**A Guide Map To Collaborative Filtering**

**Loc Nguyen**

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

**Phung Do**

Faculty of Information System, University of Information Technology, Ho Chi Minh city, Vietnam

**Abstract**

Collaborative filtering (CF) is the popular algorithm for recommendation. Therefore items which are recommended to users are determined by surveying their communities. CF has good perspective because it can cast off the 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 in-line memory database. This algorithm is simple but encounters the problem of huge data. Model-based algorithm tries to compress huge database into a model and perform 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 algorithm. 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.

**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, newspaper, etc. 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 basic algorithms applied in recommendation system: content-based filtering (CBF) and collaborative filtering (CF) [1]:

* CBF recommends an item to a user if such item is similar to other items that he likes much in the past (his rating for such item is high). Note that each item has contents which are properties and so all items compose the matrix so-called item content matrix.
* CF recommends an item to a user if his neighbors (other users that 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 the matrix so-called rating matrix.

Both of them (CBF & 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; this is the strong point of CBF. However CBF doesn’t tend towards community like CF. As such items “are hided under” user community, CBF has no ability to discover implicit items that user may like. 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 the content of items. That CF only works on users’ ratings on items is strong point of CF because CF doesn’t encounter how to analyze the rich content items. But it is also weak point because CF can do unexpected recommendation in some situations in which recommended 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. This approach 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 approach improves the precision of prediction but it takes much time for two stages; 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 approach requires both item content matrix and rating matrix.
* Compressing rating matrix into representative model which are used to predict missing values for recommendation. This is model-based for CF. Note that CF has two common approaches such as memory-based and model-based. The model-base approach applies statistics method and machine learning technique 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 though 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 approach is discussed carefully in section 3. We propose an idea for the model-based CF algorithm in section 4. Section 5 is the conclusion.

**2. Memory-based collaborative filtering**

Memory-based CF [1] algorithms use the entire or a sample of the 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, 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 the rating which user gave to item. Each row represents a user vector or rating vector that models a user; so these vectors are considered as user profiles. The vector of active user is called as active user vector.

|  |  |  |  |
| --- | --- | --- | --- |
|  | *item1* | *item2* | *item3* |
| *user1* | r11 = 1 | r12 = 2 | r13 = 1 |
| *user2* | r21 = 2 | r22 = 1 | r23 = 2 |
| *user3* | r31 = 4 | r32 = 1 | r33 = 5 |
| *user4* | r41 = 1 | r42= 2 | r43 = ? (missing) |

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

Let  and  be the normal user vector *i* and the active user *a*, respectively where *rij* is the rating of user *i* to item *j*.

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 need solving is to predict missing values of active user vector; later the items having the highest values are recommended to active user. There are two steps in process of predicting missing values:

* Finding out nearest neighbors of active user
* Computing predictive values (or predictive ratings)

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 user are. Given a threshold, users that the similarities between them and active user are equal or larger than this threshold are considered as nearest neighbors of active user. There are two ways to compute this similarity:

* Using cosine similarity measure
* Using correlation coefficient

The cosine similarity measure of two users is the cosine of the angle between two user vectors.



Where the sign “●” denotes scalar product of two vectors.

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 measure of active user (user *4*) and users *1, 2, 3*  in table *1* is:







Given a threshold *0.5* it is concluded that user *1* and user *2* are similar to user *4* than user *3* is.

Correlation coefficient which is the concept in statistics is also used to specify the similarity of two vectors. Suppose *rij* and *raj* denote the ratings of user *i* and active user *a* to item *j*, respectively. Let  and  be the average ratings of normal user *i* and active user *a*, respectively. The correlation coefficient is defined as below:



If *raj* is a missing value, it is set to be *0*. The range of correlation coefficient is from *0* 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, wee need to compute the correlation coefficient between active user (user *4*) and users *1, 2, 3*  in table *1*.

, , , 







Given a threshold *0.5* it is concluded that user *1* is very similar to user 4.

**2.2. Computing predictive values**

A predictive value or predictive rating is the value that replacing a missing value in active user vector. Suppose we have *m* nearest neighbors of active user are determined form the first step “Finding out nearest neighbors of active user”. Let *sim*() is the similarity between normal user *i* and active user *a*. Let *raj* be the predictive value for item *j* of active user vector.



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 value *r43* in active user vector *4*.

, , 

*sim*(*user4, user1*) = *0.9*

*sim*(*user4, user2*) = *0.6*



So the active user vector is =(*1, 2, 1*)

|  |  |  |  |
| --- | --- | --- | --- |
|  | *item1* | *item2* | *item3* |
| *user1* | r11 = 1 | r12 = 2 | r13 = 1 |
| *user2* | r21 = 2 | r22 = 1 | r23 = 2 |
| *user3* | r31 = 4 | r32 = 1 | r33 = 5 |
| *user4* | r41 = 1 | r42= 2 | **r43 = 1** |

**Table 2**. Rating matrix in which the missing value of active user vector is replaced by predictive value.

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. For example, item *2* is recommended to active user (user *4*) because the active user vector is = (*1, 2, 1*) and item *2* has the highest score.

**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 require immediately. Model-based approach intends to solve such problems. There are four approaches for model-based CF:

* Clustering model
* Latent model
* Markov decision process (MDP) model
* Matrix factorization model

**3.1. Clustering model based CF**

The clustering CF [2], [3] is based on the assumption that users in the same group have the same interest; so they rate items similarly. Therefore users are partitioned into groups so-called clusters which is defined as a set of similar users. Suppose each user is represented as rating vector denoted *ui*=(*ri1, ri2,…, rin*). The similarity measure between two users is the distance between them. We can use Minkowski distance, Euclidian distance or Manhattan distance.

, 



The less *distance*(*u1, u2*) is, the more similar *u1* and *u2* are. Clustering CF algorithm includes two steps:

1. Partitioning users into clusters or groups and making prediction for rating values in each cluster.
2. The concerned user who needs to be recommended is assigned to concrete cluster and her/his ratings are the same to the 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 two clustering techniques:

* Using clustering algorithms such as k-means, k-centroids…
* Using Bayesian classifier

**Clustering algorithm**

The most popular clustering algorithm is *k*-means algorithm [3] which including three following steps:

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 remaining user, the distance between it and *k* cluster means are computed. Such user model belongs to the cluster which it is most similar to; it means that if user model *ui* belong to cluster *cj*, the distance measure *distance*(*ui, cj*) is minimal.
3. After that, the means of all clusters are re-computed. If stopping condition is met then algorithm is terminated, otherwise returning step *1*.

This process is repeated until the stopping condition is met. For example, the stopping condition is that the square-error criterion is less than a pre-defined threshold. The square-error criterion is defined as below:



Where *ci* and *mi* is cluster *i* and its mean, respectively.

**Bayesian classifier**

According to Bayesian approach [1], there are two main algorithms for classifying users into groups:

* Naïve Bayesian classification [1] [3]
* Bayesian network classification [4] [5]

Suppose each user is represented as rating vector *u*=(*r1, r2,…, rn*) and let R be rating matrix composed of user vectors; . Give a set of *m* classes (clusters) C = {*c1, c2,…, cm*}, it is necessary to determine which class user u belongs to. Namely user *u* belongs to class *ci* if the posterior conditional probability of class *ci* given user *u* denoted *Pr*(*ci* | *u*) is maximal.



So what we need to do is to find out the class *ci* whose conditional probability *Pr*(*ci* | *u*) is maximal. Such class *ci* is the cluster of user *u*.According to Baysian’s rule:



Because *Pr*(*u*) is the same for all rating values *rk*, *Pr*(*ci*|*u*) is maximal if the product *Pr*(*ci*)\**Pr*(*u|ci*) is maximal. Suppose rating values *rk* are independent given a class, we have:



Finally, what we need to do is to maximize the product  with regard to *ci*.

Bayesian network [4] [5] is the directed acyclic graph which is composed of a set of nodes and a set of directed arc. Each arc represents the dependence between two nodes that the strong of such dependence is quantified by conditional probability. In context of clustering model based CF, the class of user is expressed as the top-most node *C*. For instance the naïve Bayesian method can be represented as Bayesian network.

The joint probability of network is the same to the product of probabilities in naïve Bayesian:



What we need to do is to maximize joint probability. However the Bayesian network CF is more useful than naïve Bayesian because there is no assumption about the independence of rating values *rk*. The network can be more complex in case that there are dependences between rating node *rk*. Some arcs among node *rk* occur in below network.

In situation that some Bayesian network learning algorithm 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.

**3.2. Latent class CF**

Given a set of user *X* = {*x1, x2,…, xm*} and a set of items *Y* = {*y1, y2,…, yn*}. Each observation is a pair of user/item (*x, y*) where and . The observation (*x, y*) is considered as the co-occurrence of user and item. It represents the preference or rating of user on item, for example “user *x* like/dislikes item *y*”. A latent class variable *c* is associated to each co-occurrence (*x, y*). We have a set of latent class variables . It is easy to deduce that the set of co-occurrence data is partitioned into *k* classes . The mapping  is called as latent class model or aspect model [6].

The problem which needs to be solved is how to specify the latent class model. Namely, given a co-occurrence (*x, y)*, how to determine which latent variable  is the most suitable to be associated to (*x, y)*. It means that the conditional probability *Pr*(*c | x, y*) must be computed. So the essence of latent model is the probability model in which the probability distribution *Pr*(*c | x, y*) need to be determined. The Expectation Maximization (EM) is often used to estimate such probability model. EM algorithm is performed through much iteration until stopping condition is met. Each iteration has two steps:

1. Step 1: The posterior probability *Pr*(*c*|*x,y*) is computed through parameters: *Pr*(*x|c,y*) and *Pr*(*y|c,x*) which are specified in previous iteration
2. Step 2: The parameters *Pr*(*x|c,y*) and *Pr*(*y|c,x*) are updated by current estimation *Pr*(*c*|*x,y*)

**Step 1:** computing the posterior probability *Pr*(*c*|*x,y*)

According to Bayes’ theorem, we have:



Suppose that user *x* and item *y* are independent given c. The formulation is re-written as below:



Note that two probabilities *Pr*(*x| c*) and *Pr*(*y| c*) are considered as parameters which will be updated in step 2.

**Step 2:** updating parameters *Pr*(*x| c*) and *Pr*(*y| c*)

Applying Bayes’ theorem again, we have:





Where *n*(*x,y*) is the count of the co-occurrence (*x,y*) in database (rating matrix).

**3.3. Dimensionality reduction**

Two serious problems in CF are data sparsity and the 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. Because this approach pays attention to analyzing matrix – a subject of linear algebra, it can be called as matrix factorization approach. There are two well-known dimensionality algorithms:

* Singular Value Decomposition (SVD)
* Principle Component Analysis (PCA)

**PCA**

The idea of PCA is to find out the most significant components so-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, each user denoted rating vector *ui =* (*ri1, ui2,…, rin*). Form *m* user vector, the covariance matrix *C* is computed, each element *c*ij  represents the variance of user *i* and user *j*. There are some special cases as below:

* Preferences of user *i* and user *j* are not related: *cij=0*
* Preferences of user *i* and user *j* are contrary: *cij=-1*
* Preferences of user *i* and user *j* are the same: *cij=1*

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

 where 

Eigenvalues are found out by solving the equation:



Where |.| denotes the determinant of matrix and *I* denotes the identity matrix having the same order to *C* and represents . The number *k* of eigenvalues is much smaller than *n* items*.* Note that the larger an eigenvalue is, the more significant it is. Let *E* is the matrix that is composed of *k* eigenvectors*,* user vector *u* is “compressed” as below:

 where is the mean of *u*.

The vector *u*’ is projected on base *E*. The reason why vector *u* is subtracted by before it is multiplied by *A* is to adjust *u* to its mean. It is easy to recognize that the dimension of vector *u* which is *n* is now reduced to *k* (). The rating matrix *R* that composed of *n-dimension* vectors *u* becomes matrix *R’* composed of *k-dimension* vectors *u’*. The components of *u’* are the most significant components. All other CF algorithms can be applied into *R’* instead of *R*. Moreover original vector *u* can be recovered from *u’* with the acceptable loss of information.



**3.4. MDP-based CF**

Recommendation can be considered as the sequential process including many stages. At each stage a list of items which is determined based on the last user’s rating is recommended to user. So recommendation task is the best **action** that recommender system must do at concrete stage so as to satisfy user’s interest. The recommendation becomes the process of making decision so as to choose the best action.

Suppose recommendation is the finite process having some stages. Each stage is transaction which reflects items that user rates. Let *k* be the number of last *k* rated items; so each state is denoted *s* = where *xi* is the rated item.

Let *S* is a set of states, we have s*S*. Suppose action represent possible recommendation process. Let *A* be a set of actions, we have *a**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*(*si, sj, a*) be the transition probability from current state  to next state  Given action *a*. So *T*(*si, sj, a*) expresses the possibility that user’s ratings are changed from current state to next state.

A policy *P* is defined as the function that assigns an action to pre-assumption state.



The optimal policy *P*(*s*) is the policy (action) that maximizes the reward value . Markov decision process (MDP) [7] is represented as four-tuple model  where *S, A, R, 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 regards to such four-tuple model. Suppose there are *n* stages, the value function is defined as the maximal expected sum of rewards gained over the process which starts from state *s* and stops after *n* stages.



Where  is the discount factor . The action *a* which maximizes rewards is considered as a optimal policy *P*(*s*) *= a.* So the policy iteration algorithm is often used to find out the optimal policy. It includes three basic steps:

1. Initializing the policy *P* = *P0*
2. Computing value function *vs*(*n*) for every state *s*. Each *vs*(*n*) has respective action *a* which maximizes sum of rewards. Therefore the optimal policy *P’* is set to be *a*; so *P’*(*s*)=*a*
3. If *P’ = P* then algorithm is stopped and *P* is final optimal policy. Otherwise set *P* = *P’* and return step 2.

**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. A proposed model-based approach complying with statistical method which combine Expectation Maximization (EM) and Bayesian network. EM fills in missing data in sparse matrix to be complete matrix. Bayesian network is built from complete matrix.

* The EM algorithm estimates the parameters of a probability model. EM is typically used to compute maximum likelihood estimates given incomplete samples. Because there is a reason to believe that a data set is comprised of several distinct populations, a mixture model can be used.
* The advantage of Bayesian network is probabilistic inference. However, we must solve two problems: how to perform each node in network and build the Bayesian network from the item-based matrix. Furthermore, we also ongoing research about algorithms for building Bayesian network from entropy.

4-step algorithm which combine EM and Bayesian network:

**Step 1:** Transposinguser-based matrix to item-based matrix

|  |  |  |  |
| --- | --- | --- | --- |
|  | *item1* | *item2* | *item3* |
| *user1* | r11 = 1 | r12 = 3 | r13 = **?** |
| *user2* | r21 = 3 | r22 = **?** | r23 = 5 |
| *user3* | r31 = 4 | r32 = 2 | r33 = 1 |
| *user4* | r41 = **?** | r42 = **?** | r43 = 3 |

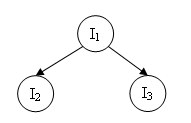
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | *user1* | *user2* | *user3* | *user4* |
| *item1* | r11 = 1 | r21 = 3 | r31 = 4 | r41 = **?** |
| *item2* | r12 = 3 | r22 = **?** | r32 = 2 | r42 = **?** |
| *item3* | r13 = **?** | r23 = 5 | r33 = 1 | r43 = 3 |

**Step 2:** The EM algorithm is an efficient iterative procedure to compute the Maximum Likelihood (ML) estimate in the presence of missing or hidden data. Each iteration of the EM algorithm consists of two processes: The E-step and the M-step.

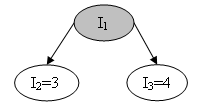
* In the expection, or E-step: the missing data are estimated given the observed data and current estimate of the model parameters.
* In the maximization, or M-step: the likelihood function is maximized under the assumption that the missing data are known.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | *user1* | *user2* | *user3* | *user4* |
| *item1* | r11 = 1 | r21 = 3 | r31 = 4 | r41 = **3** |
| *item2* | r12 = 3 | r22 = **3** | r32 = 2 | r42 = **2** |
| *item3* | r13 = **2** | r23 = 5 | r33 = 1 | r43 = 3 |

**Step 3:** Building the Bayesian network from the item-based matrix in step 2, using Entropy. Each node have five values {1, 2, 3, 4, 5} with probability.

****

**Step 4:** Suppose thatactive user U = (*I1=?, I2=3, I3=4*), the problem which needs to be solved is how to estimate value of item I1. Using inference mechanism based on evidence in Bayesian network to determine post probability of I1. For example: the post probabilities of I1 are: Pr(I1=1), Pr(I1=2), Pr(I1=3), Pr(I1=4), Pr(I1=5). Estimate value of I1 is value which the post probability is maximal.



**5. Conclusion**

Phung evaluates both memory-based and model-based and stresses on that model-based is the trend of filtering method because there are two reasons:

* its strong points (summarize its strong points)
* it is the potential approach because it can take advantage of power mathematical tools and artificial intelligent methods such as statistics, machine learning… Then Phung gives everyone the sufficient reasons for applying statistical technique like EM, Bayesian network into our algorithm.

**Reference**

1. [Su, Khoshgoftaar 2009]. Xiaoyuan Su and Taghi M. Khoshgoftaar. A Survey of Collaborative Filtering Techniques. Hindawi Publishing Corporation, Advances in Artificial Intelligence, Volume 2009, Article ID 421425, 19 pages, doi:10.1155/2009/421425.
2. [Ungar, Foster 1998]. L. H. Ungar and D. P. Foster. Clustering Methods for Collaborative Filtering. Proceedings of the Workshop on Recommendation Systems, AAAI Press, 1998.
3. [Han, Kamber 2006]. Jiawei Han and Michelline Kamber. Data Mining: Concepts and Techniques. Second Edition. © 2006 by Elsevier Inc.
4. [Breese, Heckerman, Kadie 1998]. John S. Breese, David Heckerman and Carl Kadie. Empirical Analysis of Predictive Algorithms for Collaborative Filtering. Technical Report MSR-TR-1998. Microsoft Research, Microsoft Corporation, One Microsoft Way, Redmon, WA 98052.
5. [Heckerman, Chickering, Meek, Rougnthwaite 2000]. David Heckerman, David Maxwell Chickering, Christopher Meek, Robert Rougnthwaite and Carl Kadie. Dependecy Networks for Inference, Collaborative Filtering and Data Visualization. Journal of Machine Learning Research 1 (2000) 49-75.
6. [Hofmann 2004]. Thomas Hofmann. Latent Semantic Models for Collaborative Filtering. ACM Transactionson Information Systems, Vol.22, No.1, January 2004, Pages 89-115.
7. [Shani, Heckerman, Brafman 2005]. G. Shani, D. Heckerman, and R. I. Brafman. An MDP-based Recommender System. Journal of Machine Learning Research, vol. 6, pp. 1265-1295, 2005.