a group of people with similar interests. The essence of the neighborhood-based CF algorithm [3, pp. 16-18], a prevalent memory-based CF algorithm, is to find out the nearest neighbors of a regarded user (so-called active user). Suppose we have a rating matrix in which rows indicate users and columns indicate items and each cell is a rating which a user gave to an item [3, p. 16]. In other words, each row represents a user vector or rating vector. The rating vector of active user is called as active user vector. Table 1 is an example of rating matrix with one missing value. Note, missing value is denoted by question mask (?). For example, r_{43} and r_{44} are missing values, which means that user 4 does not rate on items 3 and 4.

	Item 1	Item 2	Item 3	Item 4
User 1	$r_{11} = 1$	$r_{12} = 2$	$r_{13} = 1$	$r_{14} = 5$
User 2	$r_{21} = 2$	$r_{22} = 1$	$r_{23} = 2$	$r_{24} = 4$
User 3	$r_{31} = 4$	$r_{32} = 1$	$r_{33} = 5$	$r_{34} = 5$
User 4	$r_{41} = 1$	$r_{42} = 2$	$r_{43} = ?$	$r_{44} = ?$

Table 1. Rating matrix (user *4* is active user)

Let $u_i = (r_{i1}, r_{i2}, \ldots, r_{in})$ and $a = (r_{a1}, r_{a2}, \ldots, r_{an})$ be the normal user vector *i* and the active user vector *a*, respectively where r_{ij} is the rating of user *i* to item *j*. According to table 1, we have $u_1 = (1, 2, 1, 5)$, $u_2 = (2, 1, 2, 4)$, $u_3 = (4, 1, 5, 5)$, and $a = u_4 = (1, 2, ?, ?)$.

In situation that some cells which belong to active user vector are empty; it means that active user didn't rate respective items and rating matrix becomes sparse matrix. The problem which needs to be solved is to predict missing values of active user vector; later the items having the highest values are recommended to active user [4, p. 288]. There are two steps in process of predicting missing values [3, pp. 17-18]:

1. Finding out nearest neighbors of active user [3, pp. 17-18].
2. Computing predictive values (or predictive ratings) [3, p. 18].

Note that computing predictive values is based on finding out nearest neighbors of active user.

2.1. Finding out nearest neighbors of active user

The similarity of two user vectors is used to specify the nearest neighbors of an active user. The more the similarity is, the nearer two users are. Given a threshold, users that the similarities between them and active user are equal to or larger than this threshold are considered as nearest neighbors of active user. There are two popular similarities such as cosine similarity and Pearson correlation.

Let *I* be the set of indices of items on which user u_i rates and so we have $I = \{j: r_{ij} \neq ?\}$. Let *A* be the set of indices of items on which active user *a* rates and so we have $A = \{j: r_{aj} \neq ?\}$. Let *V* be the intersection set of *I* and *A* and so we have $V = I \cap A$, which means that *V* is the set of indices of items on which both user u_i and active user *a* rate.

The cosine similarity measure of two users is the cosine of the angle between two user vectors [3, p. 17], [4, p. 290].

$$sim(a, u_i) = \cos(a, u_i) = \frac{a \bullet u_i}{|a||u_i|} = \frac{\sum_{j \in V} r_{aj} r_{ij}}{\sqrt{\sum_{j \in V} r_{aj}{}^2} \sqrt{\sum_{j \in V} r_{ij}{}^2}}$$

Where the sign "•" denotes scalar product (dot product) of two vectors. Notations $|a|$ and $|u_i|$ denote the length (module) of *a* and u_i, respectively. Because all ratings are positive or equal 0, the range of cosine similarity measure is from 0 to 1. If it is equal to 0, two users are totally different. If it is equal to 1, two users are identical. For example, the cosine similarity measures of active user (user 4) and users 1, 2, 3 in table 1 are:

$$sim(u_4, u_1) = \frac{r_{41} r_{11} + r_{42} r_{12}}{\sqrt{r_{41}^2 + r_{42}^2} \sqrt{r_{11}^2 + r_{12}^2}} = \frac{1 * 1 + 2 * 2}{\sqrt{1^2 + 2^2} \sqrt{1^2 + 2^2}} = 1$$

$$sim(u_4, u_2) = \frac{r_{41} r_{21} + r_{42} r_{22}}{\sqrt{r_{41}^2 + r_{42}^2} \sqrt{r_{21}^2 + r_{22}^2}} = \frac{1 * 2 + 2 * 1}{\sqrt{1^2 + 2^2} \sqrt{2^2 + 1^2}} = 0.8$$

$$sim(u_4,u_3)=\frac{r_{41}r_{31}+r_{42}r_{32}}{\sqrt{r_{41}^2+r_{42}^2}\sqrt{r_{31}^2+r_{32}^2}}=\frac{1*4+2*1}{\sqrt{1^2+2^2}\sqrt{4^2+1^2}}\approx 0.65$$

Obviously, user 1 and user 2 are similar to user 4 than user 3 is, according to cosine similarity. Given a threshold 0.5, users 1 and 2 are neighbors of active user 4.

Statistical Pearson correlation is also used to specify the similarity of two vectors. Suppose r_{ij} and r_{aj} denote the ratings of user *i* and active user *a* to item *j*, respectively. Let $\bar{r}_i$ and $\bar{r}_a$ be the average ratings of normal user *i* and active user *a*, respectively. We have:

$$\bar{r}_i=\frac{1}{|I|}\sum_{j\in I}r_{ij}$$
$$\bar{r}_a=\frac{1}{|A|}\sum_{j\in A}r_{aj}$$

The Pearson correlation is defined as below [4, p. 290], [5, p. 40]:

$$sim(a,u_i)=\text{pearson}(a,u_i)=\frac{\sum_{j\in V}(r_{aj}-\bar{r}_a)(r_{ij}-\bar{r}_i)}{\sqrt{\sum_{j\in V}(r_{aj}-\bar{r}_a)^2}\sqrt{\sum_{j\in V}(r_{ij}-\bar{r}_i)^2}}$$

The range of Pearson correlation is from –1 to 1. If it is equal to –1, two users are totally different. If it is equal to 1, two users are identical. For example, we need to compute Pearson correlation between active user (user 4) and users 1, 2, 3 in table 1. We have:

$$\bar{r}_4=\frac{1+2}{2}=1.5,\bar{r}_1=\frac{1+2+1}{3}\approx 1.33,\bar{r}_2=\frac{2+1+2}{3}\approx 1.66,\bar{r}_3=\frac{4+1+5}{3}\approx 3.33$$

It implies:

$$sim(u_4,u_1)=\frac{(r_{41}-\bar{r}_4)(r_{11}-\bar{r}_1)+(r_{42}-\bar{r}_4)(r_{12}-\bar{r}_1)}{\sqrt{(r_{41}-\bar{r}_4)^2+(r_{42}-\bar{r}_4)^2}\sqrt{(r_{11}-\bar{r}_1)^2+(r_{12}-\bar{r}_1)^2}}$$
$$=\frac{(1-1.5)(1-1.33)+(2-1.5)(2-1.33)}{\sqrt{(1-1.5)^2+(2-1.5)^2}\sqrt{(1-1.33)^2+(2-1.33)^2}}\approx 0.95$$
$$sim(u_4,u_2)=\frac{(r_{41}-\bar{r}_4)(r_{21}-\bar{r}_2)+(r_{42}-\bar{r}_4)(r_{22}-\bar{r}_2)}{\sqrt{(r_{41}-\bar{r}_4)^2+(r_{42}-\bar{r}_4)^2}\sqrt{(r_{21}-\bar{r}_2)^2+(r_{22}-\bar{r}_2)^2}}$$
$$=\frac{(1-1.5)(2-1.66)+(2-1.5)(1-1.66)}{\sqrt{(1-1.5)^2+(2-1.5)^2}\sqrt{(2-1.66)^2+(1-1.66)^2}}\approx -0.95$$
$$sim(u_4,u_3)=\frac{(r_{41}-\bar{r}_4)(r_{31}-\bar{r}_3)+(r_{42}-\bar{r}_4)(r_{32}-\bar{r}_3)}{\sqrt{(r_{41}-\bar{r}_4)^2+(r_{42}-\bar{r}_4)^2}\sqrt{(r_{31}-\bar{r}_3)^2+(r_{32}-\bar{r}_3)^2}}$$
$$=\frac{(1-1.5)(4-3.33)+(2-1.5)(1-3.33)}{\sqrt{(1-1.5)^2+(2-1.5)^2}\sqrt{(4-3.33)^2+(1-3.33)^2}}\approx -0.87$$

Obviously, only user 1 is similar to user 4 according to Pearson correlation. Given a threshold 0.5, only user 1 is neighbor of active user 4.

2.2. Computing predictive values

A predictive value or predictive rating is the value that replaces a missing value in active user vector. Suppose we have *m* nearest neighbors of active user are determined from the first step "Finding out nearest neighbors of active user". Let *sim*(*a*, *u*$_i$) be the similarity between normal user *i* and active user *a*. Let r_{aj} be the predictive value for item *j* of active user vector. According to [3, p. 18], we have:

$$r_{aj}=\bar{r}_a+\frac{\sum_{i=1}^{m}(r_{ij}-\bar{r}_i)sim(a,u_i)}{\sum_{i=1}^{m}|sim(a,u_i)|}$$

For example, we have already found out two neighbors of active user (user 4), namely user 1 and user 2 from table 1, according to cosine similarity measure. It is necessary to predict the missing values r_{43} and r_{44} in active user vector 4. We have:

$$\bar{r}_4=1.5,\bar{r}_1\approx 1.33,\bar{r}_2\approx 1.66$$
$$sim(u_4,u_1)=1$$
$$sim(u_4,u_2)=0.8$$

It implies:

$$r_{43}=\bar{r}_4+\frac{(r_{13}-\bar{r}_1)sim(u_4,u_1)+(r_{23}-\bar{r}_2)sim(u_4,u_2)}{|sim(u_4,u_1)+sim(u_4,u_2)|}=1.5+\frac{(1-1.33)*1+(2-1.66)*0.8}{|1+0.8|}\approx 1.47$$
$$r_{44}=\bar{r}_4+\frac{(r_{14}-\bar{r}_1)sim(u_4,u_1)+(r_{24}-\bar{r}_2)sim(u_4,u_2)}{|sim(u_4,u_1)+sim(u_4,u_2)|}=1.5+\frac{(5-1.33)*1+(4-1.66)*0.8}{|1+0.8|}\approx 4.58$$

So the active user vector is u_4 = (1, 2, 1.47, 4.58) as seen in table 2 in which the missing values of active user vector are replaced by the predictive values.

	Item 1	Item 2	Item 3	Item 4
User 1	$r_{11} = 1$	$r_{12} = 2$	$r_{13} = 1$	$r_{14} = 5$
User 2	$r_{21} = 2$	$r_{22} = 1$	$r_{23} = 2$	$r_{24} = 4$
User 3	$r_{31} = 4$	$r_{32} = 1$	$r_{33} = 5$	$r_{34} = 5$
User 4	$r_{41} = 1$	$r_{42} = 2$	$\mathbf{r_{43} = 1.47}$	$\mathbf{r_{44} = 4.58}$

Table 2. Complete rating matrix

After step "computing predictive values", there is no missing value in active user vector, so the items having highest values are recommended to active user. Suppose we pre-define a threshold 2.5 so that the item whose rating value is greater than or equal to 2.5 is considered as potentially recommended item. Therefore, item 4 is recommended to user 4 because item 4 has high score (4.58) and user 4 does not rate on item 4 yet.

3. Model-based collaborative filtering

The main drawback of memory-based technique is the requirement of loading a large amount of in-line memory. The problem is serious when rating matrix becomes so huge in situation that there are extremely many persons using system. Computational resource is consumed much and system performance goes down; so system can't respond user request immediately. Model-based approach intends to solve such problems. There are four common approaches for model-based CF such as clustering, classification, latent model, Markov decision process (MDP), and matrix factorization.

3.1. Clustering CF

Clustering CF [6] is based on assumption that users in the same group have the same interest; so they rate items similarly. Therefore users are partitioned into groups called clusters which is defined as a set of similar users. Suppose each user is represented as rating vector denoted $u_i = (r_{i1}, r_{i2}, \ldots, r_{in})$. The dissimilarity measure between two users is the distance between them. We can use Minkowski distance, Euclidian distance or Manhattan distance.

$$distance_{Minkowski}(u_1, u_2) = \sqrt[q]{\sum_j (r_{1j} - r_{2j})^q}$$

$$distance_{Euclidian}(u_1, u_2) = \sqrt{\sum_j (r_{1j} - r_{2j})^2}$$

$$distance_{Manhattan}(u_1, u_2) = \sum_j |r_{1j} - r_{2j}|$$

The less $distance(u_1, u_2)$ is, the more similar u_1 and u_2 are. Clustering CF includes two steps:

1. Partitioning users into clusters and each cluster always contains rating values. For example, every cluster resulted from *k*-mean algorithm has a mean which is a rating vector like user vector.
2. The concerned user who needs to be recommended is assigned to concrete cluster and her/his ratings are the same to ratings of such cluster. Of course how to assign a user to right cluster is based on the distance between user and cluster.

So the most important step is how to partition users into clusters. There are many clustering techniques such as *k*-mean and *k*-centroid. The most popular clustering algorithm is *k*-mean algorithm [3] which includes three following steps [7, pp. 402-403]:

1. It randomly selects *k* users, each of which initially represents a cluster mean. Of course, we have *k* cluster means. Each mean is considered as the "representative" of one cluster. There are *k* clusters.
2. For each user, the distance between it and *k* cluster means are computed. Such user belongs to the cluster to which it is nearest. In other words, if user u_i belong to cluster c_v, the distance between u_i and mean m_v of cluster c_v, denoted $distance(u_i, m_v)$, is minimal over all clusters.
3. After that, the means of all clusters are re-computed. If stopping condition is met then algorithm is terminated, otherwise returning step 2.

This process is repeated until the stopping condition is met. There are two typical terminating conditions (stopping conditions) for *k*-mean algorithm:

- The *k* means are not changed. In other words, *k* clusters are not changed. This condition indicates a perfect clustering task.
- Alternatively, error criterion is less than a pre-defined threshold.

If the stopping condition is that the error criterion is less than a pre-defined threshold, the error criterion is defined as follows:

$$error = \sum_{v=1}^{k} \sum_{u_i \in c_v} distance(u_i, m_v)$$

Where c_v and m_v is cluster v and its mean, respectively. However, clustering CF encounters the problem of sparse rating matrix in which there are many missing values, which cause clustering algorithms to be imprecise. In order to solve this problem, Ungar and Foster [6, p. 3] proposed an innovative clustering CF which firstly groups items based on which users rate such items and then uses the item groups to help group users. Their method is a formal statistical model.

3.2. Classification CF

Each user is represented as rating vector $u_i = (r_{i1}, r_{i2}, \ldots, r_{in})$. Suppose every rating value r_{ij}, which is an integer, ranges from c_1 to c_m. For example, in 5-value rating system, we have c_1=1, c_2=2, c_3=3, c_4=4, and c_5=5. If user gives rating 5 to an item, she/he likes most such item. If user gives rating 1 to an item, she/he likes least such item. Each c_k is considered as a class and the set $C = \{c_1, c_2, \ldots, c_m\}$ is known as class set. From table 1, active user vector u_4 = (1, 2, ?, ?) has two missing value r_{43} and r_{44}. According to classification CF, predicting values of r_{43} and r_{44} is to find classes of r_{43} and r_{44} with suppose that there are only 5 classes $\{c_1$=1, c_2=2, c_3=3, c_4=4, c_5=5$\}$ in table 1.

A popular classification technique is naïve Bayesian method, in which user u_i belongs to class c if the posterior conditional probability of class c given user u_i is maximal [7, p. 351].

$$c = \underset{c_k \in C}{\operatorname{argmax}} P(c_k | u_i)$$

According to Bayes' rule, we have:

$$P(c_k | u_i) = \frac{P(u_i | c_k) P(c_k)}{P(u_i)}$$

Because $P(u)$ is the same for all rating values r_{ij} (s), the probability $P(c_k|u_i)$ is maximal if the product $P(u_i|c_k)P(c_k)$ is maximal. Therefore, we have:

$$c = \underset{c_k \in C}{\operatorname{argmax}} P(u_i | c_k) P(c_k)$$

Let I be the set of indices of items on which user u_i rates and so we have $I = \{j: r_{ij} \neq ?\}$. Suppose rating values r_{ij} (s) are independent given a class, we have:

$$P(u_i | c_k) = P(r_{ij}: j \in I | c_k) = \prod_{j \in I} P(r_{ij} | c_k)$$

Finally, what we need to do is to maximize the product $P(c_k) \prod_{j \in I} P(r_{ij} | c_k)$ with regard to c_k [1, p. 8]

$$c = \underset{c_k \in C}{\operatorname{argmax}} P(c_k) \prod_{j \in I} P(r_{ij} | c_k)$$

When user u_i is active user, the set I is replaced by the set $A = \{j: r_{aj} \neq ?\}$, as follows:

$$c = \underset{c_k \in C}{\operatorname{argmax}} P(c_k) \prod_{j \in A} P(r_{aj} | c_k)$$

Bayesian network [8, p. 40] is a directed acyclic graph which is composed of a set of nodes and a set of directed arcs. Each arc represents dependence between two nodes that the strong of such dependence is quantified by conditional probabilities. In context of clustering CF, the class of a user is expressed as the top-most node C [9, p. 499]. In figure 1 [9, p. 500], node r_i represents rating values of item i, which is also known as attribute of item i. For instance, naïve Bayesian method can be represented by Bayesian network.

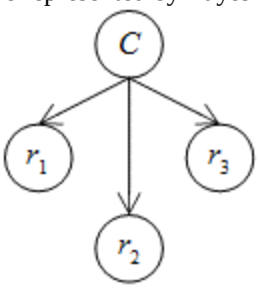

Figure 1. Bayesian network for CF

Joint probability of Bayesian network is the same to the product of probabilities in naïve Bayesian method:

$$P(c, r_1, r_2, \ldots, r_n) = P(c) \prod_{i=1}^{n} P(r_i | c)$$

What we need to do is to maximize the joint probability. However Bayesian network CF is more useful than naïve Bayesian CF because there is no assumption about independence of rating nodes r_i (s). Bayesian network

can be more complex in case that there are dependences among rating nodes r_i (s) [9, p. 500]. Such Bayesian network is called TAN network [9, p. 500]. Some arcs among nodes r_i (s) occur in TAN network, as seen in figure 2 [9, p. 500]. Please read the article "Bayesian Network Classifiers" by Nir Friedman, Dan Geiger, and Moises Goldszmidt [10] in order to comprehend Bayesian network classification.

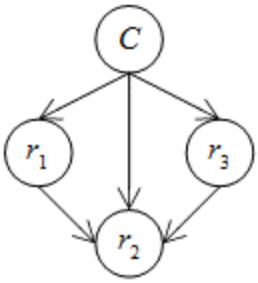

Figure 2. TAN network for CF

Some learning algorithms can be applied to specify the conditional probabilities so that the joint probability can be determined concretely. The joint probability will be more complicated and so the way to maximize it is more difficult.

3.3. Latent class model CF

Given a set of user $X = \{x_1, x_2, \ldots, x_m\}$ and a set of items $Y = \{y_1, y_2, \ldots, y_n\}$. Each observation is a pair of user/item (x, y) where $x \in X$ and $y \in Y$. According to Hofmann and Puzieha [11, p. 688], the observation (x, y) is considered as co-occurrence of user and item. It represents a preference or rating of user on item, for example, "user x likes/dislikes item y" and "users x gives rating value 5 on item y" [11, p. 688]. A latent class variable c is associated with each co-occurrence (x, y). The variable c can be a preference such as "*like*" or "*dislike*". It can be a rating value, such as 1, 2, 3, 4, and 5 in five-star rating scale [12, p. 91]. We have a set of latent class variables, $C = \{c_1, c_2, \ldots, c_k\}$. It is easy to deduce that the set of co-occurrence data $X \times Y$ is partitioned into k classes $\{c_1, c_2, \ldots, c_k\}$. The mapping $c: X \times Y \to \{c_1, c_2, \ldots, c_k\}$ is called as latent class model or aspect model developed by Hofmann and Puzieha [11, p. 689].

The problem which needs to be solved is to specify the latent class model. Namely, given a co-occurrence (x, y), how to determine which latent variable $c \in \{c_1, c_2, \ldots, c_k\}$ is most suitable to be associated with (x, y). It means that the conditional probability $P(c \mid x, y)$ must be computed. So the essence of latent class model is the probability model in which the probability distribution $P(c \mid x, y)$ need to be determined. Hofmann and Puzieha used expectation maximization (EM) algorithm to estimate such probability model. EM algorithm is performed through many iterations until stopping condition is met. According to Hofmann and Puzieha [11, p. 689], each iteration has two steps as follows:

1. The posterior probability $P(c \mid x, y)$ is computed through two parameters $P(x \mid c)$ and $P(y \mid c)$ which are specified in previous iteration.
2. The parameters $P(x \mid c)$ and $P(y \mid c)$ are updated by current estimation $P(c \mid x, y)$.

The common stopping condition is that there is no significant change in two parameters $P(x \mid c)$ and $P(y \mid c)$ for two successive iterations. It is necessary to explain how to compute the posterior probability $P(c \mid x, y)$ in step 1. According to Bayes' theorem, we have [11, p. 689]:

$$P(c|x,y) = \frac{P(c)P(x,y|c)}{\sum_{i=1}^{k} P(c_i)P(x,y|c_i)}$$

Suppose user x and item y are independent given c. The equation above is re-written [11, p. 689]:

$$P(c|x,y) = \frac{P(c)P(x|c)P(y|c)}{\sum_{i=1}^{k} P(c_i)P(x|c_i)P(y|c_i)}$$

Two probabilities $P(x \mid c)$ and $P(y \mid c)$ are considered as parameters which will be updated in step 2. In the other words, the current posterior probability $P(c \mid x, y)$ is used to calculate parameters $P(x \mid c)$ and $P(y \mid c)$ in step 2 as follows [11, p. 689]:

$$P(x|c) = \frac{\sum_y n(x,y)P(c|x,y)}{\sum_{x'}\sum_y n(x',y)P(c|x',y)}$$

$$P(y|c) = \frac{\sum_x n(x,y)P(c|x,y)}{\sum_x \sum_{y'} n(x,y')P(c|x,y')}$$

Where $n(x, y)$ is the count of co-occurrences (x, y) in rating database (rating matrix). Note, $x \in X$, $x' \in X$, $y \in Y$, and $y' \in Y$. As a result, given active user x and item y, latent class model CF will determine $P(c_i \mid x, y)$ over all $\{c_1, c_2, \ldots, c_k\}$, the predicted rating value of x on y is the class c so that $P(c \mid x, y)$ get maximal.

$$c = \underset{c_i \in C}{\operatorname{argmax}} P(c_i|x,y)$$

3.4. Markov decision process (MDP) based CF

According to Shani, Heckerman, and Brafman [13, p. 1265], recommendation can be considered as a sequential process including many stages. At each stage a list of items which is determined based on the last user's rating that is recommended to user. So recommendation task is the best *action* that recommender system must do at a concrete stage so as to satisfy user's interest. The recommendation becomes the process of making decision so as to choose the best action. The Markov decision process (MDP) based CF is proposed by Shani, Heckerman, and Brafman [13].

Suppose recommendation is the finite process having some stages. Each stage is a transaction which reflects items that user rates. Let S be a set of states and so we have $s \in S$. Let k be the number of last k rated items; so each state is denoted $s = \langle x_1, x_2, \ldots, x_k \rangle$ where x_i is a rated item [13, p. 1272]. Suppose action represents possible recommendation process. Let A be a set of actions, we have $a \in A$. The *reward* function $R(a, s)$ is used to compute the measure expressing the likeliness that action a is done given state s. The more $R(a, s)$ is, the more suitable action a is to state s. Let $T(a, s_i, s_j)$ be the transition probability from current state $s_i \in S$ to next state $s_j \in S$ given action $a \in A$. So $T(a, s_i, s_j)$ expresses the possibility that user's ratings are changed from current state to next state. A policy π is defined as the function that assigns an action to pre-assumption state at current stage.

$$\pi(s) = a \in A$$

Markov decision process (MDP) [7] is represented as a four-tuple model $\langle S, A, R, T \rangle$ [13, p. 1270] where S, A, R, and T are a set of states, a set of actions, reward function, and transition probability density, respectively. Now the essence of making decision process is to find out the optimal policy with regard to such four-tuple model. At current stage, the value function $v_\pi(s)$ is defined as the expected sum of rewards gained over the decision process when using policy π starting from state s [13, p. 1270].

$$v_\pi(s) = R(\pi(s), s) + \gamma \sum_{s_j \in S} T\big(\pi(s), s, s_j\big) v_{\pi'}\big(s_j\big)$$

Where γ is the discount factor $0 \leq \gamma \leq 1$ and $v_{\pi'}(s)$ is the value function using previous policy π'. Now the essence of making decision process is to find out the optimal policy that maximizes the value function $v_\pi(s)$ at current stage. So the policy iteration algorithm is often used to find out the optimal policy [13, p. 1271]. It includes three basic steps [13, pp. 1270-1271]:

1. The previous policy π' is initialized as a null function and the optimal policy π is initialized arbitrarily. The previous value function is initialized as $v_{\pi'}(s) = 0$ for all $s \in S$.
2. Computing value function $v_\pi(s)$ for every state s as follows:

   $$v_\pi(s) = R(\pi(s), s) + \gamma \sum_{s_j \in S} T\big(\pi(s), s, s_j\big) v_{\pi'}\big(s_j\big)$$

   Given such $v_\pi(s)$, the optimal policy π is the one that maximize value function as follows:

   $$\pi(s) = \underset{a \in A}{\operatorname{argmax}} \left\{ R(a, s) + \gamma \sum_{s_j \in S} T\big(a, s, s_j\big) v_\pi\big(s_j\big) \right\}$$

3. If $\pi = \pi'$ then algorithm is stopped and π is the final optimal policy. Otherwise set $\pi' = \pi$ and return step 2.

3.5. Matrix factorization based CF

Matrix factorization relates to techniques to analyze and take advantages of rating matrix with regard to matrix algebra. Concretely, matrix factorization based CF aims to two goals. The first goal is to reduce dimension of rating matrix [14, p. 5]. The second goal is to discover potential features under rating matrix [14, p. 5] and such features will serve a purpose of recommendation. There are some models of matrix factorization in context of CF such as Latent Semantic Analysis (LSA), Latent Dirichlet Allocation (LDA), Principle Component Analysis (PCA), and Singular Value Decomposition (SVD). This research concerns PCA and SVD.

Firstly we focuses on how to reduce dimension of rating matrix. Two serious problems in CF are data sparseness and huge rating matrix which cause low performance. When there are so many users or items, some of them don't contribute to how to predict missing value and so they become unnecessary. In other words the rating matrix has insignificant rows or columns. Dimensionality reduction aims to get rid of such redundant rows or columns so as to keep principle or important rows/columns. As result the dimension of rating matrix is reduced as much as possible. After that other CF approaches can be applied to the rating matrix whose dimension is reduced in order to make recommendation. Principle Component Analysis (PCA) is a popular technique of dimensionality reduction.

The idea of PCA is to find out the most significant components called patterns in the data population without loss of information. In context of CF, patterns are users who often rates on items or items are considered by many users. Suppose there are m users and n items and each user is represented as rating vector $u_i = (r_{i1},$

$r_{i2},\ldots, r_{in}$). Of course the rating matrix R has m rows and n columns. Let I be the set of indices of items on which user i rates and so we have $I = \{k: r_{ik} \neq ?\}$. Let J be the set of indices of items on which user j rates and so we have $J = \{k: r_{jk} \neq ?\}$. Let V be the intersection set of I and J and so we have $V = I \cap J$, which means that V is the set of indices of items on which both user i and user j rate. Let $\bar{r}_i$ and $\bar{r}_j$ be the average ratings of normal user i and user j, respectively. We have:

$$\bar{r}_i = \frac{1}{|I|}\sum_{k\in I} r_{ik}$$

$$\bar{r}_j = \frac{1}{|J|}\sum_{k\in J} r_{jk}$$

The covariance of u_i and u_j, denoted $cov(u_i, u_j)$ is defined as follows [15, p. 5]:

$$\begin{cases} \text{if } |V| \geq 2: cov(u_i, u_j) = \dfrac{\sum_{k\in V}(r_{ik} - \bar{r}_i)(r_{jk} - \bar{r}_j)}{|V| - 1} \\ \text{if } |V| < 2: cov(u_i, u_j) = 0 \end{cases}$$

The covariance $cov(u_i, u_j)$ represents the correlation relationship between u_i and u_j. If $cov(u_i, u_j)$ is positive, preferences of user i and user j are directly proportional. If $cov(u_i, u_j)$ is negative, preferences of user i and user j are inversely proportional. The covariance matrix C is composed of all covariance (s) as follows [15, pp. 7-8]:

$$C = \begin{pmatrix} cov(u_1,u_1) & cov(u_1,u_2) & \cdots & cov(u_1,u_n) \\ cov(u_2,u_1) & cov(u_2,u_2) & \cdots & cov(u_2,u_n) \\ \vdots & \vdots & \ddots & \vdots \\ cov(u_n,u_1) & cov(u_n,u_2) & \cdots & cov(u_n,u_n) \end{pmatrix}$$

Note that C is symmetric and have n rows and n columns. Matrix C is characterized by its eigenvalues and eigenvectors. Eigenvectors determine an orthogonal base of C. Each eigenvalue λ_i is corresponding to an eigenvector ε_i. Both eigenvalue λ_i and eigenvector ε_i satisfy the following equation:

$$C\varepsilon_i = \lambda_i\varepsilon_i, \forall i = 1,2,..,n$$

Eigenvalues are found out by solving the following equation:

$$|C - \lambda I| = 0$$

Where,

$$\lambda = \begin{pmatrix} \lambda_1 \\ \lambda_2 \\ \vdots \\ \lambda_n \end{pmatrix}$$

Note that $|.|$ denotes the determinant of matrix and I is identity matrix.

$$I = \begin{pmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{pmatrix}$$

Note that the larger an eigenvalue is, the more significant it is. Given an eigenvalue λ_i, the respective eigenvector is a solution of following equation:

$$(C - \lambda_i I)x = 0$$

Let p be much smaller than n ($p << n$) and let B is the matrix that is composed of k eigenvectors. So B is $n_x p$ matrix:

$$B = (\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_k) = \begin{pmatrix} \varepsilon_{11} & \varepsilon_{21} & \cdots & \varepsilon_{p1} \\ \varepsilon_{12} & \varepsilon_{22} & \cdots & \varepsilon_{p2} \\ \vdots & \vdots & \ddots & \vdots \\ \varepsilon_{1n} & \varepsilon_{2n} & \cdots & \varepsilon_{pn} \end{pmatrix}$$

User vector u_i is "compressed" as below [15, p. 16]:

$$u_i' = B^T(u_i)^T = B^{-1}(u_i)^T$$

Note that T denotes transposition operation. Because B is a base, its inverse B^{-1} is also its transposed matrix, $B^{-1} = B^T$. If any missing value occurs in u_i, it is ignored. The vector u_i is projected on B, which resulted the compressed vector u_i'. It is easy to recognize that the dimension of vector u_i' is now reduced to p with $p << n$ and B is a base of p-dimension space [15, p. 15]. The rating matrix R is "compressed" into R' as follows:

$$R' = B^T R^T = B^{-1} R^T$$

The rating matrix R that composed of n-dimension vectors u_i becomes matrix R' composed of p-dimension vectors u_i'. The components of u_i' are the most significant components. All other CF algorithms can be applied into R' instead of R so as to make recommendation. After making recommendation, original vector u_i can be recovered from u_i' with acceptable loss of information so as to retrieve actual rating values [15, p. 19].

$$(u_i)^T \approx B^{-1}u_i'$$

Recall that the second goal of matrix factorization based CF is to discover potential features under rating matrix [14, p. 5] and such features will serve a purpose of recommendation. Later on we will know that dimensionality reduction is similar to discovering potential features. In fact, significant components resulted from PCA technique are latent features of items. Now we focuses on how to extract features matrices of users and items by Singular Value Decomposition (SVD). Given original rating matrix R, the SVD technique finds user feature matrix U and item features matrix V such that [14, p. 4]:

$$R = U\Lambda V^T$$

Where T denotes vector and matrix transposition operation. If R is m_xn matrix in which there are m users and n items then, Λ is eigenvalue matrix and it is diagonal p_xp matrix as follows:

$$\Lambda = \begin{pmatrix} \sqrt{\lambda_1} & 0 & \dots & 0 \\ 0 & \sqrt{\lambda_2} & \dots & 0 \\ & & \ddots & \vdots \\ \vdots & \vdots & & \\ 0 & 0 & \dots & \sqrt{\lambda_p} \end{pmatrix}$$

There are p eigenvalues λ_i (s) which are solutions of following equations whose unknown variable is λ.

$$\begin{cases} |RR^T - \lambda I_m| = 0 \\ |R^T R - \lambda I_n| = 0 \end{cases}$$

Where I_m and I_n are m_xm identity matrix and n_xn identity matrix, respectively. Note that the notation |.| denotes determinant of matrix. In identity matrix, diagonal elements are 1 and remaining elements are 0. Following is 3_x3 identity matrix.

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

There are p eigenvectors u_i' corresponding to r eigenvalues, which are solutions of following equation whose unknown variable is x.

$$(RR^T - \lambda_i I_m)x = 0$$

There are p eigenvectors v_j' corresponding to r eigenvalues, which are solutions of following equation whose unknown variable is x.

$$(R^T R - \lambda_i I_n)x = 0$$

The user feature matrix U and the item feature matrix V are composed of r eigenvectors u_i and r eigenvectors v_i as follows:

$$U_{m_xp} = (u_1', u_2', \dots, u_r') = \begin{pmatrix} u_{11} & u_{21} & \dots & u_{p1} \\ u_{12} & u_{22} & \dots & u_{p2} \\ \vdots & \vdots & \ddots & \vdots \\ u_{1m} & u_{2m} & \dots & u_{pm} \end{pmatrix}$$

$$V_{n_xp} = (v_1', v_2', \dots, v_r') = \begin{pmatrix} v_{11} & v_{21} & \dots & v_{p1} \\ v_{12} & v_{22} & \dots & v_{p2} \\ \vdots & \vdots & \ddots & \vdots \\ v_{1n} & v_{2n} & \dots & v_{pn} \end{pmatrix}$$

Where $u_i' = (u_{i1}, u_{i2}, \dots, u_{im})^T$ and $v_j' = (v_{j1}, v_{j2}, \dots, v_{jn})^T$ are column normalized eigenvectors for users and items, respectively. They are mutually orthogonal. Let $u_i = (u_{1i}, u_{2i}, \dots, u_{pi})^T$ and $v_j = (v_{1j}, v_{2j}, \dots, v_{pj})^T$ be feature vectors for users and items, respectively. We also have:

$$U_{m_xp} = \begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_m \end{pmatrix} = \begin{pmatrix} u_{11} & u_{21} & \dots & u_{p1} \\ u_{12} & u_{22} & \dots & u_{p2} \\ \vdots & \vdots & \ddots & \vdots \\ u_{1m} & u_{2m} & \dots & u_{pm} \end{pmatrix}$$

$$V_{n_xp} = \begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_m \end{pmatrix} = \begin{pmatrix} v_{11} & v_{21} & \dots & v_{p1} \\ v_{12} & v_{22} & \dots & v_{p2} \\ \vdots & \vdots & \ddots & \vdots \\ v_{1n} & v_{2n} & \dots & v_{pn} \end{pmatrix}$$

The rating r_{ij} that user i rates on item j is recovered from user feature vector u_i, item feature vector v_j, and eigenvalue matrix Λ as follows:

$$r_{ij} = (u_i)^T \Lambda v_j = (u_{1i}, u_{2i}, \dots, u_{pi})\Lambda \begin{pmatrix} v_{1j} \\ v_{2j} \\ \vdots \\ v_{pj} \end{pmatrix}$$

The main problem of SVD is to solve the equations $|RR^T - \lambda I_m| = 0$ and $|R^T R - \lambda I_n| = 0$ so as to find our eigenvalues, which in turn determines features matrices U and V but it is impossible to solve such equations

because *R* is sparse matrix which has many missing values. In [15, p. 3], the solution is to minimize the loss function so as to determine *U* and *V*. The loss function is defined as follows [15, p. 3]:

$$l(U,V) = \frac{1}{2}\sum_{i}\sum_{j}\left(r_{ij} - (u_i)^T v_j\right)^2 + \frac{k_u}{2}\sum_{i=1}^{m}|u_i|^2 + \frac{k_v}{2}\sum_{j=1}^{n}|v_j|^2$$

Where,

$$|u_i| = \sqrt{{u_{1i}}^2 + {u_{2i}}^2 + \cdots + {u_{pi}}^2}$$

$$|v_j| = \sqrt{{v_{1j}}^2 + {v_{2j}}^2 + \cdots + {v_{pj}}^2}$$

Note, r_{ij} (s) are only rating values that user *i* rated on item *j* in rating matrix. The k_u and k_v are regularization coefficients for preventing over-fitting and they are real numbers pre-defined from 0 to 1. The loss function has two matrix variables *U* and *V* but we can consider that it has *m* variable u_i and *n* variables v_j. Determining *U* and *V* is to find out u_i^* and v_j^* that minimize the loss function, which becomes an optimization problem:

$$\left(u_i^*, v_j^*\right) = \underset{u_i, v_j}{\operatorname{argmin}}\, l(U,V)$$

When u_i^* and v_j^* are found out, the predicted value (estimated value) of r_{ij} is:

$$\hat{r}_{ij} = (u_i^*)^T v_j^*$$

It is expected that $\hat{r}_{ij}$ is approximated to r_{ij}. The gradients of $l(U, V)$ with regard to u_i and v_j are [15, p. 4]:

$$\frac{\partial l(U,V)}{\partial u_i} = -\sum_{j}\left(r_{ij} - (u_i)^T v_j\right)^2 \left(v_j\right)^T + k_u (u_i)^T$$

$$\frac{\partial l(U,V)}{\partial v_j} = -\sum_{i}\left(r_{ij} - (u_i)^T v_j\right)^2 (u_i)^T + k_v \left(v_j\right)^T$$

Note, gradients, which are partial derivatives, are row vectors whereas u_i and v_j are columns vectors. Gradient descent algorithm is the common method to solve the optimization problem, includes three steps as follows:

1. Feature vectors u_i and v_j are initialized arbitrarily. Regularization coefficients k_u and k_v are set as real numbers in interval [0, 1]. A learning rate *t* is set as a real number in interval [0, 1].
2. Calculating both gradients $\frac{\partial l(U,V)}{\partial u_i}$ and $\frac{\partial l(U,V)}{\partial v_j}$. Later on, re-calculating u_i and u_j as follows:

$$u_i = u_i - t\left(\frac{\partial l(U,V)}{\partial u_i}\right)^T$$

$$v_j = v_j - t\left(\frac{\partial l(U,V)}{\partial v_j}\right)^T$$

3. If both gradients $\frac{\partial l(U,V)}{\partial u_i}$ and $\frac{\partial l(U,V)}{\partial v_j}$ are approximated to 0 then, the algorithm stops. Otherwise, going back step 2.

The final u_i and v_j resulted from the gradient descent algorithm are u_i^* and v_j^* that minimize the loss function, which means that feature matrices *U* and *V* are determined.

4. A proposed model-based approach complying with statistical method

We also give a new idea for model-based approach so as to gain high accuracy and solve the problem of sparse matrix by applying evidence-based inference techniques. The proposed model-based approach complies with statistical method which combines Expectation Maximization (EM) algorithm and Bayesian network. Firstly, EM fills in missing data existing in sparse matrix so as to achieve full matrix. Later Bayesian network is built from complete matrix. The prediction for recommendation is based on inference mechanism of Bayesian network. Thus user's old ratings are considered as evidences which are provided to Bayesian network in order to predict her/his new ratings. So EM algorithm and Bayesian network are important issues; especially, Bayesian network is core model in our approach.

EM algorithm estimates parameters of probability model. EM is typically used to compute maximum likelihood estimation given incomplete samples. Because the data set (rating matrix) is comprised of several distinct populations, EM will be applied into the mixture model. The advantage of Bayesian network is evidence-based inference with high reliability but we must solve two problems:

- How to represent each node in network because there is a boom of combinations of conditional probabilities when the number of values of each node is not determined carefully. For example, if a node has *n* values and *m* parent nodes then it requires n^m conditional probabilities which are stored as entries in network. The huge number of entries is the main cause of low performance.

- How to build Bayesian network from item-based matrix. It is ongoing research. We intend to apply entropy algorithm into learning Bayesian network structure.

In general the proposed approach includes 4 steps:

Step 1: Transposing user-based rating matrix to item-based rating matrix which is resource provided for EM and Bayesian network. Table 3 is an example of transposition of user-based matrix to item-based matrix.

	Item 1	Item 2	Item 3
User 1	$r_{11} = 1$	$r_{12} = 3$	$r_{13} =$ **?**
User 2	$r_{21} = 3$	$r_{22} =$ **?**	$r_{23} = 5$
User 3	$r_{31} = 4$	$r_{32} = 2$	$r_{33} = 1$
User 4	$r_{41} =$ **?**	$r_{42} =$ **?**	$r_{43} = 3$

→

	User 1	User 2	User 3	User 4
Item 1	$r_{11} = 1$	$r_{21} = 3$	$r_{31} = 4$	$r_{41} =$ **?**
Item 2	$r_{12} = 3$	$r_{22} =$ **?**	$r_{32} = 2$	$r_{42} =$ **?**
Item 3	$r_{13} =$ **?**	$r_{23} = 5$	$r_{33} = 1$	$r_{43} = 3$

Table 3. Transposing user-based matrix to item-based matrix

Step 2: EM algorithm is used to fill in missing values so as to gain full rating matrix. EM has many iterations. Each iteration consists of two steps such as E(xpectation)-step and M(aximization)-step..

- E-step: missing data are estimated given observed data and current estimate of model parameters.
- M-step: likelihood function is maximized under assumption that missing data are known so as to improve the model parameters.

Suppose table 4 is an example of full item-based rating matrix in which there is no missing value because missing values represented by question masks in table 3 are now filled by estimated values such as $r_{41} = 3$, $r_{22} = 3$, $r_{42} = 2$, and $r_{13} = 2$.

	User 1	User 2	User 3	User 4
Item 1	$r_{11} = 1$	$r_{21} = 3$	$r_{31} = 4$	$r_{41} =$ **3**
Item 2	$r_{12} = 3$	$r_{22} =$ **3**	$r_{32} = 2$	$r_{42} =$ **2**
Item 3	$r_{13} =$ **2**	$r_{23} = 5$	$r_{33} = 1$	$r_{43} = 3$

Table 4. Full item-based rating matrix

Step 3: Building the Bayesian network from the full item-based rating matrix in step 2 by using some learning algorithms. Suppose each node represents an item and has five values {1, 2, 3, 4, 5}. According to a sample Bayesian network shown in figure 3, node I_1 representing item 1 has 5 prior probabilities because it has no parent. Node I_2 representing item 2 has 5^5 conditional probabilities because it has parent node I_1. Similarly, node I_3 representing item 3 has 5^5 conditional probabilities because it has parent node I_1 too.

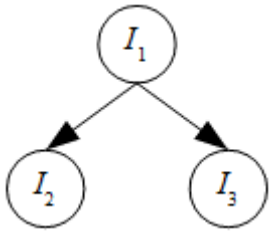

Figure 3. A sample Bayesian network with 3 item nodes

Step 4: Suppose that the active user only rated on item 1 among three items, we have $U = \{I_1=5, I_2=?, I_3=?\}$. The problem which needs to be solved is how to estimate values of I_2 and I_3. Therefore, I_1 becomes evidence node (I_1=5) whereas I_2 and I_3 become hidden nodes or hypothesis nodes. Hidden nodes represent potentially recommended items. In figure 4, hidden nodes I_2 and I_3 is shaded.

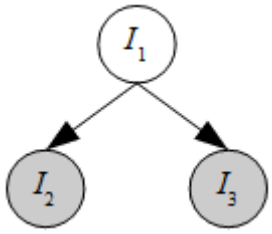

Figure 4. A sample Bayesian network with evidence node I_1 and hidden nodes I_2, I_3

We must determine posterior probabilities of I_2 and I_3 by using inference mechanism based on evidence in Bayesian network. For example, posterior probabilities of I_2 are $P(I_2=1 \mid I_1=5) = 0.1$, $P(I_2=2 \mid I_1=5) = 0.2$, $P(I_2=3 \mid I_1=5) = 0.2$, $P(I_2=4 \mid I_1=5) = 0.1$, and $P(I_2=5 \mid I_1=5) = 0.4$. Because $P(I_2=5 \mid I_1=5) = 0.4$ is maximal, the estimated value of I_2 is 5.

Similarly, we continue to calculate posterior probability of I_3. Suppose we have $P(I_3=1 \mid I_1=5) = 0.2$, $P(I_3=2 \mid I_1=5) = 0.1$, $P(I_3=3 \mid I_1=5) = 0.2$, $P(I_3=4 \mid I_1=5) = 0.3$, and $P(I_3=5 \mid I_1=5) = 0.2$. Because $P(I_3=4 \mid I_1=5) = 0.3$ is maximal, the estimated value of I_3 is 4. Finally, item 2 is highly recommended to active user because the estimated value of I_2 (=5) is larger than the estimated value of I_3 (=4). Moreover, given I_1=5, the posterior probability of I_2=5, which is $P(I_2=5 \mid I_1=5) = 0.4$, is larger than the posterior probability of I_3=4, which is $P(I_3=4 \mid I_1=5) = 0.3$.

5. Conclusions

CF has two common approaches such as memory-based and model-based. The model-based approach applies statistics method and machine learning technique to mining rating matrix. Although the model-based approach doesn't give result which is as precise as the combination of CF and CBF approaches, it can solve the problem of huge database and sparse matrix. We also give a new idea for model-based approach so as to gain high accuracy and solve the problem of sparse matrix by applying evidence-based inference techniques. It is the potential approach because it can take advantage of power mathematical tools and artificial intelligent methods such as statistics and machine learning. It is complicated but interesting and so, the research is ongoing.

References

[1] X. Su and T. M. Khoshgoftaar, "A Survey of Collaborative Filtering Techniques," *Advances in Artificial Intelligence,* vol. 2009, 2009.

[2] P. Melville, R. J. Mooney and R. Nagarajan, "Content-Boosted Collaborative Filtering," in *Proceedings of the SIGIR-2001 Workshop on Recommender Systems*, New Orleans, 2001.

[3] R. D. Torres Júnior, "Combining Collaborative and Content-based Filtering to Recommend Research Paper," Universidade Federal do Rio Grande do Sul, Porto Alegre, 2004.

[4] B. Sarwar, G. Karypis, J. Konstan and J. Riedl, "Item-based Collaborative Filtering Recommendation Algorithms," in *Proceedings of the 10th international conference on World Wide Web*, Hong Kong, 2001.

[5] H. Ma, I. King and M. R. Lyu, "Effective Missing Data Prediction for Collaborative Filtering," in *SIGIR '07 Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, Amsterdam, 2007.

[6] L. H. Ungar and D. P. Foster, "Clustering Methods for Collaborative Filtering," in *AAAI Workshop on Recommender Systems*, Madison, 1998.

[7] J. Han and M. Kamber, Data Mining: Concepts and Techniques, 2nd Edition ed., J. Gray, Ed., San Francisco, CA: Morgan Kaufmann Publishers, Elsevier, 2006, p. 743.

[8] R. E. Neapolitan, Learning Bayesian Networks, Upper Saddle River, New Jersey: Prentice Hall, 2003, p. 674.

[9] X. Su and T. M. Khoshgoftaar, "Collaborative Filtering for Multi-class Data Using Belief Nets Algorithms," in *The 18th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'06)*, Arlington, VA, USA, 2006.

[10] N. Friedman, D. Geiger and M. Goldszmidt, "Bayesian Network Classifiers," *Machine Learning*, vol. 29, no. 2-3, pp. 131-163, November 1997.

[11] T. Hofmann and J. Puzieha, "Latent Class Models for Collaborative Filtering," in *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI '99)*, San Francisco, CA, USA, 1999.

[12] T. Hofmann, "Latent Semantic Models for Collaborative Filtering," *ACM Transactions on Information Systems (TOIS)*, vol. 22, no. 1, pp. 89-115, January 2004.

[13] G. Shani, D. Heckerman and R. I. Brafman, "An MDP-based Recommender System," *Journal of Machine Learning Research*, vol. 6, no. 2005, pp. 1265-1295, September 2005.

[14] B. M. Sarwar, G. Karypis, J. A. Konstan and J. T. Riedl, "Application of Dimensionality Reduction in Recommender System - A Case Study," in *ACM WEBKDD Workshop*, 2000.

[15] L. I. Smith, "A tutorial on Principal Components Analysis," Cornell University, 2002.

[16] C.-C. Ma, "A Guide to Singular Value Decomposition for Collaborative Filtering," National Taiwan University, Taipei, Taiwan, 2008.