Mining customer preference ratings for product recommendation using the support vector machine and the latent class model

William K. Cheung[1], James T. Kwok[1], Martin H. Law[1] &
Kwok-Ching Tsui[2]
[1]Department of Computer Science
Hong Kong Baptist University
Kowloon Tong, Hong Kong
[2]Intelligent Systems Research Group
BT Laboratories, MLB1/pp12, BT Adastral Park
Martlesham Health, Ipswich, Suffolk, UK IP5 3RE

Abstract

As Internet commerce becomes more popular, customers' preferences on various products can now be readily acquired on-line via various e-commerce systems. Properly mining this extracted data can generate useful knowledge for providing personalized product recommendation services. In general, recommender systems use two complementary techniques. *Content-based* systems match customer interests with products attributes, while *collaborative filtering* systems utilize preference ratings from other customers. In this paper, we address some problems faced by these two systems, and study how machine learning techniques, namely the support vector machine and the latent class model, can be used to alleviated them.

1 Introduction

Product recommendation is one of the most important business activities in attracting customers. With the advent of the World Wide Web, on-line companies can now acquire customers' preferences and recommend products accordingly on a one-to-one basis in real time, and more importantly, at a much lower cost. Such kind of systems are commonly called *recommender* *systems*. Based on the type of information used, recommender systems can be further categorized as *content-based* or *collaborative*.

Content-based systems provide recommendations by matching customer interests with product attributes. Sometimes, there are a large number of product attributes, and existing systems rely heavily on preprocessing steps that select or extract "important" features from the products. These steps, however, are often *ad hoc* and do not always show consistent improvement. In this paper, we propose the use of the *support vector machine* (SVM) [13, 14] instead. Unlike other machine learning methods, SVM's performance is related not to the input dimensionality, but to the *margin* with which it separates the data. Experimentally, SVM has achieved superior performance on a number of high-dimensional data sets (e.g. [8]).

Collaborative systems, on the other hand, utilize the overlap of preference ratings among customers for product recommendation [5, 11, 12]. The correlation coefficient is commonly used, which, however, is sensitive to the sparsity of rating information. Model-based techniques can be used to alleviate this problem. In particular, the *latent class model* (LCM) [6] is adopted and extended in this paper. The LCM is a family-of-mixture model originally proposed for the modeling of the co-occurrence of two random variables. Recently, promising results have also been reported on applications like document categorization and texture segmentation.

The rest of this paper is organized as follows. Section 2 describes content-based recommender systems, with particular emphasize 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 the following, let $\mathcal{X} = \{x_i\}_{i=1}^N$ be the set of customers, $\mathcal{Y} = \{y_j\}_{j=1}^M$ be the set of products, and $\mathbf{V} = \{v_{ij}\}_{ij}$ be the *customer-product* matrix in which entry v_{ij} denotes customer x_i's preference rating for product y_j.

In content-based systems, all products in $\mathcal{Y}$ are described by a common set of features extracted from the available product descriptions. Each y_j is thus represented by a feature vector $\mathbf{f}_j$. Moreover, individual customer's preferences are predicted solely from the products that he/she has rated, by analyzing the relationship between the preference ratings and the corresponding product features.

A number of techniques have been used for content-based recommendation. The simplest ones include simple keyword matching [3]. However, the use of keywords suffers from the well-known problems of synonymy and polysemy. Another popular technique is the naive Bayes classifier [9], which relies on the simple, though often unrealistic, assumption that features are probabilistically independent of one another. Other algorithms, such as the winnow algorithm [10] and rule-based systems [1], have also been used.

2.1 The Problem of Feature Selection

The presence of either too few or too many product features are problematic for content-based recommender systems. With too few features, there will be insufficient information to learn the customer profile. With too many features, a large number of parameters will be resulted, and existing techniques rely heavily on preprocessing steps that select "useful" features. However, it is likely that many of these 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. Experimentally, the effectiveness of feature selection is also quite controversial. Another important question that has not been addressed thoroughly is how many features should be selected. Choosing a small number may remove important discriminative features, while choosing a large number defeats the original purpose of performing dimension reduction.

2.2 Support Vector Machine

Without the need for feature selection, SVM has performed very well in a number of high-dimensional data sets. Its power stems from automatic regularization and also framing the computational problem as a quadratic programming problem. In this section, we briefly describe SVM in the context of product recommendation.

2.2.1 Model Training

Assume that customer x has provided preference ratings for m products. The corresponding training set $\mathcal{D}$ will be $\{(\mathbf{f}_j, v_j)\}_{j=1}^m$, with the product features $\mathbf{f}_j$ as input and the preference ratings $v_j \in \{\pm 1\}$ as output. The SVM first maps $\mathbf{f}$ to $\mathbf{u} = \phi(\mathbf{f})$ in a feature space $\mathcal{F}$. When the data is linearly separable in $\mathcal{F}$, the SVM constructs a hyperplane $\mathbf{w}^T\mathbf{u} + b$ in $\mathcal{F}$ for which the separation between the positive and negative examples is maximized. It can be shown that $\mathbf{w} = \sum_{j=1}^m \alpha_j v_j \mathbf{u}_j$, where $\boldsymbol{\alpha} = (\alpha_1, \dots, \alpha_m)$ 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 $\mathbf{v}^T = (v_1, \dots, v_m)$ and $\mathbf{Q}$ is a symmetric matrix with entries $v_j v_k \mathbf{u}_j^T \mathbf{u}_k$. To obtain $\mathbf{Q}$, one does not need to get $\mathbf{u}_j$ and $\mathbf{u}_k$ explicitly. Instead, one can use a *kernel* $K(\cdot, \cdot)$ 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 $K(\mathbf{f}_j, \mathbf{f}_k) = (\mathbf{f}_j^T \mathbf{f}_k + 1)^d$. Moreover, $\mathbf{Q}$ is always positive semi-definite and so there is no local optima for the QP problem.

When the training set is not separable in $\mathcal{F}$, the SVM algorithm introduces non-negative slack variables $\xi_j \geq 0$. The resultant problem becomes: minimize $\frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{j=1}^{m} \xi_j$, subject to $v_j a(\mathbf{f}_j, \mathbf{w}) \geq 1 - \xi_j, j = 1, \ldots, m$. Here, C is a customer-defined parameter, and ξ_j, whenever it is nonzero, measures the (absolute) difference between v_j and

$$a(\mathbf{f}_j, \mathbf{w}) = \mathbf{w}^T \mathbf{u}_j + b = \sum_k \alpha_k v_k K(\mathbf{f}_j, \mathbf{f}_k) + b. \tag{2}$$

Again, this minimization problem can be transformed to a QP problem: maximize (1) subject to the constraints $\mathbf{0} \leq \boldsymbol{\alpha} \leq C\mathbf{1}$ and $\boldsymbol{\alpha}^T \mathbf{v} = 0$.

2.2.2 Recommendation

On determining whether to recommend a product (with feature vector $\mathbf{f}$) to customer x, $a(\mathbf{f}, \mathbf{w})$ in (2) is used as an estimate for the customer's preference. The larger its value, the more preferable is the product.

3 Collaborative Recommendation

In the following, let $\mathcal{Y}_h^i \subset \mathcal{Y}$ be the set of products rated by customer x_i and $\mathcal{Y}_r^i = \mathcal{Y} \setminus \mathcal{Y}_h^i$ be the set of products not yet rated by the same customer. Collaborative systems estimate customer's preferences for products in $\mathcal{Y}_r^i$ based on the overlap between his/her preference ratings for products in $\mathcal{Y}_h^i$ and those of the other customers. Algorithms for collaborative recommendation, in general, can be categorized into two classes, *memory-based* and *model-based* [2]. Memory-based approaches utilize the entire customer database to estimate his/her preferences for the unrated products. Different estimates have been proposed, such as the Pearson correlation coefficient [11, 12]. Model-based systems, on the other hand, use the database to learn a model and then use this model in estimation. Different statistical models, such as the naive Bayes classifier and the Bayesian network [2], have been used.

3.1 The Sparsity and First-Rater Problems

Recommender systems using collaborative filtering assume the presence of a large enough number of customers willing to provide preference ratings to many products, and this may not be the case in reality (*sparsity* problem). Model-based methods are usually superior to memory-based methods in this respect, as they can impose constraints via the models. Difficulties also arise when a new product comes into the market and thus has no previous preference information (*first-rater* problem). Integration of content-based and collaborative approaches is a promising paradigm to alleviate this problem.

3.2 The Latent Class Model

In this model-based approach, the preference patterns of different customers are assumed to come from several "latent" categories (or preference patterns) $\mathcal{Z} = \{z_1, ..., z_K\}$. 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. The joint probability distribution of x and y can be expressed as:

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

where $P(x|z)$ and $P(y|z)$ are the class-conditional multinomial distributions and $P(z)$ is the class prior probability. Conditional independence of x and y given z implies that once the preference pattern z is known, the customer preference is no longer depending on his/her ratings for products.

3.2.1 Model Training

Here, one has to pre-define the number of latent classes. Parameters of the LCM (including $P(z)$, $P(x|z)$ and $P(y|z)$) are then estimated by the expectation and maximization (EM) algorithm, which alternates until convergence between the E-step

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

and the M-step

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

3.2.2 Recommendation Within the Training Set

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

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

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 [6] does not discuss how the LCM can be used to provide recommendations to customers not in the training set. 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 $y_j \in \mathcal{Y}_r^j = \mathcal{Y} \backslash \mathcal{Y}_h$ is equal to $P(y_j|x_n) = \sum_{z \in \mathcal{Z}} P(z|x_n) P(y_j|z)$. Here, the only unknown, $P(z|x_n)$, is the probability that x_n falls in the latent class z. Based on the customer's preference history $\mathcal{Y}_h$ and assuming a constant $P(y_h)$, $P(z|x_n)$ can then be estimated as:

$$P(z|x_n) \simeq \hat{P}(z|x_n, \mathcal{Y}_h^n) \propto \sum_{y_h \in \mathcal{Y}_h} P(y_h|z) P(z) n(x_n, y_h). \tag{4}$$

According to (4), the estimation of $P(z|x_n)$ is thus equivalent to a simple correlation between $P(y_h|z)$ and $n(x_n, y_h)$ weighted by $P(z)$.

4 Evaluation

Customer preference information about the movies are obtained from the EachMovie database, which consists of 72,916 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 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.

Evaluation will be based on three different measures. The first one is the traditional classification accuracy. The second one is the *break-even* point, which has been commonly used in the area of information retrieval. Here, movies in the test set are ordered with decreasing preference (estimate) v_{ij}, 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 [2]. 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 neural 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:

$$\text{utility} = (R_i - R_i^{min})/(R_i^{max} - R_i^{min}). \tag{5}$$

4.1 Content-Based Recommendation

In this section, we compare five content-based recommendation techniques, including the naive Bayes classifier, 1-nearest-neighbor classifier, the SVM, the decision tree classifier C4.5 and its associated production rule generator C4.5rules. To provide a baseline reference, we have also included the majority classifier, which always predicts the most frequent class.

4.1.1 Movie Information

Information about the movies are extracted from the Internet Movie Database (http://www.imdb.com). The following 12 features are extracted from each movie record:

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

For the multi-valued features, 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”, etc. The total size of the resultant set of features is 6620.

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 1628 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 (5) 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 $a(\mathbf{f}_j, \mathbf{w})$ in (2). For C4.5rules, we rank by the distance (based on the simplified value difference metric [4]) between $\mathbf{f}_j$ and its nearest rule. For C4.5 and the majority classifiers, such an ordering cannot be produced and hence only the accuracies are reported.

4.1.3 Results

Table 1 shows the performance of different content-based recommendation methods. As can be seen, the SVM is superior in all three measures.

	accuracy (%)	break-even point (%)	utility (%)
SVM	77	80.3	65
naive Bayes	76	78.8	61
C4.5rules (#feature=100)	74	76.0	52
C4.5rules (#feature=400)	75	75.1	52
1-nearest-neighbor	69	76.2	45
C4.5 (#feature=100)	74	-	-
C4.5 (#feature=400)	74	-	-
majority	75	-	-

Table 1: Performance of different content-based recommender systems.

4.2 Collaborative Recommendation

4.2.1 Experimental Setup

In this section, we compare the LCM and the standard memory-based method using the Pearson correlation coefficient (P-Corr). A subset of 100 customers and 500 movies from the EachMovie dataset is used. The customers are selected in such a way that they have provided at least 20 ratings among the 500 movies. Ratings of the first 90 customers are used for training while those of the remaining 10 are used for testing. In the EM algorithm, $P(z)$ is initialized randomly. For $P(y|z)$, each column of the matrix $\mathbf{V}$ is considered as a feature vector and the K-mean clustering is applied. $P(y|z)$ is then initialized to one if y is in cluster z and zero otherwise. $P(x|z)$ is initialized similarly, except that rows of $\mathbf{V}$ are now taken to be the feature vector in the clustering process. The three performance measures used in content-based recommendation are also used here for evaluation. Since the LCM ranks the movies based on $P(y|x)$, a threshold is needed to compute the classification accuracy. In our experiments, we consider that customer x likes movie y if $P(y|x) > 0.5/N_y$.

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. This is because the sparsity problem becomes more significant as the preference history decreases. This improvement is also in line with the argument that model-based approaches can effectively alleviate the sparsity problem. Moreover, notice that overfitting occurs when the number of clusters increases from 6 to 10 and then to 15. The question of selecting an optimal number of latent classes, however, remains an open research issue.

length	no. of			break-even			
of	latent	accuracy (%)		point (%)		utility (%)	
history	classes	LCM	P-Corr	LCM	P-Corr	LCM	P-Corr
250	6	63	61	75.6	75.6	57	59
	10	62		77.0		62	
	15	63		75.5		59	
125	6	61	60	75.6	73.5	57	58
	10	62		76.6		63	
	15	62		75.3		60	
83	6	61	50	75.6	73.3	58	58
	10	60		77.3		61	
	15	62		72.3		60	
63	6	61	51	75.6	71.0	59	58
	10	58		77.2		62	
	15	60		72.3		59	
10	6	62	40	72.0	63.1	56	43
	10	61		71.0		56	
	15	62		70.5		55	

Table 2: Performance of two collaborative recommender systems using preference histories of different lengths. Testing results are based on the last 250 movies.

5 Conclusion

In this paper, we applied the support vector machine for content-based recommendation. This yields superior performance to other traditional content-based techniques, while also avoiding 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.

Acknowledgments

The EachMovie dataset for this paper was generously provided by Digital Equipment Corporation. The authors would also like to thank Thorsten Joachims for his SVM-Light [7] used in the experiments.

References

[1] C. Basu, H. Hirsh, and W. Cohen. Recommendation as classification: Using social and content-based information in recommendation. In

Proceedings AAAI, 1998.

[2] 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*, 1998.

[3] 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.

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

[5] 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.

[6] 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.

[7] 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, 1998.

[8] 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.

[9] 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.

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

[11] 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.

[12] 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.

[13] V. Vapnik. *The Nature of Statistical Learning Theory.* Springer Verlag, 1995.

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