Mining Customer Product Ratings for Personalized Marketing

Kwok-Wai Cheung [1] James T. Kwok [2] Martin H. Law [2] Kwok-Ching Tsui [1]

[1]Department of Computer Science
Hong Kong Baptist University
Kowloon Tong
Hong Kong
{william,tsuikc}@comp.hkbu.edu.hk

[2]Department of Computer Science
Hong Kong University of Science and Technology
Clear Water Bay
Hong Kong
{jamesk,martin}@cs.ust.hk

Abstract

With the increasing popularity of Internet commerce, a wealth of information about the customers can now be readily acquired on-line. An important example is the customers' preference ratings for the various products offered by the company. Successful mining of these ratings can thus allow the company's direct marketing campaigns to provide automatic, *personalized* product recommendations. In general, these recommender systems are based on two complementary techniques. Content-based systems match customer interests with information about the products, while collaborative systems utilize preference ratings from the other customers. In this paper, we address some issues faced by these systems, and study how recent machine learning algorithms, namely the support vector machine and the latent class model, can be used to alleviate these problems.

Keywords: Recommender systems, personalized marketing, support vector machine, latent class model

1 Introduction

Direct marketing is a promotion process which motivates customers to place orders through various channels [33]. A conventional example is direct mail, which sends out appealing customized catalogs or coupons to targeted customers. In order for this to work, one needs to have an accurate customer segmentation based on a good understanding of the customers, so that relevant product information can be delivered to different customer segments. Moreover, the costs involved in the related marketing research have to be carefully controlled. With the recent advances in the World Wide Web and the booming e-business, on-line companies can now acquire individual customer's information via the Internet in real-time and at a much lower cost. Based on these acquired information, detailed customer profiles can be built to support one-to-one marketing and other personalized services (*personalization*). The effectiveness of direct marketing in cyberspace can thus be greatly improved.

Another consequence of moving the storefront to the cyberspace is that a tremendous amount of product information can now be made available to the customers at a very low cost. However, customers, who were used to having only a limited range of product choices due to physical and/or time constraints, are now facing the problem of *information overload*. An effective way to increase customer satisfaction and consequently customer loyalty should be one that helps the customers identify products according to their interests. This, again, calls for the provision of personalized product recommendations.

Nowadays, *recommender systems* have been implemented by many big Web retailers, such as `Amazon.com` and `CDNow.com`. Typically, they use an intelligent engine to mine the customer's ratings records and then create predictive user models for product recommendation. These customer ratings may either be acquired explicitly by form-filling or implicitly via an intelligent agent [12]. Purchase records can also be good indicators of customer preferences. Moreover, software products for recommender systems are also available, such as NetPerceptions' `Personalization engine`, Andromedia's `LikeMind` and Manna's `FrontMind`. All these can be integrated directly into a company's Web server to provide personalized product recommendations.

Based on the underlying technology, recommender systems can be broadly categorized as *content-based* or *collaborative*. Content-based systems provide recommendations to a customer by automatically matching his/her interests with product contents, and have been reported in the literature on recommending webpages [40], newsgroup messages [31] and news items [9]. Notice that recommendations are made without relying on information provided by the other customers. Moreover, for certain products such as movies as will be discussed in Section 4, content information (such as the cast of a certain movie) can now be readily extracted from the Web. While extracting product contents may be straight forward, the resultant set of product attributes can be too large to handle. Existing systems rely heavily on preprocessing steps that select a manageable set of "important" features from these attributes. These, however, are often *ad hoc* and do not always show consistent improvement (The Feature Selection Problem).

Collaborative systems, on the other hand, provide recommendations by utilizing overlap of preference ratings to combine the opinions of "like-minded" customers. Using this word-of-mouth approach, customers may now be able to receive recommendations for products that are dissimilar in content to those he/she has previously rated, as long as other like-minded customers prefer them. The concept of collaboration was first used by Goldberg *et al.* for filtering e-mails [19], and then quickly pursued for product recommendation [43, 45]. Most pioneering collaborative systems are based on the use of correlation coefficients, which, however, are known to be sensitive to the sparsity of preference ratings (The Sparsity Problem) and also problematic when new products are introduced (The First-Rater Problem) [28].

In this paper, we propose the use of the *support vector machine* (SVM) [8, 48] and the *latent class model* (LCM) [22] to circumvent some of the aforementioned problems. Unlike other learning methods, SVM's performance is related not to the number of features in the system, but to the *margin* with which it separates the data. Experimentally, SVM has achieved superior performance on a number of high-dimensional data sets. The LCM is a statistical mixture model which assumes that the observed data come from a set of hidden causes [34]. It has recently been applied to document categorization [18] and texture segmentation [23]. Here, the use of the LCM allows some "latent" clusters of preference ratings to be formed so that the sparse ratings distribution can effectively be smoothed out.

While the techniques discussed in this paper are generally applicable to any recommender system, for performance evaluation purposes, we consider here a specific recommendation task, namely, movie recommendation. This task has been popularly used as a testbed for recommender algorithms [3, 5, 6, 7, 11, 20, 21, 22], and a number of commercial websites (such as http://movies.eonline.com and http://reel.com) are providing such service. Moreover, to be more specific, we will use the Internet Movie Database (IMDb) for movie contents and the EachMovie database for movie ratings. These two databases exemplify the typical problems faced by recommender systems mentioned above. For example, as will be detailed in Section 4.1.1, the number of features extracted from the IMDb is very large (equal to 6,620). As for the EachMovie database, there are a total of 1,628 movies and 72,916 users. The number of possible movie-user pairs is thus $1628 \times 72916 = 1.187 \times 10^8$. The number of movie ratings actually present, however, is 2,811,983, which is equivalent to a data density of merely 2.4%.

The rest of this paper is organized as follows. Section 2 describes content-based recommender systems, with particular emphasis on the feature selection problem, and then an introduction to the SVM. Section 3 describes collaborative recommender systems, together with the sparsity and first-rater problems, and then an introduction to the LCM and our extension. Evaluation results on the SVM and the extended LCM are presented in Section 4, and the last section gives some concluding remarks.

2 Content-Based Recommendation

In content-based systems, products are described by a common set of attributes extracted from the available product descriptions. Customer's preferences are then predicted by analyzing the relationship between the product ratings that particular individual has given and the corresponding product attributes. A number of techniques have been used. A rudimentary method is by simple keyword matching [10]. However, the use of keywords suffers from the well-known problems of synonymy and polysemy in information retrieval [14]. Moreover, customer preferences are often too complex to be adequately captured by a set of keywords. Another popular technique that is capable of producing accurate recommendations is the naive Bayes classifier [36]. This, however, relies on the simple and often unrealistic assumption that product attributes are probabilistically independent of one another. Other algorithms, such as rule-based systems [5] and the winnow algorithm [41], have also been used.

2.1 The Problem of Feature Selection

A central problem in content-based recommender systems is the need to identify a sufficiently large set of key attributes. When the set is too small, obviously there will be insufficient information to learn the customer profile. However, having too many attributes can also be problematic, as this results in a large number of trainable parameters in the model and consequently poor performance [35]. This is further aggravated by the usual feature representation scheme of using a set of binary features to represent multi-valued attributes. For example, the attribute Cast that describes the cast of a movie is typically represented by a set of binary features such as "Cast includes Dustin Hoffman", "Cast includes Bruce Willis", "Cast includes Leonardo DiCaprio", etc. Because of the large number of possible actors and actresses, the resultant set of features can be very sizable[1], and it is therefore not surprising that a straight forward application of the approach may yield inferior performance [5].

To circumvent this problem, existing techniques thus rely heavily on manual/automatic preprocessing steps that select "useful" features [13]. However, the effectiveness of feature selection is quite controversial. Many of the selected features contain redundant information. Moreover, a feature that appears to be a poor predictor on its own may turn out to have great discriminative power in combination with others. Besides, another important question that has not been addressed thoroughly is how many features should be selected. In practice, this has to be chosen based on the individual's experience.

Some machine learning algorithms, such as decision trees, have built-in methods of selecting features while generating the target concept, and may thus appear immune to the above controversy. However, [2, 32] have shown that these learning algorithms are not effective at minimizing the number of features in the face

[1]For example, the total number of features used in our experiments in Section 4.1 is 6,620.

of many irrelevant features, as may exist in content-based recommendation.

The issue of dimension reduction has also been studied widely in statistics and pattern recognition. All the methods mentioned above fall under the so-called *filter model* [26], in which they are performed as a preprocessing step. John *et al.* [26] proposed a *wrapper model* which uses a learning algorithm, together with cross-validation, as an alternative evaluating function in choosing the required features. However, this model can become very computationally expensive when there are a lot of candidate features.

2.2 Support Vector Machine

SVM has performed very well in a number of high-dimensional data sets even without feature selection. For example, it outperforms radial basis networks on recognizing the US postal service database of handwritten digits (using 16×16 bitmap as input) [44]. Promising results are also reported in face detection [39] (where the input dimensionality is 283) and text categorization [17, 24, 29] (where the input dimensionality is over 10,000). Its power stems from automatic regularization and also framing the computational problem as a quadratic programming problem. In this section, we briefly review SVMs for classification. Interested readers may refer to [8, 48] for details.

2.2.1 Learning the SVM Model

Assume that the customer has provided preference ratings for m products. To learn the SVM, we are given a training set containing tuples of the form $\{(\mathbf{f}_j, v_j)\}_{j=1}^m$, with $\mathbf{f}_j$ (as input) being product j's feature vector and v_j (as output) being the corresponding preference rating. Here, the preference rating can either be $+1$ (indicating that the user likes this product) or -1 (indicating that the user does not like this product). Thus, in the context of pattern recognition, we have a binary classification problem, and the task is to separate products that the customer likes from those that he/she dislikes.

First consider the simpler case when the data is *linearly separable*, i.e., the two classes of products in the above binary classification problem can be perfectly separated by a linear surface. The SVM constructs a hyperplane[2] $\mathbf{w}^T\mathbf{f} + b$ for which the separation between the two classes is maximized (Figure 1). Mathematically, this is equivalent to minimizing $\|\mathbf{w}\|$ and it can be shown that $\mathbf{w} = \sum_{j=1}^m \alpha_j v_j \mathbf{f}_j$, where $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_m)^T$ can be found by solving the following quadratic programming (QP) problem:

$$\text{maximize } W(\boldsymbol{\alpha}) = \boldsymbol{\alpha}^T\mathbf{1} - \frac{1}{2}\boldsymbol{\alpha}^T\mathbf{Q}\boldsymbol{\alpha}, \tag{1}$$

under the constraints $\boldsymbol{\alpha} \geq \mathbf{0}$ and $\boldsymbol{\alpha}^T\mathbf{v} = 0$, where $\mathbf{1} = (1, 1, \ldots, 1)^T, \mathbf{v} = (v_1, \ldots, v_m)^T$ and $\mathbf{Q}$ is a $m \times m$ symmetric matrix with entries $v_j v_k \mathbf{f}_j^T \mathbf{f}_k$. Notice that $\mathbf{Q}$ is always positive semi-definite and so there is no

[2]Here, $\mathbf{w}$ and b are the weight vector and bias of the hyperplane respectively. The superscript T denotes vector transpose.

local optima for the QP problem. For those α_j's greater than zero, it can be shown that the corresponding training examples must lie along the margins of the decision boundary, and these are called the *support vectors*.

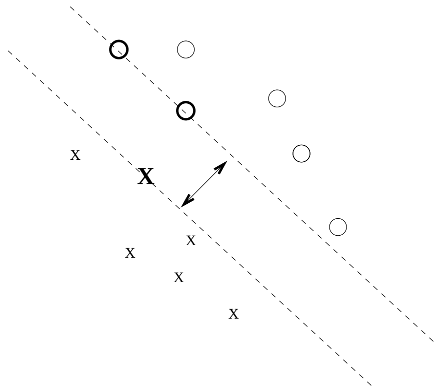


Figure 1: In this figure, patterns belonging to the two classes are denoted by $\circ$ and $\times$ respectively, and the double arrow denotes the margin that is to be maximized by the SVM algorithm. The resultant support vectors are shown in boldface.

When the training set is not linearly separable, the SVM algorithm accommodates the discrepancies by introducing non-negative slack variables ξ_j's that (whenever nonzero) measure the difference between v_j and the corresponding SVM output $a_j = \mathbf{w}^T\mathbf{f}_j + b$. Then, instead of minimizing $\|\mathbf{w}\|$ in the separable case, the problem now becomes:

$$\text{minimize } \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{j=1}^{m}\xi_j,$$

subject to $v_j a_j \geq 1 - \xi_j$ and $\xi_j \geq 0$, for $j = 1, \ldots, m$ where C is a user-defined regularization parameter. Again, this minimization problem can be transformed to a QP problem, as: maximize EQ.(1) subject to the constraints $C\mathbf{1} \geq \boldsymbol{\alpha} \geq \mathbf{0}$ and $\boldsymbol{\alpha}^T\mathbf{v} = 0$.

So far, we have only considered separating the two classes of products by a hyperplane. It is often more effective to perform the separation by nonlinear hyper-surfaces, say by a polynomial surface of degree 4. In SVM, this is achieved by nonlinearly mapping the original product feature vectors to another space $\mathcal{F}$ via a mapping $\mathbf{u} = \phi(\mathbf{f})$, and then apply the same hyperplane algorithm in $\mathcal{F}$. The only change in the computation is that entries of $\mathbf{Q}$ now become $v_j v_k \mathbf{u}_j^T\mathbf{u}_k$. In general, it can be cumbersome to explicitly compute $\mathbf{u}_j, \mathbf{u}_k$ and then the dot product. For example, to construct a polynomial of degree 4 in a 200 dimensional space, $\mathbf{u}_j$ will have dimensionality of over a billion. However, instead, it can be shown that the computation can be made much more efficient by using a *kernel* function $K(\cdot,\cdot)$ such that $K(\mathbf{f}_j, \mathbf{f}_k) = \mathbf{u}_j^T\mathbf{u}_k$. For example, the kernel for a polynomial classifier of degree d is $K(\mathbf{f}_j, \mathbf{f}_k) = (\mathbf{f}_j^T\mathbf{f}_k + 1)^d$. Discussions on more kernels can be found in [48].

2.2.2 Making Recommendations

Once the SVM model for a particular customer has been learned, one can then determine whether to recommend another product (with feature vector $\mathbf{f}$) to this customer. This is done by using $\mathbf{w}^T\mathbf{f} + b$ as an estimate of the customer's preference. The larger its value, the more preferable is the product. More sophisticated techniques, based on a Bayesian analysis of SVM [30], may also be used.

3 Collaborative Recommendation

In the following, let $\mathcal{Y}$ be the whole set of products, $\mathcal{Y}_h \subset \mathcal{Y}$ be the subset that have been rated by customer x and $\mathcal{Y}_r = \mathcal{Y} \setminus \mathcal{Y}_h$ be the subset that have not been rated by the same customer. Collaborative systems estimate customer x's preferences for products in $\mathcal{Y}_r$ based on the overlap between his/her preference ratings for products in $\mathcal{Y}_h$ and those of the other customers. According to Breese *et al.* [7], collaborative systems can be categorized as *memory-based* and *model-based.* The memory-based approach computes the proximities of ratings between the targeted customer and each of the others in the entire database, and then estimate the preferences of the targeted customer for the unrated products accordingly. Different proximity estimates have been proposed, with the Pearson correlation coefficient being the most common [43, 45]. Model-based systems, on the other hand, use the preference ratings in the database to learn a model and then use this model in estimation. Different statistical models, including a variety of clustering models [22, 47] and the Bayesian network [7], have been used.

Memory-based methods were proposed much earlier and are known to suffer from two problems, namely the *sparsity* and the *first-rater* problems. Model-based methods were recently proposed mainly for alleviating the sparsity problem to achieve higher recommendation accuracy.

3.1 The Sparsity and First-Rater Problems

Collaborative recommender systems assume the presence of a large enough number of customers willing to provide preference ratings to many products. However, this may not be the case in reality. In fact, as most of the pioneering collaborative systems are based on the use of correlation coefficients (memory-based) and the effectiveness of the coefficients relies heavily on the degree of overlapping among the customer ratings, accuracy of the predicted ratings degrades significantly when the available ratings are sparse (sparsity problem). To alleviate this limitation, Konstan *et al.* [28] enhanced their Usenet news recommender system by partitioning the ratings and the user correlations databases based on the Usenet newsgroup hierarchy. Having demonstrated that grouping ratings information is an effective remedy, different clustering and related model-based techniques have then been applied to collaborative recommendation with promising results

[22, 27, 47]. Besides being capable of providing more accurate predictions, the trained models also possess important information for market analysis. For example, product and customer segments may be derived from the clustering structure.

An extreme form of the sparsity problem is the first-rater problem, which arises when a new product is introduced into the market and thus has no previous ratings information available either for computing the correlations (in memory-based approaches) or training the models (model-based approaches). Although it is common practice for a lot of companies to deliver all the new product information to all their customers indiscriminately, some form of customization will still be preferred if the list of new products is long and/or the new products fall into many different categories (e.g., books, CDs, etc.). In this respect, the content-based approach should be able to play an important complementary role. For example, the Fab system [4] is one of the pioneering works along the direction of integrating the two recommendation techniques for solving the first-rater problem. However, in this paper, we emphasize on the use of the LCM and so the integration issue will not be addressed.

3.2 Latent Class Model

The LCM is a model-based approach. It assumes that preference ratings of the customers come from a pre-defined number of "latent" classes $\mathcal{Z} = \{z_1, ..., z_K\}$. Each latent class thus corresponds to the preference pattern of a group of like-minded customers.

In the following, let (x, y) be the observation that customer x has evaluated product $y \in \mathcal{Y}$, and $n(x, y)$ be the corresponding preference rating[3]. The joint probability distribution of x and y can be expressed as:

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

where $P(x|z)$ and $P(y|z)$ are the class-conditional multinomial distributions and $P(z)$ is the class prior probability. The dependency relationships among the three random variables are shown in Figure 2. Conditional independence of x and y given z implies that once the preference pattern z of a targeted customer is known, the products to be recommended are basically independent of the targeted customer[4].

3.2.1 Learning the LCM

To learn the LCM, we use a training set containing the preference ratings $n(x, y)$ of a set of customers on the products in $\mathcal{Y}$. Parameters in the distributions $P(z)$, $P(x|z)$ and $P(y|z)$ of the LCM are then estimated by the

[3] In Hofmann's paper [22], $n(x, y)$ denotes the number of times the pair (x, y) has been observed and the customer's preference is represented by another random variable. Here, we assume that these two are the same.

[4] Mathematically, conditional independence of x and y given z means $P(x, y|z) = P(x|z)P(y|z)$.

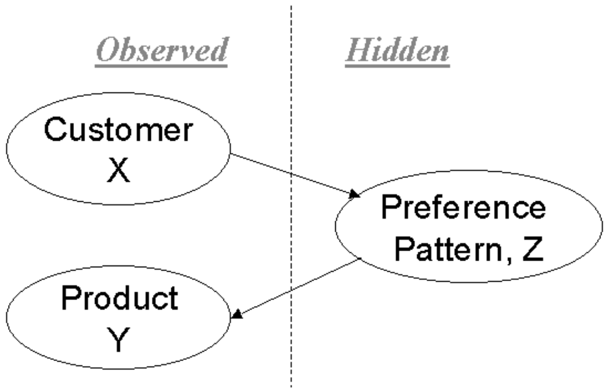


Figure 2: Dependency diagram of the LCM.

expectation maximization (EM) algorithm [15], which is an efficient optimization algorithm for maximum-likelihood estimation with missing information. In the LCM, the missing information is which latent classes the customers and products belong to. The EM algorithm involves two steps: the E-step and the M-step. The E-step computes the expected values of the missing information based on the current model estimate:

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

while the M-step computes the maximum-likelihood model estimates given the expected values of the missing information:

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

The E-step and M-step are then iterated until convergence.

3.2.2 Recommendation Within the Training Set

Using the Bayes rule, the probability that customer x buys product y is:

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

where $P(z|x) = P(x|z)P(z)/\sum_{z' \in \mathcal{Z}} P(x|z')P(z')$. With a number of products, they can then be sorted by $P(y|x)$ when providing recommendations.

3.2.3 Recommendation Outside the Training Set

Hofmann and Puzicha [22] did not discuss on how the LCM can be used to provide recommendations to customers whose ratings have not been used to train the LCM. Here, we propose a method by using preference ratings that the customer has rated so far. Let $x_n \notin \mathcal{X}$ be the new customer. The probability of recommending product $y \in \mathcal{Y}_r = \mathcal{Y} \backslash \mathcal{Y}_h$ is equal to $P(y|x_n) = \sum_{z \in \mathcal{Z}} P(z|x_n)P(y|z)$. Here, the only unknown, $P(z|x_n)$, is the probability that x_n falls in the latent class z. To estimate its value, we use the only available information about the new customers x_n, i.e., his/her ratings for products in $\mathcal{Y}_h$. This is then used to compute the similarity between those ratings and the probability distribution of $y \in \mathcal{Y}_h$ given z. Mathematically, $P(z|x_n)$ is estimated as:

$$\begin{aligned} P(z|x_n) &= \sum_{y_h \in \mathcal{Y}_h} \hat{P}(z, y_h|x_n) \\ &= \sum_{y_h \in \mathcal{Y}_h} P(z|y_h)P(y_h|x_n) \\ &\propto \sum_{y_h \in \mathcal{Y}_h} P(y_h|z)P(z)P(x_n, y_h) \\ &\simeq \sum_{y_h \in \mathcal{Y}_h} P(y_h|z)P(z)n(x_n, y_h) \end{aligned} \tag{3}$$

where the prior $P(y_h)$ on the products in $\mathcal{Y}_h$ is assumed constant. The estimation of $P(z|x_n)$ based on EQ.(3) is equivalent to an inner product between $P(y_h|z)$ and $n(x_n, y_h)$ weighted by $P(z)$. If the probability distribution on $\mathcal{Y}_h$ is similar to the distribution of $\{n(x_n, y_h) : y_h \in \mathcal{Y}_h\}$, the inner product will result in a higher value, indicating that it is highly likely for x_n belonging to the preference pattern z.

In the extreme case where the new customer has not rated anything before, $\mathcal{Y}_h = \emptyset$ and $P(y|x_n)$ degenerates to $P(y|x_n) = P(y) = \sum_{z \in \mathcal{Z}} P(z)P(y|z)$. Recommendations are thus simply based on the "averaged" opinions of all customers in the training set. In other words, the most popular products in $\mathcal{Y}$ among all customers in $\mathcal{X}$ will be recommended.

4 Evaluation

For performance evaluation, we form our dataset by using the Internet Movie Database (IMDb)[5] for movie contents and the EachMovie database[6] for movie ratings. The EachMovie database was collected by the EachMovie collaborative filtering site, deployed by the Compaq Systems Research Center for an 18-month period from 1995 through 1997. Some collaborative systems [20, 21] have used another movie ratings database

[5]The IMDb database is available at http://www.imdb.com.

[6]The EachMovie database is available at http://www.research.compaq.com/SRC/eachmovie.

from the MovieLens system[7], which contains 122,176 ratings from 1173 users. Other works using smaller, proprietary databases have also been reported [3, 5]. The advantages, however, in our selection of the IMDb and EachMovie databases are that they are large (with a total of 1,628 movies and 2,811,983 user ratings from 72,916 users), publicly available and also have been widely used in previous studies (e.g., [6, 7, 11, 22]). The ratings in the EachMovie are discretized into 6 levels, as 0, 0.2, 0.4, 0.6, 0.8 and 1. In the following, we define a movie as "interesting" to an individual customer if his/her preference rating for this movie is greater than 0.5.

In our experiments, two different performance measures have been used. The first one we use is the *break-even* point, which is the point at which *recall* equals *precision*, and has been commonly used in information retrieval. In the current context, movies in the test set are ordered by decreasing preference (estimate) v_{ij} and 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. If one is more interested in the quality of the highly ranked items, the precision rate at low recall rate (say 10%) can also be used. The second measure is based on the expected utility used in [7] for evaluating the quality of the predicted product ranking. Again, we utilize the list 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 customer with an exponential decay. The expected utility for customer x_i is then:

$$R_i = \sum_j \frac{\max(v_{ij} - d, 0)}{2^{(j-1)/(\beta-1)}},$$

where d is the neutral vote (here, we take 0.5) and β is the viewing half-life (which is 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:

$$\text{utility} = \frac{R_i - R_i^{min}}{R_i^{max} - R_i^{min}}. \tag{4}$$

4.1 Content-Based Recommendation

In this section, we compare five techniques that can be used for content-based recommendation, including the naive Bayes classifier, 1-nearest-neighbor classifier, the SVM, the decision tree classifier C4.5 [42] and its associated production rule generator C4.5rules.

4.1.1 Movie Information

Information about the movies, as mentioned above, are extracted from the Internet Movie Database (IMDb). The following 12 attributes are extracted from each movie record:

[7] The MovieLens system is developed by the University of Minnesota, and is available at http://movielens.umn.edu.

- Continuous attributes: Release date and Runtime.
- Multi-valued attributes in which each movie can take on at most one value : Language, Certification, Directors, Producers, Original music and Writers. Note that Directors, Producers, Original music and Writers may actually involve more than one person. However, for simplicity, we will only consider the first one that appears on the list.
- Multi-valued attributes in which each movie may take on multiple values: Genre, Country, Plot keyword and Cast. Because of the possibly large number of actors, we extract only the first 10 from each movie.

For the multi-valued attributes, we take the popular approach of representing each of them as a set of binary features. For example, the Cast feature will be represented as a set of binary features such as "Cast includes Dustin Hoffman", "Cast includes Bruce Willis", "Cast includes Leonardo DiCaprio", etc. The total size of the resultant set of features is 6620. Moreover, here we adopt a purely content-based approach and so no collaborative information (e.g. such as User comments, External reviews and Newsgroup reviews) from the IMDb has been used. As demonstrated in previous studies [36], the incorporation of collaborative content can be expected to further boost the performance.

4.1.2 Experimental Setup

Results reported here are based on 5-fold cross-validation, averaged over 100 customers randomly selected from the EachMovie database. All 1,628 movies are used, and no feature selection is performed except for C4.5 and C4.5rules. Moreover, recall that computations of both the break-even point and the utility measure in EQ.(4) require ranking the movies by decreasing preference estimates. For the naive Bayes classifier, this is performed by ranking the movies by decreasing posterior odds. For the SVM, we rank the movies by decreasing $\mathbf{w}^T\mathbf{f} + b$. For C4.5rules, we rank by the distance (based on the simplified value difference metric [16]) between $\mathbf{f}_j$ and its nearest rule.

4.1.3 Results

Table 1 shows the performance of different content-based recommendation methods. As can be seen, the superiority of SVM is statistically significant under both measures.

	utility (%)	break-even point (%)
SVM	**63.6***	**80.3***
naive Bayes	60.0	78.8
C4.5rules (#feature=100)	52.3	75.1
C4.5rules (all features)	52.6	76.0
1-nearest-neighbor	45.5	76.2

Table 1: Performance of different content-based recommender systems. The best results are indicated with boldface. The results marked with asterisks mean that the improvement is 99% confident according to the t-test.

4.2 Collaborative Recommendation

4.2.1 Experimental Setup

In this section, we compare the LCM with the standard memory-based method using the Pearson correlation coefficient (P-Corr). The first 500 movies and a random subset of 500 customers[8] from the EachMovie dataset are used. 5-fold cross-validation is adopted for the evaluation. For each customer in the test set, we use ratings for the first 250 products to estimate the latent class probability $P(z|x_n)$ and those for the remaining 250 to compute the performance. For training the LCM, the EM algorithm we used can only converge to a local minimum in general, and its performance thus depends on the model initialization step. In our implementation, we have tried two different schemes. The first one is random initialization. The second one is based on a simple clustering scheme. Under the second scheme, $P(z)$ is initialized to $1/N_z$. For $P(y|z)$, ratings for all the customers of a particular product are collected into one feature vector (with all the missing ratings are set to zero) and then the K-means clustering algorithm is applied (with $K = N_z$). Moreover, $P(y|z)$ is initialized to one if y is in cluster z, and zero otherwise. $P(x|z)$ is initialized similarly, and an individual customer's ratings on all the products are collected as a feature vector for clustering. Experimental results reveal that both schemes give similar results. All the results reported in the following are based on random initialization. For performance evaluation, we have used break-even point, precision at 10% of recall and utility.

4.2.2 Results

Table 2 tabulates the results. Performance of the LCM is generally superior to that of P-Corr, especially when the preference history is short. We have also performed the t-test for our cross-validation results. As

[8]These are randomly selected from customers that have provided at least 50 ratings for the 500 movies.

revealed in the table, based on the utility measure, the improvement is statistically significant in most of the cases, especially when the length of preference history is short. This improvement due to LCM is also in line with the argument that model-based approaches can effectively alleviate the sparsity problem. However, as the history length continues to increase, the performance of the LCM is found to be saturated while that of P-Corr keeps improving. This probably implies model deficiency in the LCM and further research effort is required along the model enhancement direction.

Besides, it is observed that the improvement is not that significant when the evaluation is based on the break-even point but is much more prominent when the precision rate at 10% recall rate is used. Similar observation is also shown in Figures 3(a) and 3(b), the recall-precision curves for P-Corr and the LCM ($N_z = 15$) respectively. By comparing the two figures, it is noted that both the P-Corr and the LCM perform similarly at around the break-even point. But when the recall is low (which is more common in recommender systems as customers normally do not prefer to go through a long list to look for interested items), the LCM achieves a significantly higher precision when compared to P-Corr.

Also, the performance of the LCM first improves as the number of latent classes increases. It then degrades when the number continues to increase, possibly caused by overfitting. According to our experiment, the optimal number of latent classes varies for different preference history lengths. The question of selecting an optimal number of latent classes remains an open research issue.

5 Conclusion

In this paper, we applied two recent machine learning methods for recommender systems. We used the support vector machine for content-based recommendation. This yields superior performance when compared to other traditional content-based techniques, while at the same time avoids the problem of feature selection. For collaborative recommendation, we extended the latent class model to recommend products to customers outside the training set. Experimentally, this model-based approach can effectively alleviate the sparsity problem.

An important characteristic of product recommendation is that the size of the customer pool and the number of available products are usually very large. So, other than accuracy, system efficiency is also a major concern. For both SVM and LCM, most of the computational cost is involved in the training process, and this can be done off-line in the initial training phase. After that, the resultant model is a compact representation of the raw ratings information, and thus the time and space complexities on making recommendations are quite low. With the (possibly batched) arrival of more preference ratings and/or new products, the models will still have to be re-trained. The SVM has been experimentally shown to perform very well under incremental training [46]. As for the EM algorithm, both fast and incremental versions have

history length	No. of latent classes	utility (%) LCM	utility (%) P-Corr	precision at 10% recall (%) LCM	precision at 10% recall (%) P-Corr	break-even point (%) LCM	break-even point (%) P-Corr
2.4	1	57.66	50.15	89.08	81.20	79.33	77.84
	2	**63.36***		92.90		**80.19**	
	3	62.49		92.32		80.07	
	5	62.96		**93.20***		80.10	
	8	62.31		92.29		80.04	
4.3	1	57.66	55.78	89.08	86.43	80.59	79.28
	2	**64.34***		**93.76***		80.60	
	3	64.08		93.43		80.50	
	5	64.17		93.47		**80.66**	
	8	63.81		93.62		80.47	
6.2	1	57.66	57.08	89.08	86.43	79.33	79.43
	2	64.78		94.17		80.77	
	3	64.86		**94.21***		80.78	
	5	**65.08***		94.05		80.75	
	8	64.63		93.69		**80.89**	
8.7	1	57.66	58.54	89.08	87.61	79.33	79.79
	2	64.81		94.06		80.83	
	3	64.86		94.07		80.79	
	5	**65.25***		**94.21***		80.91	
	8	64.78		93.47		**80.98**	
11.2	1	57.66	61.37	89.08	89.70	79.33	80.55
	2	64.82		**93.94**		80.83	
	3	64.86		93.92		80.80	
	5	**65.16**		93.91		**80.92**	
	8	64.81		93.65		80.87	

Table 2: Performance comparison between LCM and P-Corr. The best results are indicated with boldface. It is noted that LCM outperforms P-Corr in all the cases. The results marked with asterisks mean that the improvement is 99% confident according to the t-test.

also been reported recently [37, 38].

While content-based recommendation and collaborative recommendation are complementary in nature, in situations where both the product descriptions and the preference ratings are abundant, it would further boost the performance by integrating these two approaches. Recently, this is being investigating by a number of researchers [1, 4, 5, 11, 20], and we will continue to investigate this in the future.

Acknowledgments

This research has been partially supported by the Research Grants Council of the Hong Kong Special Administrative Region under grant HKUST2033/00E and the Hong Kong Baptist University under grant FRG/99-00/II-36P. The EachMovie dataset for this paper was provided by Digital Equipment Corporation.

The authors would also like to thank Thorsten Joachims for his SVM-Light [25] used in the experiments.

References

[1] C.C. Aggarwal, J.L. Wolf, K.L. Wu, and P.S. 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*, pages 201–212, San Diego, August 1999.

[2] H. Almuallim and T.G. Dietterich. Learning with many irrelevant features. In *Proceedings of the Ninth National Conference on Artificial Intelligence*, pages 547–552, Anaheim, CA, USA, 1991.

[3] J. Alspector, A. Kolez, and N. Karunanithi. Comparing feature-based and clique-based user models for movie selection. In *Proceedings of the Third ACM Conference on Digital Libraries*, pages 11–18, 1998.

[4] M. Balabanović and Y. Shoham. Content-based, collaborative recommendation. *Communications of the ACM*, 40(3):66–72, March 1997.

[5] C. Basu, H. Hirsh, and W. Cohen. Recommendation as classification: Using social and content-based information in recommendation. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence*, pages 714–720, Madison, WI, USA, July 1998.

[6] D. Billsus and M.J. Pazzani. Learning collaborative information filter. In *Proceedings of the Fifteenth International Conference on Machine Learning*, pages 46– 54, Madison, WI, 1998.

[7] J.S. 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.

[8] C.J.C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):955–974, 1998.

[9] V.L. Centeno, C.F. Panadero, and C.D. Kloos. Personalizing your electronic newspaper. In *Proceedings of the 4th Euromedia Conference*, pages 26–28, April 1999.

[10] M. Claypool, A. Gokhale, and T. Miranda. Combining content-based and collaborative filters in an online newspaper. In *SIGIR'99 Workshop on Recommender Systems: Algorithms and Evaluation*, 1999.

[11] M.K. Condliff and D.D. Lewis. Bayesian mixed-effects models for recommender systems. In *Proceedings of the SIGIR-99 Workshop on Recommender Systems: Algorithms and Evaluation*, Berkeley, CA, August 1999.

[12] Robert Cooley, Bamshad Mobasher, and Jaideep Srivastava. Data preparation for mining world wide web browsing patterns. *Knowledge and Information Systems*, 1(1):5–32, February 1999.

[13] M. Dash and H. Liu. Feature selection for classification: A survey. *Intelligent Data Analysis*, 1(3), 1997.

[14] S. Deerwester, S.T. Dumais, G.W. Furnas, T.K. Landauer, and R. Harshman. Indexing by latent semantic analysis. *Journal of the American Society of Information Science*, 41(6):391–407, September 1990.

[15] A.P. Dempster, N.M. Laird, and D.B. Rubin. Maximum-likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B*, 39:1–38, 1977.

[16] P. Domingos. Unifying instance-based and rule-based induction. *Machine Learning*, 24:141–168, 1996.

[17] S. Dumais. Using SVMs for text categorization. *IEEE Intelligent Systems*, pages 21–23, 1998.

[18] D. Gildea and T. Hofmann. Topic-based language models using EM. In *Proceedings of Sixth European Conference On Speech Communication and Technology (Eurospeech99)*, 1999.

[19] D. Goldberg, D. Nichols, B.M. Oki, and D. Terry. Collaborative filtering to weave an information tapestry. *Communications of the ACM*, 35(12):61–70, December 1992.

[20] N. Good, J.B. Schafer, J.A. Konstan, A. Borchers, B. Sarwar, J. Herlocker, and J. Riedl. Combining collaborative filtering with personal agents for better recommendations. In *Proceedings of the 1999 Conference of the American Association of Artifical Intelligence (AAAI-99)*, July 1999.

[21] J.L. Herlocker, J.A. Konstan, A. Borchers, and J. Riedl. An algorithmic framework for performing collaborative filtering. In *Proceedings of the 1999 Conference on Research and Development in Information Retrieval*, August 1999.

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

[23] T. Hofmann, J. Puzicha, and J.M. Buhmann. Unsupervised texture segmentation in a deterministic annealing framework. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(8):803–818, August 1998.

[24] T. Joachims. Text categorization with support vector machines: Learning with many relevant features. In *European Conference on Machine Learning*, 1998.

[25] T. Joachims. Making large-scale support vector machine learning practical. In B. Schölkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods – Support Vector Learning*, pages 169–184. MIT Press, 1999.

[26] G.H. John, R. Kohavi, and K. Pfleger. Irrelevant features and the subset selection problem. In W.W. Cohen and H. Hirsh, editors, *Machine Learning: Proceedings of the Eleventh International Conference*, pages 121–129, San Francisco, CA, 1994. Morgan Kaufmann Publishers.

[27] Arnd Kohrs and Bernard Merialdo. Clustering for collaborative filtering applications. In *Computational Intelligence for Modelling Control and Automation*. IOS Press, Vienna, 1999.

[28] J.A. Konstan, B. N. Miller, D. Maltz, J. L. Herlocker, L.R. Gordon, and J. Riedl. Applying collaborative filtering to Usenet news. *Communications of the ACM*, 40(3):77–87, March 1997.

[29] J.T. Kwok. Automated text categorization using support vector machine. In *Proceedings of the International Conference on Neural Information Processing*, pages 347–351, Kitakyushu, Japan, October 1998.

[30] J.T. Kwok. Moderating the outputs of support vector machine classifiers. *IEEE Transactions on Neural Networks*, 10:1018–1031, 1999.

[31] K. Lang. NewsWeeder: Learning to filter Netnews. In *Proceedings of the Twelfth International Conference on Machine Learning*, pages 331–339, Tahoe City, CA, 1995.

[32] D.D. Lewis and M. Ringuette. A comparison of two learning algorithms for text categorization. In *Third Annual Symposium on Document Analysis and Information Retrieval*, pages 81–93, Las Vegas, NV, April 1994.

[33] William J. McDonald. *Direct Marketing*. McGraw-Hill, Singapore, 1998.

[34] G.J. McLachlan and K.E. Basford. *Mixture Models: Inference and Applications to Clustering*. Dekker, New York, 1988.

[35] T.M. Mitchell. *Machine Learning*. McGraw-Hill, 1997.

[36] R.J. Mooney and L. Roy. Content-based book recommending using learning for text categorization. In *SIGIR'99 Workshop on Recommender Systems: Algorithms and Evaluation*, 1999.

[37] A. Moore. Very fast EM-based mixture model clustering using multiresolution kd-tree. *Neural Information Systems Processing*, December 1998.

[38] R.M. Neal and G.E. Hinton. A view of the EM algorithm that justifies incremental, sparse, and other variants. In Michael I. Jordan, editor, *Learning in Graphical Models*, pages 355–368. Kluwer Academic Publishers, Dordrecht, 1998.

[39] E. Osuna, R. Freund, and F. Girosi. Training support vector machines: an application to face detection. In *Proceedings of Computer Vision and Pattern Recognition*, Puerto Rico, June 1997.

[40] M. Pazzani and D. Billsus. Learning and revising user profiles: The identification of interesting web sites. *Machine Learning*, 27:313–331, 1997.

[41] M.J. Pazzani. A framework for collaborative, content-based and demographic filtering. *Artificial Intelligence Review*, 1999.

[42] J.R. Quinlan. *C4.5 : Programs for Machine Learning*. Morgan Kaufmann, 1993.

[43] 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*, pages 175–186, Chapel Hill, NC, 1994.

[44] B. Schölkopf, K. Sung, C. Burges, F. Girosi, P. Niyogi, T. Poggio, and V. Vapnik. Comparing support vector machines with Gaussian kernels to radial basis function classifiers. *IEEE Transactions on Signal Processing*, 45(11):2758–2765, November 1996.

[45] 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.

[46] N.A. Syed, H. Liu, and K.K. Sung. Incremental learning with support vector machines. In *Workshop on Support Vector Machines, International Joint Conference on Artificial Intelligence*, 1999.

[47] L.H. Ungar and D.P. Foster. Clustering methods for collaborative filtering. In *Recommender Systems - Papers from the AAAI Workshop*, Madison, WI, July 1998.

[48] V. Vapnik. *Statistical Learning Theory*. Wiley, 1998.

About the Authors –

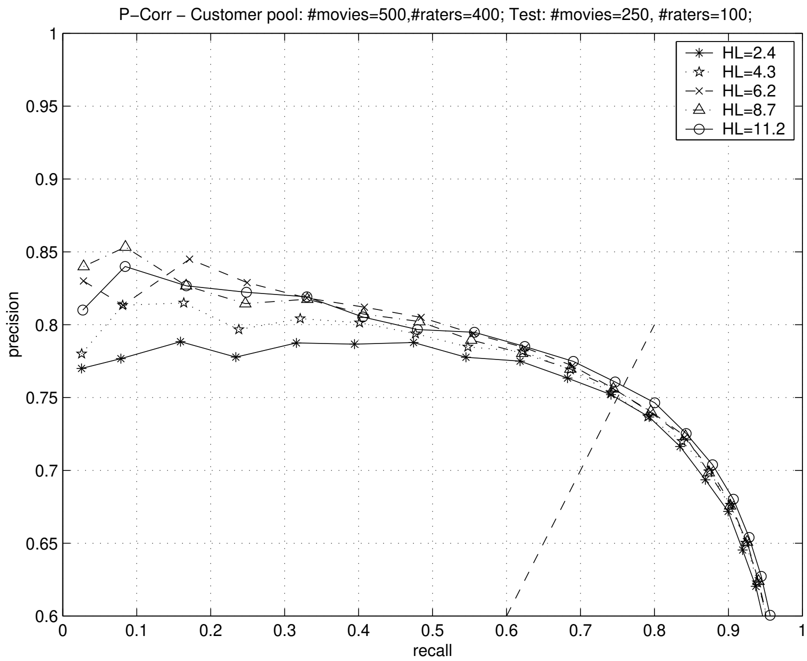
Kwok-Wai Cheung received his B.Sc. and M.Phil degrees both in electronic engineering from the Chinese University of Hong Kong, and his Ph.D. degree in computer science from the Hong Kong University of Science and Technology. He was the recipient of the Croucher Foundation Studentship and the Edward Youde Memorial Scholarship in 1991 and 1993 respectively. He is currently an Assistant Professor in the Department of Computer Sciencem, Hong Kong Baptist University. His current research interests include pattern recognition, machine learning, data mining and intelligent agents.

James T. Kwok received the Ph.D. degree in computer science from the Hong Kong University of Science and Technology in 1996. He was an Assistant Professor in the Hong Kong Baptist University. Currently, he is an Assistant Professor in the Department of Computer Science, Hong Kong University of Science and

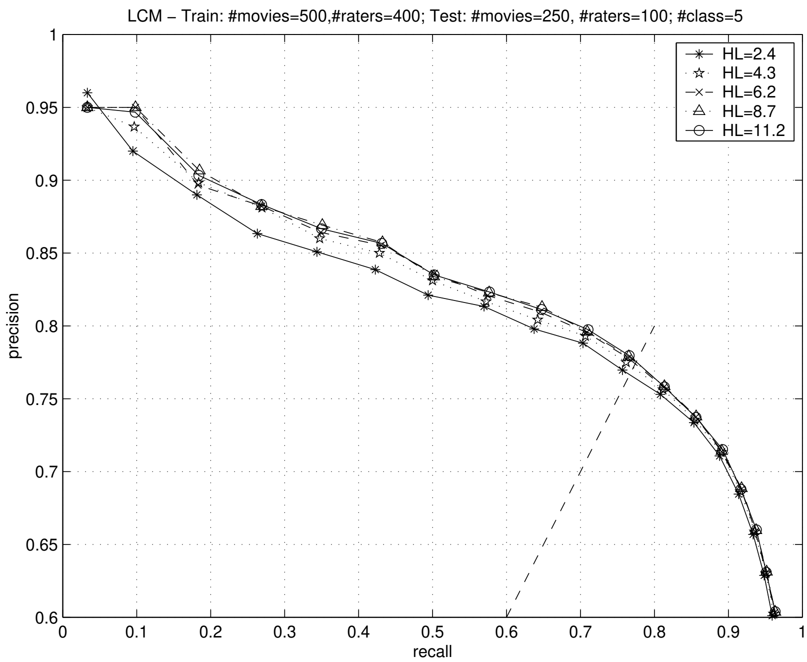
Technology. His research interests include support vector machines, artificial neural networks, pattern recognition and machine learning.

Martin H. Law

Kwok-Ching Tsui



(a) P-Corr



(b) LCM (N_z=5)

Figure 3: The recall-precision curves for the P-Corr and the LCM. It is noted that the LCM is superior to the P-Corr, especially when the recall is low.