Extended latent class models for collaborative recommendation

Cheung Ching Ching

IEEE Transactions on Systems, Man, and Cybernetics

Extended Latent Class Models for Collaborative Recommendation

Kwok-Wai Cheung, Kwok-Ching Tsui and Jiming Liu

***Abstract*— With the advent of the WWW, providing just-in-time personalized product recommendations to customers becomes possible. Collaborative recommender systems utilize the correlation between customer preference ratings to identify “like-minded” customers and predict their product preference. One factor determining the success of the recommender systems is the prediction accuracy, which in many cases is limited by lacking adequate ratings (the sparsity problem). Recently, the use of latent class model (LCM) has been proposed to alleviate this problem. In this paper, we first study how the LCM can be extended to handle customers and products outside the training set. In addition, we propose the use of a pair of LCMs (called dual latent class model – DLCM), instead of a single LCM, to model customers’ likes and dislikes separately so as to enhance the prediction accuracy. Experimental results based on the EachMovie dataset show that DLCM outperforms both LCM and the conventional correlation-based method when the available ratings are sparse.**

***Index Terms*— Collaborative fitering, recommender systems, personalization, latent class models**

I. Introduction

Product recommendation is one of the most important business activities for attracting customers. With the advent of the World Wide Web, on-line companies can now recommend products to their customers on a one-to-one basis in real time, and more importantly, at a much lower cost. Different recommender systems have been proposed in the literature [1], [2] and related products/services have also been released in the market (e.g., Andromedia.com, Netperception.com). Based on the underlying technology, recommender systems can be broadly categorized as *content-based* or *collaborative*.

Content-based recommender systems match customer interest profiles (e.g., revealed by their highly rated products) with the product attributes (or *features*) when making recommendations. Different machine learning [3], [4] and information retrieval [5], [6] algorithms have been proposed for profile representation and ratings prediction. One successful application of the content-based approach is personalized Web pages recommendation (e.g., Letizia [7]). In order for the approach to be effective, sufficiently rich and accurate product information as well as personal profiles should be available. Besides, the product attributes have to be carefully chosen for the product and profile Bad choices of features result in recommender systems with either low discriminating power (the *shallow-analysis* problem) or bias in reflecting the customer interest (the *over-specialization* problem) [8].

Authors are with Department of Computer Science, Hong Kong Baptist University, Kowloon Tong, Hong Kong. E-mail: {william, tsuikc, jiming}@comp.hkbu.edu.hk

Collaborative recommender systems are based on the similarity between customer preference ratings for computing recommendations. As the approach does not rely on product contents, it does not possess the two problems of the content-based approach and thus has widely been used for recommending products where product descriptions are either lacking or found to be too specific to be useful. Many different techniques have been proposed for collaborative recommendation, including the most original correlation-based methods [9], [10], latent semantic indexing (LSI) [11], [12], Bayesian learning [13], [14], etc. Successful application domains include recommendation of Usenet articles [9], musics [10], etc. In order for collaborative recommendation to be accurate, a large enough number of customers willing to provide preference ratings for the products are required, and the product coverage of their ratings should have significant overlaps. However, this may not be the case in reality because of either lacking such a large customer pool or new products being encountered (the *sparsity* problem). Applying simple clustering or some statistical cluster models to the preference ratings has been demonstrated to be able to improve the local density of the ratings and is considered to be a promising remedy for the sparsity problem [15], [16].

In this paper, we first describe a statistical cluster model — the latent class model (LCM), originally proposed by Hofmann *et al.* for collaborative filtering [15], and study how a properly trained LCM can also be used to handle customers and products outside the training set for recommendation. Also, we argue that the LCM is limited in terms of correctly modeling like and dislike ratings and propose a dual latent class model (DLCM) which is trained using two sets of data converted from the original ratings, one with ratings for liked items and another with those for disliked ones. This modification allows the groupings of customers with similar *likes* and *dislikes* to be captured and thus improve the overall predictive power of the model. Experiments based on the EachMovie dataset were conducted for performance evaluation. It was found that DLCM outperforms LCM and a conventional correlation-based method when the ratings are sparse.

II. Collaborative Recommender Systems

The concept of collaborative recommendation (also called the word-of-mouth approach) was first used in Goldberg *et al.*’s e-mail filtering system [17]. The idea was then quickly pursued for product recommendation. In this section, we further elaborate the sparsity problem and briefly survey some existing methods proposed in the literature for alleviating it.

A. The Sparsity Problem

Most of the pioneering collaborative systems use the correlation-based approach for recommendation prediction. For example, in [9], the predicted rating of customer x for product y is computed as

$$p_{x,y} = \bar{r}_x + \kappa \sum_i w(x,i)(r_{i,y} - \bar{r}_i),$$

where $r_{i,y}$ denotes the recorded ratings of customer i for product y, $\bar{r}_i$ denotes the expected rating of customer i over all the products, $w(x,i)$ denotes the Pearson correlation coefficient (P-Corr) between the ratings of customers x and i, given as

$$w(x,i) = \frac{\sum_j (r_{x,j} - \bar{r}_x)(r_{i,j} - \bar{r}_i)}{\sqrt{\sum_j (r_{x,j} - \bar{r}_x)^2 \sum_j (r_{i,j} - \bar{r}_i)^2}}$$

and κ is a normalization constant. The correlation-based approach (also called *memory-based* in [13]) has been known to be problematic when the available ratings are sparse. To alleviate the problem, both statistical and non-statistical *model-based* methods have been proposed so that problem-specific prior knowledge can be incorporated to interpolate the missing ratings. An extreme form of the sparsity problem is called the *first-rater problem*, which arises when there are new products introduced into the market with no previous rating at all. In that case, either some ratings have to be collected or content-based information has to be explored for recommendations personalization.

B. Model-Based Methods

The spirit behind model-based methods is to incorporate prior knowledge into the problem formulation so that missing customer ratings can be properly interpolated for improving the generalization performance. Besides, if the model is properly chosen, the estimated values of the model parameters can provide useful information about the characteristics of customers and products (cf. data mining), which in many cases is useful for further market analysis and management decision making. For example, customer segmentation can be readily achieved via the use of cluster models. Various methods have been proposed in the literature, including cluster models [15], dependency models [18], classifer models [19] and subspace methods [12].

III. Latent Class Models

The latent class model (LCM) is a statistical model under the family of mixture models. Hofmann *et al.* adopted it for modeling the co-occurence of two random variables and successfully applied it to document catagorization [20] and collaborative filtering [15]. It is a cluster model which assumes that customer preference ratings come from a number of *latent* classes (or hidden preference patterns), each corresponds to a group of like-minded customers and their corresponding set of preferred products. The predicted ratings are computed based on a probabilistic graphical model with three random variables representing the customers, their liked products and the preference patterns.

Mathematically speaking, let $z \in \mathcal{Z} = \{z_1, ..., z_K\}$ denote a latent class variable, where z_i represents the i^{th} hidden preference pattern. Also, let (x,y) denote the observation that customer $x \in \mathcal{X} = \{x_1, ..., x_N\}$ has evaluated product $y \in \mathcal{Y} = \{y_1, ..., y_M\}$ and $n(x,y)$ denote the corresponding preference rating of customer x for product y.[1] The joint probability distribution of x and y can be expressed as

$$P(x,y) = \sum_{z' \in Z} P(z')P(x|z')P(y|z') \qquad (1)$$

where $P(x|z)$ and $P(y|z)$ are class-conditional multinomial distributions and $P(z)$ are the class prior probabilities. So, x and y are assumed to be conditionally independent if z is given.

A. Model Training

In the model training phase, the number of hidden preference patterns, K, is first assumed[2] and the parameters of the LCM, including $\{P(z)\}$, $\{P(x|z)\}$ and $\{P(y|z)\}$, are then estimated accordingly. The total number of model parameters is $K + N \times K + M \times K$. The expectation and maximization (EM) algorithm is typically used. The EM algorithm is an efficient optimization algorithm for solving the maximum-likelihood estimation problem with missing information (the hidden preference patterns in our case). It involves two steps: the E-step and the M-step. The E-step is for computing the expected values of the missing information based on the current estimate of the model parameters and the M-step is for computing the maximum-likelihood estimates of the model parameters using the expected values of the missing information. To train a LCM, the E-step and the M-step can be formulated, given as

E-step

$$P(z|x,y) = \frac{P(z)P(x|z)P(y|z)}{\sum_{z'} P(z')P(x|z')P(y|z')}$$

M-step

$$\begin{aligned}
P(z) &= \frac{\sum_{x',y'} n(x',y')P(z|x',y')}{\sum_{x',y',z'} n(x',y')P(z'|x',y')} \\
P(y|z) &= \frac{\sum_{x'} n(x',y)P(z|x',y)}{\sum_{x',y'} n(x',y')P(z|x',y')} \\
P(x|z) &= \frac{\sum_{y'} n(x,y')P(z|x,y')}{\sum_{x',y'} n(x',y')P(z|x',y')}
\end{aligned}$$

The EM algorithm alternates the two steps until it converges to a local maximum. The generalization performance of the converged solution depends on both the model initiation as well as the model complexity (i.e., the number of preference patterns). In this paper, an algorithm similar to K-means clustering is used for the model initialization. Also, models with different number of preference patterns are trained for subsequent performance comparison.

[1]In Hofmann's paper, [15] $n(x,y)$ denotes the number of times the pair (x,y) has been observed and the customer preference rating is represented by another random variable. Here we assume that the number of observations should carry similar information as the customer rated preference.

[2]Automatically determining the optimal number of hidden patterns is under the field of model selection, which is not addressed in this paper.

B. Recommendation Prediction

To provide personalized recommendations to customers inside the training set [15], the probability that a customer x buys a product y can be computed as

$$P(y|x) = \sum_{z' \in \mathcal{Z}} P(z'|x)P(y|z') \tag{2}$$

where

$$P(z|x) = \frac{P(x|z)P(z)}{\sum_{z' \in \mathcal{Z}} P(x|z')P(z')}. \tag{3}$$

Products can then be sorted according to their associated values of $P(y|x)$.

IV. Extensions to the LCM

Although the LCM has been shown to be promising for collaborative recommendation, there are at least two limitations preventing it from being effectively applied. First, only customers in the training set can be served and only products in the training set can be recommended. Even though preference ratings can be collected from a newly registered customer, the computational cost required for re-training the model makes real-time recommendation impossible. The second limitation is related to model accuracy. The original LCM cannot distinguish between missing ratings and ratings of value zero. This means that the *dislike* ratings (i.e., ratings with low values) cannot be properly utilized for discovering customer preference patterns, resulting in limited prediction power of the LCM.

In this section, we describe how the LCM can be extended for handling customers and products outside the training set. Besides, we propose a dual LCM which divides customer ratings into "like" and "dislike" sets to remedy the model deficiency of the LCM.

A. Recommending Products to New Customers

New customers are here referred to the ones not existing in the training set but some preference ratings have been collected from them. Our goal here is to recommend existing products to these customers as well without retraining the model.

Let $x_n \notin \mathcal{X}$ denote a new customer and $\mathcal{Y}_h \subset \mathcal{Y}$ the set of products he has rated so far. The probability of recommending product $y_r \in \mathcal{Y}_r = \mathcal{Y} \backslash \mathcal{Y}_h$ can be computed as

$$P(y_r|x_n) = \sum_{z \in \mathcal{Z}} P(z|x_n)P(y_r|z) \tag{4}$$

where all except $P(z|x_n)$ have been estimated and stored in the LCM. To estimate $P(z|x_n)$ which is the probability that customer x_n falls into the preference pattern z,

$$\begin{aligned} P(z|x_n) &\simeq \hat{P}(z|x_n, \mathcal{Y}_h) \\ &\propto \sum_{y_h \in \mathcal{Y}_h} P(z|y_h)\hat{P}(x_n|y_h) \\ &\propto \sum_{y_h \in \mathcal{Y}_h} P(z|y_h)n(x_n, y_h) \\ &\propto \sum_{y_h \in \mathcal{Y}_h} P(y_h|z)P(z)n(x_n, y_h) \end{aligned} \tag{5}$$

where we assume that $P(y_h)$ is constant. According to Eq.(5), it is noted that the estimation of $P(z|x_n)$ is equivalent to a simple correlation between $P(y_h|z)$ and $n(x_n, y_h)$ weighted by $P(z)$.

For the extreme case when the new customer has not rated anything before, i.e., $\mathcal{Y}_h = \emptyset$, Eq.(4) degenerates to

$$P(y|x_n) = P(y) = \sum_{z \in \mathcal{Z}} P(z)P(y|z). \tag{6}$$

which gives recommendations simply based on the "averaged" opinions of all the customers in the training set, or in other words, the most popular products in $\mathcal{Y}$ among all the customers in $\mathcal{X}$.

As the role of products and customers can be interchanged in the LCM, recommending new products y_n can also be achieved in a similar manner. Ratings for new products can be collected from various customers in the training set and the parameters $P(z|y_n)$ can be estimated accordingly.

B. DLCM — Modeling Likes and Dislikes

Apart from the inability to handle data unknown in the training step, another important limitation of the LCM is that dislike[3] ratings are not properly modeled. Under the LCM, the correlation of a customer's ratings with a set of bad ratings (rating value =0) is intrinsically identical to that with a set of missing ratings (rating value =unknown). That is, both zero-rated and unrated products do not affect the iterations in the model training step at all. Since the fact that a customer does not like a movie is obviously not identical to that a customer has not rated that movie, this model deficiency is highly likely to limit the prediction accuracy of the LCM. To relieve the limitation, we convert the ratings data $\{n(x, y)\}$ into two data sets $\{n^+(x, y)\}$ and $\{n^-(x, y)\}$ where the former one denotes like ratings and the latter one denotes dislike ratings. The conversion is performed using a threshold (or natural vote). Ratings in $\{n(x, y)\}$ with values higher than the threshold remain unchanged in $\{n^+(x, y)\}$. Ratings with values lower than the threshold are set to zeros in $\{n^+(x, y)\}$. Similarly, ratings in $\{n(x, y)\}$ with values lower than the threshold are set to $\{1 - n(x, y)\}$ in $\{n^-(x, y)\}$. Ratings with values higher than the threshold are set to zeros in $\{n^-(x, y)\}$. In that case, $\{n^+(x, y)\}$ contains only like ratings and $\{n^-(x, y)\}$ contains only dislike ratings. For missing ratings, as there are no cues about their values, they are set to 0.5 in both $\{n^+(x, y)\}$ and $\{n^-(x, y)\}$. Eq.(7) and Eq.(8) are the formula for the conversion.

$$n^+(x, y) = \begin{cases} n(x, y) & : \ for\ n(x, y) >= 0.5 \\ 0 & : \ for\ n(x, y) < 0.5 \\ 0.5 & : \ for\ n(x, y)\ is\ missing \end{cases} \tag{7}$$

$$n^-(x, y) = \begin{cases} 1 - n(x, y) & : \ for\ n(x, y) < 0.5 \\ 0 & : \ for\ \ n(x, y) >= 0.5 \\ 0.5 & : \ for\ n(x, y)\ is\ missing \end{cases} \tag{8}$$

[3]Here, for ratings within the range of [0,1], those within [0,0.5) refers to as dislike ratings while those within [0.5,1] refers to as like ratings.

For example, if there is a set of ratings {0.0, 0.8, ??, 0.4, ??, 0.6, 0.2, 1.0}, it will be converted to {0.0, 0.8, 0.5, 0.0, 0.5, 0.6, 0.0, 1.0} (like ratings) and {1.0, 0.0, 0.5, 0.6, 0.5, 0.0, 0.8, 0.0} (dislike ratings).

Using the two sets of converted ratings, we then train two LCMs, one using $\{n^{+}(x,y)\}$ and another using $\{n^{-}(x,y)\}$. A dual LCM (DLCM) is thus defined as the pair of LCMs, containing two sets of parameters: $\{\ P^{+}(y|z),\ P^{+}(x|z),\ P^{+}(z)\ \}$ and $\{\ P^{-}(y|z),\ P^{-}(x|z),\ P^{-}(z)\ \}$. The former set summarizes the like ratings and models how likely x "likes" y. Similar to the orginal LCM, z represents the preference pattern. The latter set of parameters summarizes the dislike ratings and models how likely x "dislikes" y, and z now represents the dislike patterns.

C. Recommendation

To recommend products using the proposed DLCM, Bayes factor can be computed. Bayes factor is defined as the ratio of the posterior probabilities that customer x likes and dislikes product y, given as

$$\begin{aligned}\frac{P(L=1|x,y)}{P(L=0|x,y)} &= \frac{P(x,y|L=1)}{P(x,y|L=0)} \times \frac{P(L=1)}{P(L=0)} \\ &= \frac{P^{+}(x,y)}{P^{-}(x,y)} \times \frac{P(L=1)}{P(L=0)} \\ &= \frac{P^{+}(y|x)}{P^{-}(y|x)} \times \frac{P^{+}(x)}{P^{-}(x)} \times \frac{P(L=1)}{P(L=0)} \quad (9) \\ &\simeq \frac{P^{+}(y|x)}{P^{-}(y|x)} \times \frac{P(L=1)}{P(L=0)}. \quad (10)\end{aligned}$$

The higher the value of the Bayse factor, the higher the chance that the corresponding product item is liked by the customer. Here, L denotes a bi-variate random variable indicating "like" if $L = 1$ and "dislike" if $L = 0$. Also, it is assumed that the customer prior probabilities $P^{+}(x)$ and $P^{-}(x)$ are constant and identical, which is used when jumping from Eq.(9) to Eq.(10). This assumption can be justified by the fact that all the customers are treated to be identical before any ratings are collected from them.

According to the Eq.(10), the prior probabilites $P(L = 0)$ and $P(L = 1)$, are computed by counting the numbers of "likes" and "dislikes" in the training set using 0.5 as the threshold. $P^{+}(y|x)$ and $P^{-}(y|x)$ are obtained using the EM algorithm similar to that of the standard LCM (see Section III-A). Then, we can compute the Bayes factor for each of the product items and rank them accordingly.

V. Experimental Results

A. Experiment Setup

In order to evaluate the effectiveness of the proposed extensions, experiments have been performed using the EachMovie database. The full set of the database consists of 72,916 customer preference ratings for 1628 different movies. The ratings are discretized into 6 levels, as 0, 0.2, 0.4, 0.6, 0.8 and 1. In this empirical study, ten subsets of data are sampled from different parts of the database in such a way that each data subset contains 180 customers' preference ratings for 500 different movies and we use the ratings of the first 100 customers for training and the remaining for testing. Also, the ten data subsets form two groups, denoted as {`data_100_*`} and {`data_50_*`}. The former one contains customers having at least 100 ratings and the latter one contains customers having at least 50 ratings. Using these two groups of datasets, we can evaluate recommendation performance under different level of data sparsity. It is noted that the datasets in {`data_100_*`} also have higher degree of ratings overlapping (65.2-72.7 co-rated items on average) when compared with those in {`data_50_*`} (28.5-39.4 on average).

We then applied the three methods — the Pearson correlation method (P-Corr), the latent class model (LCM) and the dual latent class model (DLCM) to the datasets for performance comparison.

B. Performance Evaluation

In [15], perplexity is used as the performance measure. However, it cannot be used here since we are trying to predict the preferences of customers *not* appearing in the training set. Instead, our evaluation is based on three different measures. The first one is the traditional classification *accuracy*. In our experments, we calculate the accuracy of P-Corr, LCM and DLCM via thresholding on the predicted ratings, the posterior probabilities and the Bayes factor respectively. The corresponding thresholds are set to be 0.5, $0.5/M$ and 1. The second measure is the *break-even* point, which is commonly used in area of information retrieval. Here, movies in the test set are ordered with decreasing preference (predicted), and the break-even point is the point at which *recall* equals *precision*. In the current context, recall is the percentage of interesting movies that can be located, whereas precision is the percentage of movies that are predicted to be interesting and are really interesting to the customer. The third measure is based on the expected *utility* used in [13]. Again, we utilize the list as used in computing the break-even point. We assume that each successive item in this list will be less likely to be viewed by the user with an exponential decay. Then, for customer x, the expected utility of this list is:

$$R_x = \sum_j \frac{\max(n(x,j)-d,0)}{2^{(j-1)/(\beta-1)}},$$

where d is the neutral rating (here, we take 0.5) and β is the viewing half-life (set to 5). We also compute the maximum and minimum achievable utilities R_i^{max} and R_i^{min}, and the final score is then computed as:

$$\text{utility} = (R_i - R_i^{min})/(R_i^{max} - R_i^{min}). \quad (11)$$

For all the three measures, we only compute them based on the rated movies in the test set.

Comparing the three performance measures, the breakeven point and the accuracy can be viewed as evaluating the ranking quality at a coarse-level as both of them are not sensitive if adjacent items in the ranked output list are flipped. Instead, detailed ranking quality can be revealed using the utility measure. For example, if the ranked recommendation output is { movieA, movieB , movieC } and they are all liked by the customer with the same order of preference, the accuracy and recall (which in turns affects the break-even point) will remain unchanged even though their orders are changed. However, the utility measure will be different.

C. Results and Discussions

1) Performance Comparison Among P-Corr, LCM and DLCM: We have performed a number of experiments to compare the recommendation performance of P-Corr, LCM and DLCM. The results are tabulated in Table I, II and III.

Performance measure: break-even point			
Datasets: {`data_100_*`} (# customers=180; # products=500)			
# rated items	P-Corr (%)	LCM (%)	DLCM (%)
2.1	72.3 (3.3)	73.3 (2.2)	**75.3** (1.6)*
10.9	75.9 (2.4)	74.5 (1.9)	**76.7** (1.6)
19.1	76.4 (2.1)	74.6 (2.1)	**76.9** (1.5)
25.5	76.8 (1.9)	74.7 (2.0)	**77.0** (1.8)
33.1	**77.2** (2.0)	74.5 (2.0)	**77.2** (1.9)
Datasets: {`data_50_*`} (# customers=180; # products=500)			
# rated items	P-Corr (%)	LCM (%)	DLCM (%)
1.2	68.0 (3.5)	70.6 (1.9)	**71.3** (2.0)*
6.3	70.9 (2.1)	71.0 (2.0)	**72.2** (1.4)
11.5	71.9 (1.7)	70.9 (2.0)	**72.2** (1.6)
14.2	72.2 (1.5)	71.0 (2.0)	**72.3** (1.6)
18.6	**72.5** (1.6)	70.8 (2.0)	72.3 (1.7)

TABLE I

PERFORMANCE COMPARISON USING BREAK-EVEN POINT. EACH REPORTED VALUE IS AN AVERAGE OVER FIVE DATA SETS. THE BRACKETED VALUE IS THE CORRESPONDING STANDARD DEVIATION. THE COMPARISON IS DONE BETWEEN P-CORR AND THE BEST OF LCM AND DLCM. RESULTS WITH {*} INDICATE THAT THEY ARE SIGNIFICANTLY BETTER WITH 80% CONFIDENCE ACCORDING TO T-TEST.

Performance measure: accuracy			
Datasets: {`data_100_*`} (# customers=180; # products=500)			
# rated items	P-Corr (%)	LCM (%)	DLCM (%)
2.1	65.4 (4.5)	70.2 (1.7)	**73.7** (2.4)***
10.9	**75.3** (1.9)	72.0 (2.1)	73.9 (2.4)
19.1	**75.5** (2.1)	72.2 (2.2)	73.8 (2.4)
25.5	**75.7** (2.2)	72.3 (2.3)	73.8 (2.4)
33.1	**75.9** (2.1)	72.3 (2.2)	73.8 (2.4)
Datasets: {`data_50_*`} (# customers=180; # products=500)			
# rated items	P-Corr (%)	LCM (%)	DLCM (%)
1.2	59.3 (1.3)	73.9 (3.2)	**78.3** (4.6)***
6.3	76.9 (3.7)	74.9 (2.5)	**78.4** (4.6)
11.5	78.0 (4.8)	75.2 (3.0)	**78.3** (4.6)
14.2	**78.4** (4.5)	75.2 (3.0)	78.3 (4.6)
18.6	**78.7** (4.3)	75.1 (3.1)	78.3 (4.6)

TABLE II

PERFORMANCE COMPARISON USING ACCURACY. RESULTS WITH {***} INDICATE THAT THEY ARE SIGNIFICANTLY BETTER WITH 95% CONFIDENCE ACCORDING TO T-TEST.

Based on Table I and II, it is observed that LCM outperforms P-Corr significantly when the number of items rated by the new customer is small (which is used to be the case in practice). In particular, according to Table II, when the number of rated items is around 2 for the data sets in {`data_50_*`}, LCM has an accuracy of 73.9% while that of P-Corr is 59.3%

Performance measure: expected utility			
Datasets: {`data_100_*`} (# customers=180; # products=500)			
# rated items	P-Corr (%)	LCM (%)	DLCM (%)
2.1	48.9 (4.6)	**61.1** (3.5)**	59.6 (2.7)
10.9	63.6 (1.8)	**65.6** (2.0)*	62.2 (2.4)
19.1	**67.5** (1.5)*	65.7 (2.0)	62.8 (1.4)
25.5	**68.8** (1.2)***	65.9 (2.0)	63.5 (1.6)
33.1	**69.6** (1.4)***	65.8 (1.9)	63.8 (1.7)
Datasets: {`data_50_*`} (# customers=180; # products=500)			
# rated items	P-Corr (%)	LCM (%)	DLCM (%)
1.2	48.6 (3.3)	62.6 (1.6)	**62.9** (1.9)***
6.3	58.5 (2.5)	**64.6** (2.8)***	63.9 (1.9)
11.5	64.1 (2.4)	**64.2** (3.2)	63.6 (2.1)
14.2	**64.4** (2.6)	**64.4** (3.3)	63.6 (1.8)
18.6	**67.2** (0.9)*	64.2 (3.2)	64.0 (1.9)

TABLE III

PERFORMANCE COMPARISON USING EXPECTED UTILITY. RESULTS WITH {*},{**} AND {***} INDICATE THAT THEY ARE SIGNIFICANTLY BETTER WITH 80%, 90% AND 95% CONFIDENCE ACCORDING TO T-TEST.

(14.6% better). Consistent results were obtained for the other two performance measures, i.e., breakeven point and expected utility. However, when the new customer rates more and more items, LCM (model-based) start losing its advantages and P-Corr (memory-based) eventually has better performance. According to tabulated results, as the number of rated items is increased to around 18, LCM's accuracy rate continues to raise and eventually saturated at around 75.1% while P-Corr's raises to 78.7% (3.6% higher than that of LCM). This most probably is due to the deficiency of the LCM to model both like and dislike ratings. It also echos a similar situation in the field of pattern recognition where the simple k-nearest neighbor method can outperform other powerful model-based approaches like artifical neural networks when the training set is sufficiently large and clean.

By using the proposed DLCM, we succeeded in improving the prediction power of LCM. As revealed in Table I, II and III, DLCM, as inherited from the model-based advantages of LCM, performs significantly better than P-Corr when the number of rated items is small. When there are more ratings, we found DLCM in general can achieve accuracy rates and breakeven points comparable to those of P-Corr. In particular, According to Table II and I, when the number of rated item is around 18 for the data set {`data_50_*`}, the accuracy and break-even point of P-Corr are 78.7% and 72.5% while those of DLCM are 78.3% and 72.3%. The differences are smaller than 0.4%. However, when the expected utility is used instead as the performance measure, the performance of the DLCM, though still better than that of P-Corr, is not as good as that of the LCM. For example, according to Table III, when the number of rated items is around 18, the expected utilities of P-Corr, LCM and DLCM are 67.2%, 64.2% and 64.0% respectively. Such an observation implies that the current implementation of the DLCM is capable of improving the ordering of the products at a coarse level (as revealed by the measures of break-even point and accuracy) but not at a fine level (as revealed by the utility measure). Further investigation along this direction is needed.

2) Different Number of Latent Classes: All the aforementioned experiments assume that the number of latent classes K is set to 10. In fact, estimating the optimal number of latent classes (i.e., hidden preference patterns) is crucial for controling the model complexity of LCM and DLCM so that they are flexible enough to capture the true patterns but at the same time strict enough to avoid spurious patterns due to noise in the data. In this paper, we did not attempt the problem of estimating the optimal number of latent classes, which is definitely an important open research problem. Instead, we tried three different number of $K = \{5, 10, 15\}$.[4] Generally speaking, better results were obtained by setting K to 10 or 15, instead of 5. This reveals that the number of hidden patterns should be somehow more than five. Also, the performance difference between the two cases K=10 and K=15 is not significant and overfitting (i.e., models with bigger K work worser) was observed for some cases.

3) Different Degrees of Ratings Overlapping: To further understand the advantage of adopting model-based methods like LCM and DLCM, we can compare the results based on the two groups of datasets {`data_100_*`} and {`data_50_*`} which are reported in the upper and lower halves of Table I, II and III. Also, we used the bold face to highlight the best performance among the three methods. It is noted that more reported values with bold face were found in the columns of LCM and DLCM in the lower halves (i.e., {`data_50_*`}) of the three tables when compared with the upper halves (i.e., {`data_100_*`}). It indicates that the performance gain (or improvement) due to LCM and DLCM, when compared with P-Corr, is more prominant for the datasets in {`data_50_*`} which has a significantly lower degree of ratings overlapping. Such an observation coheres with our understanding that the model-based approach has the advantage of interpolating missing data, especially when the data is sparse.

VI. Conclusion

In this paper, the latent class model has first been extended so that it can be used to recommend products to new customers. The extension was empirically found to outperform the standard Pearson correlation method only when the number product items rated by the new customers is small. As the limitation is mainly caused by the instrinsic deficiency of the adopted latent class model in discriminating missing and zero (dislike) ratings, a dual latent class model was then proposed, which corrects the deficiency by modeling the like and dislike ratings separately. The empirical results show that such a separation can further improve the model's prediction power.

Acknowledgment

The EachMovie dataset is by courtesy of Digital Equipment Corporation and this research is jointly supported Hong Kong Baptist University via Faculty Research Grant FRG/99-00/II-36P and RGC Grant HKBU/2090/01E.

[4]Due to the space limitation, more experimental details can be found in [21].

References

[1] "*Recommender Systems - Papers from the AAAI Workshop*," AAAI, Tech. Rep. WS-98-08, 1998.

[2] *Proceedings of the SIGIR-99 Workshop on Recommender Systems: Algorithms and Evaluation*, University of California, Berkeley, Aug. 1999.

[3] R. Mooney and L. Roy, "Content-based book recommending using learning for text categorization," in *Proceedings of the SIGIR-99 Workshop on Recommender Systems: Algorithms and Evaluation*, Berkeley, CA, Aug. 1999.

[4] M. Pazzani, J. Muramatsu, and D. Billsus, "Syskill & Webert: Identifying interesting web sites," in *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, Portland, 1996, pp. 54–61.

[5] B. Sheth and P. Maes, "Evolving agents for personalized information filtering," in *Proceedings of the Ninth IEEE Conference on Artificial Intelligence for Applications*, 1993, pp. 345–352.

[6] I. Stadnyk and R. Kass, "Modeling users' interests in information filters," *Communications of the ACM*, vol. 35, no. 12, pp. 49–50, Dec. 1992.

[7] H. Lieberman, "Letizia: An agent that assist Web browsing," in *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence (IJCAI97)*, Montreal, Canada, Aug. 1995.

[8] M. Balabanović and Y. Shoham, "Content-based, collaborative recommendation," *Communications of the ACM*, vol. 40, no. 3, pp. 66–72, Mar. 1997.

[9] P. Resnick, N. Iacovou, M. Suchak, P. Bergstorm, and J. Riedl, "GroupLens: An open architecture for collaborative filtering of netnews," in *Proceedings of ACM 1994 Conference on Computer Supported Cooperative Work*, Chapel Hill, NC, 1994, pp. 175–186.

[10] U. Shardanand and P. Maes, "Social information filtering: Algorithms for automating 'word of mouth'," in *Proceedings of the Computer-Human Interaction Conference (CHI95)*, Denver, CO, May 1995.

[11] M. Pryor, "The effects of singular value decomposition on collaborative filtering," Compute Science Department, Dartmouth College, Tech. Rep. PCS-TR98-338, June 1998.

[12] J. Jiang, M. Berry, J. Donato, G. Ostrouchov, and N. Grady, "Mining consumer product data via latent semantic indexing," *Intelligent Data Analysis*, vol. 3, pp. 377–398, 1999.

[13] J. Breese, D. Heckerman, and C. Kadie, "Empirical analysis of predictive algorithms for collaborative filtering," in *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*, Madison, WI, July 1998.

[14] L. Ungar and D. Foster, "Clustering methods for collaborative filtering," in *Recommender Systems - Papers from the AAAI Workshop*, Madison, WI, July 1998.

[15] T. Hofmann and J. Puzicha, "Latent class models for collaborative filtering," in *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence (IJCAI99)*, 1999, pp. 688–693.

[16] J. Konstan, B. N. Miller, D. Maltz, J. L. Herlocker, L. Gordon, and J. Riedl, "Applying collaborative filtering to Usenet news," *Communications of the ACM*, vol. 40, no. 3, pp. 77–87, Mar. 1997.

[17] D. Goldberg, D. Nichols, B. Oki, and D. Terry, "Collaborative filtering to weave an information tapestry," *Communications of the ACM*, vol. 35, no. 12, pp. 61–70, Dec. 1992.

[18] C. Aggarwal, J. Wolf, K. Wu, and P. Yu, "Horting hatches an egg: A new graph-theoretic approach to collaborative filtering," in *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge discovery and data mining*, San Diego, Aug. 1999, pp. 201–212.

[19] D. Billsus and M. Pazzani, "Learning collaborative information filter," in *Proceedings of the Fifteenth International Conference on Machine Learning*, Madison, WI, 1998, pp. 46– 54.

[20] T. Hofmann, "Learning and representing topic - a hierarchical mixture model for word occurences in document databases," in *Conference for Automated Learning and Discovery, Workshop on Learning from Text and the Web*, CMU, 1998.

[21] K.-W. Cheung, K.-C. Tsui, and J. Liu, "Extended latent class models for collaborative recommendation," Hong Kong Baptist University, Tech. Rep. COMP-03-017, 2003.

List of Tables

I Performance comparison using break-even point. Each reported value is an average over five data sets. The bracketed value is the corresponding standard deviation. The comparison is done between P-Corr and the best of LCM and DLCM. Results with {*} indicate that they are significantly better with 80% confidence according to t-test. 5
II Performance comparison using accuracy. Results with {***} indicate that they are significantly better with 95% confidence according to t-test. 5
III Performance comparison using expected utility. Results with {*},{**} and {***} indicate that they are significantly better with 80%, 90% and 95% confidence according to t-test. 5

Footnotes

1) In Hofmann's paper, [15] $n(x, y)$ denotes the number of times the pair (x, y) has been observed and the customer preference rating is represented by another random variable. Here we assume that the number of observations should carry similar information as the customer rated preference.
2) Automatically determining the optimal number of hidden patterns is under the field of model selection, which is not addressed in this paper.
3) Here, for ratings within the range of [0,1], those within [0,0.5) refers to as dislike ratings while those within [0.5,1] refers to as like ratings.
4) Due to the space limitation, more experimental details can be found in [21].