Machine Learning, 57, 35–59, 2004
© 2004 Kluwer Academic Publishers. Manufactured in The Netherlands.

Mining Skewed and Sparse Transaction Data for Personalized Shopping Recommendation

CHUN-NAN HSU chunnan@iis.sinica.edu.tw
Institute of Information Science, Academia Sinica, Taiwan

HAO-HSIANG CHUNG r89057@csie.ntu.edu.tw
HAN-SHEN HUANG hanshen@iis.sinica.edu.tw
Department of Computer Science and Information Engineering, National Taiwan University

Editors: Nada Lavrač, Hiroshi Motoda, Tom Fawcett

Abstract. A good shopping recommender system can boost sales in a retailer store. To provide accurate recommendation, the recommender needs to accurately predict a customer's preference, an ability difficult to acquire. Conventional data mining techniques, such as association rule mining and collaborative filtering, can generally be applied to this problem, but rarely produce satisfying results due to the skewness and sparsity of transaction data. In this paper, we report the lessons that we learned in two real-world data mining applications for personalized shopping recommendation. We learned that extending a collaborative filtering method based on ratings (e.g., GroupLens) to perform personalized shopping recommendation is not trivial and that it is not appropriate to apply association-rule based methods (e.g., the IBM SmartPad system) for large scale prediction of customers' shopping preferences. Instead, a probabilistic graphical model can be more effective in handling skewed and sparse data. By casting collaborative filtering algorithms in a probabilistic framework, we derived HyPAM (Hybrid Poisson Aspect Modelling), a novel probabilistic graphical model for personalized shopping recommendation. Experimental results show that HyPAM outperforms GroupLens and the IBM method by generating much more accurate predictions of what items a customer will actually purchase in the unseen test data. The data sets and the results are made available for download at `http://chunnan.iis.sinica.edu.tw/hypam/HyPAM.html`.

Keywords: graphical models, user profiles, collaborative filtering, shopping recommendation, transaction data

1. Introduction

A good shopping recommender system can boost sales in a retailer store by reminding customers to purchase additional items not in their original shopping lists. With the most recent telecommunication technologies, recommendation can be pushed to potential customers via the Internet as well as all types of mobile and wireless devices. To provide personalized recommendation requires ability to accurately predict a customer's shopping preference, but this is difficult to achieve because usually there are tens of thousands of product items for tens of thousands of customers and the number of possible preference orderings is huge. Another challenge lies in the fact that it is intractable to collect and model all factors that may affect each individual customer's behavior. On the other hand, the problem is not totally out of reach. A large chain store can generate millions of transactions everyday. Large sets of transaction records are always available for data mining systems to achieve adequate predictive accuracy for practical use.

In 2001, we had a chance to collaborate with a local retail supermarket to develop a personalized shopping recommender. After surveying a variety of technologies and available data, we agreed a specification of the recommender. The specification requires that for each customer, the recommender should produce a ranked list of items in the order of his/her preference, given his/her historical shopping record. The goal of this specification is to use accurate ranked lists for individual customers to support various marketing decision making in addition to personalized recommendation. Though we also had customers' demographic data, containing their age, gender, address, income level, etc., we decided to use historical shopping records only because customers rarely provided correct data.

Initially, we tried to directly apply an existing data mining technique: association rule mining (Agrawal & Srikant, 1994). Though association rule mining was originally developed for mining retailer transaction data, very few useful association rules could be discovered for shopping recommendation. Examining the data revealed that this is because the data set was extremely skewed and sparse. The data set was *skewed* in the sense that a large portion of sales is concentrated in a small number of items. The data set was also very *sparse*, in the sense that each customer only purchased a very small subset of the entire set of available items. This is typical for transaction data in retailer stores. Our collaborator suspected that this is partly because certain items become highly popular in a short period of time due to discounts, holidays and other campaign events. We removed the shopping records of those items from the data set. However, the resulting data set was still skewed and sparse and the number of discovered association rules was further reduced.

Then we applied a variety of collaborative filtering techniques, which are known to handle sparse data better. We achieved some preliminary results, but those techniques usually assume that a database of preference ratings from customers is given. The translation of transaction data into preference ratings usually depends on unreliable heuristics. To address the problem at hand, we cast previous work in collaborative filtering in a probabilistic framework. That allows us to analytically compare different techniques and understand their underlying assumptions. Based on the results of the analysis, we developed HyPAM (Hybrid Poisson Aspect Modelling), an approach that learns a probabilistic graphical model from historical transaction data. Given a customer's shopping record, the learned model can predict his/her future shopping preference by estimating the probability that he/she is interested in a given item without a database of preference ratings.

Meanwhile, we tried several different metrics to empirically evaluate HyPAM and other recommenders. Again, many metrics for evaluating recommenders, such as absolute deviation, are based on ratings and not applicable to shopping recommendation. We adopted two metrics, *rank score* (Breese, Heckerman, & Kadie, 1998) and *lift index* (Ling & Li, 1998) to evaluate ranked lists produced by the recommenders. However, rank score favors a ranked list that predicts one or two items at the top accurately, while lift index favors one with a better overall predictive accuracy. To remedy this discrepancy, we developed a visualization method called the *rank plot* that presents the ranked lists as a scatter plot. With the rank plots, we can visualize the quality of ranked lists from both global and local perspectives. We report the experimental results of HyPAM and two well-known recommendation systems, GroupLens (Resnick et al., 1994) and the recommender in IBM SmartPad system (Lawrence et al., 2001). We selected to present the results of these two methods because they are representatives of the collaborative filtering methods and data mining methods, respectively. Results show that HyPAM outperforms GroupLens and the IBM method by a large margin.

In addition to the data set from our collaborator, we obtained another data set from a different retailer chain store to ensure that our finding is not a special case in one transaction data set. Due to the nature of this store, the data set is even more skewed and sparse. We repeated the experiment with this data set and obtained similar results. We will report both results together in this paper.

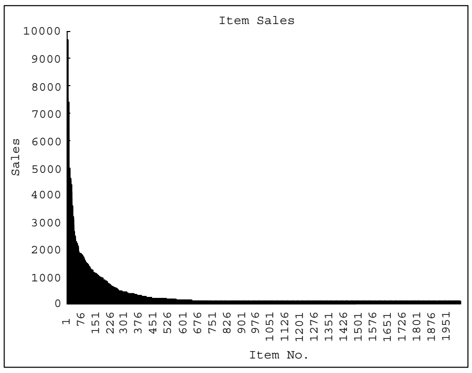
This paper reports the lessons that we learned during the process. We learned that, because the transaction data sets are extremely skewed and sparse, extending a collaborative filtering method based on ratings to perform personalized shopping recommendation is not trivial and that it is not appropriate to apply association-rule based methods for large scale prediction of customers' shopping preferences. Instead, a probabilistic graphical model is more effective for handling skewed and sparse data. This is basically established by our experimental results which show that HyPAM outperforms the other two recommenders by generating much more accurate predictions of what items a customer will actually purchase in the unseen test data. We also provide explanation on why this is the case. The remainder of this paper reports how we learned this lesson. Section 2 illustrates the skewness and sparsity of the data sets from two real-world retailer stores. Section 3 reviews previous recommender systems. Section 4 presents how we cast collaborative filtering algorithms in a probabilistic framework. Section 5 presents the derivation of the HyPAM model and how to learn such a model from transaction data. Section 6 reports the experimental results and Section 7 summarizes and reviews the lessons learned.

2. Transaction data

The data sets used in our research are the courtesy of two large local retailer stores, Ta-Feng and B&Q. Ta-Feng is a membership retailer warehouse that sells a wide range of merchandise, from food and grocery to office supplies and furniture. The data set contains shopping records collected in a time span of four months, from November, 2000 to February, 2001. Each record consists of four attributes: the shopping date, customer ID, product ID, and the amount of purchase. Shopping records with the same customer ID and the same shopping date are considered as a transaction. There are 119,578 transactions and 32,266 distinguishable customers in this data set. B&Q is a large international DIY home improvement and garden retailer chain. Our data set is provided by the largest B&Q store in Taipei. This data set contains the transaction data of the fourth quarter in 2001. The attributes of this data set are the same as those of Ta-Feng data set. There are totally 1,120,193 transactions and 607,064 customers. Both stores adopt a common commodity classification standard that consists of a three-level product taxonomy. Products of Ta-Feng are classified into 201 product classes and 2012 sub-classes, while B&Q has 146 classes and 577 subclasses. Table 1 summarizes the statistics of the data sets.

Table 1. Statistics of the data sets for the experiment.

Data set	Total transactions	Total customers	Mean/Median items per trn.	Class/ Subclass
Ta-Feng	119,578	32,366	6.83/5	201/2012
B&Q	1,120,193	607,064	4.30/3	146/577



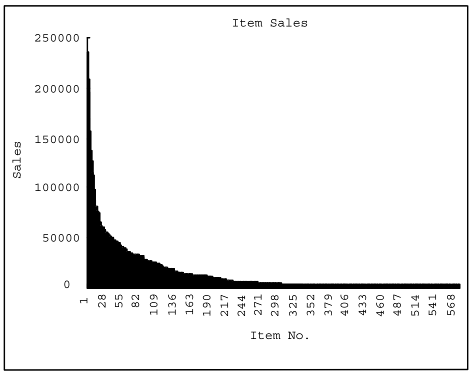


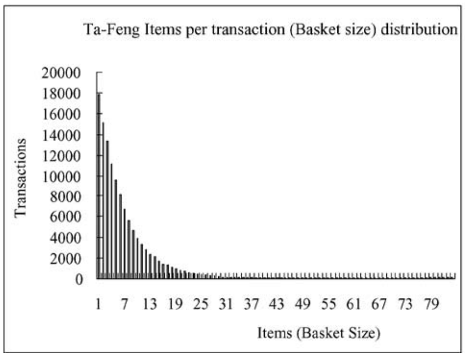
Figure 1. Data skewness of Ta-Feng (left) and B&Q (right). Items are re-numbered by their order of items sold.

Figure 1 shows the skewness of the items sold in Ta-Feng and in B&Q. The sales figures are skewed and concentrated on a very small portion of products. This is an example of "*the 80-20 rule*" known in business management. The skewness of the data makes it easy to accurately recommend a popular item but very difficult for a recommender to identify potential customers for the items in the tail of the curve. Unfortunately, a recommender is supposed to promote the sales of those items.

The sparsity of the data set is illustrated by the distribution of the number of different items in each transaction (i.e., the basket size), which is skewed with a long right tail, as shown in figure 2. The number of transactions declines exponentially with increased number of items. Both Ta-Feng and B&Q data sets are very sparse. The median and mean of the items in a transaction of Ta-Feng data set is 5 and 6.83, respectively. Due to the nature of the merchandise, B&Q data set is sparser than that of Ta-Feng. The median of the number of items in each transaction is only 3 and the mean is 4.30. In a sparse data set, a large number of transactions involve only a small number of items, which implies that the information in each transaction that can be used by a recommender is limited.

3. Review of previous work

Previous work in recommender systems can be classified into two categories: *content-based filtering* and *collaborative filtering* (Goldberg et al., 1992). Content-based filtering recommends users the items similar to those they like before, while collaborative filtering



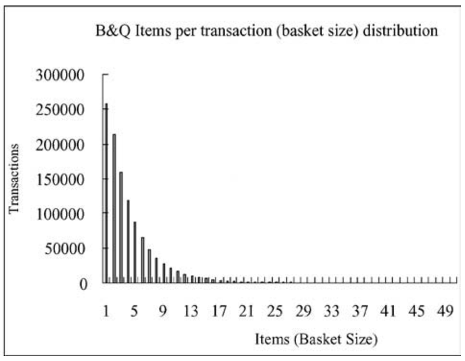


Figure 2. Data sparsity of Ta-Feng (left) and B&Q (right): items per transactions.

utilizes large transaction data sets to reveal the similarity that associates users and items to predict their preference toward a given item. When product information is easy to obtain and contains comparable features (i.e., for book stores or video rentals), content-based filtering can achieve useful results. But in the domain of shopping recommendation, products are usually heterogeneous and incomparable. In contrast, a large amount of transaction data is available for shopping recommendation and hence collaborative filtering is deemed more suitable.

Collaborative filtering can then be further classified into *memory-based* and *model-based* (Breese, Heckerman, & Kadie, 1998). In memory-based approaches, the whole transaction database is used in recommendation. No abstract model is derived from the database. For example, GroupLens (Resnick et al., 1994) and Ringo (Shardanand & Maes, 1995) predict a customer's preference for an item by calculating weighted average of other customers' numerical ratings for the same item. In model-based approaches, models are learned from the database in advance and are queried subsequently for recommendations (see e.g., Billsus & Pazzani, 1998; Breese Heckerman, & Kadie, 1998; Cadez, Smyth, & Mannila, 2001; Hofmann, 1999; Popescul et al., 2001).

Although collaborative filtering can be generally applied to shopping recommendation, most of previous methods are based on ratings from customers. But in shopping recommendation, there is no *explicit rating* (Breese, Heckerman, & Kadie, 1998) in the transaction records and we need to transform them into ratings heuristically. The resulting records are referred to as *implicit ratings* (Breese, Heckerman, & Kadie, 1998). Pennock, Horvitz, and Giles (2000) show an example of such transformation.

More recently, many data mining approaches to shopping recommendation have been published. One prevailing method is to mine association rules (Agrawal & Srikant, 1994) of products and then used the mined rules for recommendation (Brijs et al., 2000; Lawrence et al., 2001). However, since retail transaction databases are usually extremely skewed and sparse, association rule mining may not be appropriate for the problem of predicting customer behavior at the individual level (Apte et al., 2002; Cadez, Smyth, & Mannila, 2001). To address the limitation of association rules, Cadez, Smyth, and Mannila (2001) proposed an alternative probabilistic approach called *predictive profiling*. Predictive profiling is a mixture model based on a linear combination of simple probabilistic models. Unlike predictive profiling, HyPAM models complex relationships *within* and *between* customers and products in a graphical model and therefore can predict heterogeneous customer preferences.

4. Collaborative filtering in a probabilistic framework

In this section, we cast previous work in collaborative filtering in a probabilistic framework, which allows us to understand the underlying assumptions made by different collaborative filtering algorithms and how well these assumptions match the skew and sparse distributions of the transaction data. From a probabilistic viewpoint, we can compare different recommenders so that we can derive a better probabilistic model for personalized shopping recommendation. Meanwhile, we also formally define the problem of recommendation based on ratings, the shopping recommendation problem, and the notations to be used throughout the paper.

4.1. Recommendation based on ratings

Suppose we have a set of I customers (or users), $u_1, \ldots, u_I$, and a set of J items, $m_1, \ldots, m_J$. The items could be products or product classes. In the problem of recommendation based on ratings, we have a database of three-tuples (u, m, r) representing that r is the rating of item m by user u. Note that usually a user only rates a very small portion of all items.

4.1.1. The basic model. Let U, R_j, and A be the random variables representing a user, a rating to item m_j, and a type of relation that associates U and R_j, respectively. Then we can model this problem as a probabilistic graphical model as shown in figure 3, where we have two probabilistically equivalent graphs for the model. An intuitive interpretation of the model is that the relation type A influences the distributions of user U and rating R_j (see the second graph in figure 3). If A is a latent variable, this simple model is equivalent to *the aspect model* (Hofmann, 1999), which has a multinomial latent variable to model the relationships between a customer and his/her rating of an item. The idea is that the distributions of the customers and their ratings are governed by a small number of typical preference patterns represented by a latent *aspect* class, which is the value of A. Since the preference patterns are probabilistic, the aspect model maintains the flexibility that a different preference pattern may apply to a customer when he/she chooses to purchase an item in different situations. For example, shopping for daily groceries is one situation and

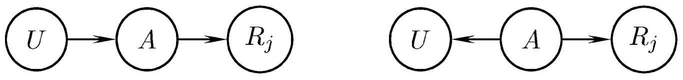

$$\begin{tikzcd} U \arrow[r] & A \arrow[r] & R_j \end{tikzcd} \qquad \begin{tikzcd} U & A \arrow[l] \arrow[r] & R_j \end{tikzcd}$$

Figure 3. Basic model of collaborative filtering for recommendation based on ratings.

preparing a party is another. Popescul et al. (2001) extends the aspect model to incorporate content information (i.e., keywords) for a paper recommendation application.

To predict the rating of u_i for m_j, we use the model to estimate the distribution $P(R_j \mid u_i)$:

$$P(R_j = k \mid u_i) = \sum_A P(R_j = k \mid A) P(A \mid u_i). \tag{1}$$

The predictive rating p_{ij} can be determined by:

$$p_{ij} = \arg\max_k P(R_j = k \mid u_i), \tag{2}$$

where R_j is an integer in an interval. When R_j can be a real, we use

$$p_{ij} = E_{P(R_j|u_i)}(R_j). \tag{3}$$

By specifying how to estimate $P(R_j \mid A)$ and $P(A \mid u_i)$, we can reformulate a collaborative filtering method as an instance of the above basic model. In the remainder of this subsection, we will cast two well-known collaborative filtering algorithms in a unified probabilistic framework by reformulating them as an instance of this basic model.

4.1.2. GroupLens. In addition to the aspect model, we review two other well-known collaborative filtering methods via our framework. Consider GroupLens (Resnick et al., 1994), which predicts p_{ij} by calculating the weighted average ratings from other users. In spite of its simplicity, GroupLens is proved to perform consistently well for a wide range of data sets in experimental studies (Breese, Heckerman, & Kadie, 1998).

Let r_{ij} be the known absolute rating of u_i to m_j. In GroupLens, absolute ratings have to be transformed into relative ratings:

$$\tilde{r}_{ij} = r_{ij} - \frac{1}{|M^i|} \sum_{j' \in M^i} r_{ij'},$$

where $\tilde{r}_{ij}$ denotes the relative rating that u_i gives to m_j, and M^i the set of the indices of the rated items.

The predictive relative rating $\tilde{p}_{ij}$ is calculated by the following equation:

$$\tilde{p}_{ij} = \kappa \sum_{i' \in U^j} \sigma_{ii'} \tilde{r}_{i'j}$$

where κ is a normalizing factor and U^j is the set of the indices of the users who have rated m_j and $\sigma_{ii'}$ is the similarity between u_i and $u_{i'}$:

$$\sigma_{ii'} = \frac{\sum_j \tilde{r}_{ij} \tilde{r}_{i'j}}{\sqrt{\sum_j \tilde{r}_{ij}^2 \sum_j \tilde{r}_{i'j}^2}}$$

To cast GroupLens in the basic model, we treat R_j as the predictive relative rating $\tilde{p}_{ij}$, and A as the users in U^j. Then GroupLens is an instance of the basic model with the distributions defined by:

$$P(A = u_{i'} \mid U = u_i) = \kappa |\sigma_{ii'}|,$$

and

$$P(R_j \mid A = u_{i'}) = \begin{cases} 1: & \text{if } R_j = \tilde{r}_{i'j} \cdot \dfrac{\sigma_{ii'}}{|\sigma_{ii'}|} \\ 0: & \text{otherwise} \end{cases}$$

Since $P(A = u_{i'} \mid U = u_i)$ cannot be negative, we use $\frac{\sigma_{ii'}}{|\sigma_{ii'}|}$ to handle the negative correlation in GroupLens. At last, $\tilde{p}_{ij}$ can be obtained by Eq. (3).

4.1.3. Personality diagnosis. Another approach considered here is *personality diagnosis* (PD) by Pennock, Horvitz, and Giles (2000). They already gave a probabilistic interpretation of PD as a Naive Bayes network. PD and GroupLens differ in the following aspects. First, PD uses absolute ratings and Eq. (2) to make the final prediction. Second, $\sigma_{ii'}$ is calculated purely based on probability. Third, even if u_i has rated m_j, the information is assumed to be noisy.

Figure 4 shows PD by extending the basic model. Variable U is treated as user clusters, although every user forms a cluster in PD. To estimate $P(R_j \mid u_i)$, we use the ratings that u_i has given:

$$P(R_j \mid u_i) = \sum_{i'} P(U = u_{i'} \mid u_i) P(R_j \mid U = u_{i'}) \tag{4}$$

$$= \sum_{i'} \underbrace{P(u_{i'} \mid r_{i1}, \ldots, r_{iJ})}_{(A)} \underbrace{P(R_j \mid u_{i'})}_{(B)}. \tag{5}$$

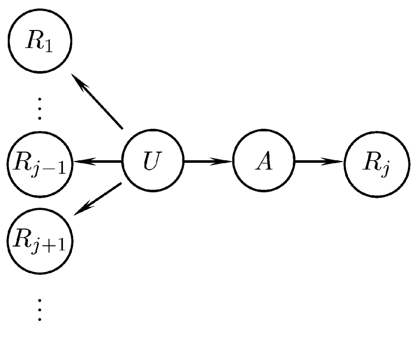


Figure 4. Extended basic model for PD.

The basic model (i.e., (B) in Eq. (5)) is extended by adding a cluster model in (A). The model becomes a Naive Bayes network if we intentionally make node A redundant:

$$P(A = u_{i'} \mid U = u_i) = \begin{cases} 1: & \text{if } i' = i \\ 0: & \text{otherwise} \end{cases} \tag{6}$$

Then, the network behaves as if R_j is connected with U. This completes our casting of PD as an extension of our basic model.

4.2. *Shopping recommendation*

In shopping recommendation, our goal is to predict the relative preference of u_i among items m_j. Let M be the random variable representing a preferred item. We can model shopping recommendation with a model as shown in figure 5. This model is almost identical as the basic model for recommendation based on ratings except that we need one model for each R_j, the rating for item m_j, while the model in figure 5 is for all items.

To estimate the preferences of u_i, we calculate $P(M \mid u_i)$, the probabilities that u_i likes each item:

$$P(M = m_j \mid u_i) = \sum_A P(M = m_j \mid A) P(A \mid u_i). \tag{7}$$

The relative preferences are consequently determined by sorting the probabilities in descending order. Hence, a shopping recommender can be defined by $P(M \mid A)$ and $P(A \mid u_i)$.

Consider the recommender of IBM SmartPad system (Lawrence et al., 2001), which will be referred to as the IBM method. We can cast the IBM method as an instance of our basic model as follows. A three-layer product taxonomy is given as the input. For example, products "7-up six-pack" and "Pepsi six-pack" may be classified in subclass "soda" and "soda" is classified in class "beverage." Suppose there are C subclasses in the input taxonomy, denoted by $g_1, \ldots, g_C$. Customer u_i is described as vector $\mathbf{u}_i$ with C elements such that the c-th element in $\mathbf{u}_i$ is the normalized spending of u_i on product subclass g_c. Product m_j is described as vector $\mathbf{m}_j$, also with C elements, such that the c-th element is the degree of the association between g_j and g_c, where g_j is the subclass to which m_j belongs. The degree of the association is then determined by a heuristic based on whether there exists an association rule between g_c and g_j. The degree takes values from 1, 0.5, 0.25, and 0. Association rules between product subclasses must be mined from the transaction data in advance.

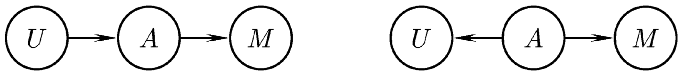


Figure 5. Basic model of collaborative filtering for shopping recommendation.

Given $\mathbf{u}_i$ and $\mathbf{m}_j$, the relative preference s_{ij} of u_i to m_j can be estimated by the cosine of the angle between $\mathbf{u}_i$ and $\mathbf{m}_j$:

$$s_{ij} = \rho_j \frac{\mathbf{u}_i \cdot \mathbf{m}_j}{\|\mathbf{u}_i\| \|\mathbf{m}_j\|}, \tag{8}$$

where ρ_j is a constant indicating how much the store wants to recommend m_j to its customers. If we set ρ_j to be the profit margin of m_j, Eq. (8) is equivalent to recommending products with high expected profit margins.

The probabilistic interpretation of $\mathbf{u}_i$ and $\mathbf{m}_j$ is as follows. Let $P(g_c \mid u_i)$ be the probability that u_i prefers g_c. $\mathbf{u}_i$ can be regarded as $[P(g_1 \mid u_i), \ldots, P(g_C \mid u_i)]^T$, and $P(g_c \mid u_i)$ is estimated by the normalized spending on g_c. Note that the two-step normalization process in the IBM method does not guarantee that the elements in $\mathbf{u}_i$ always sum to one. We can normalize $\mathbf{u}_i$ again to satisfy this restriction without affecting Eq. (8). Similarly, we can also normalize $\mathbf{m}_j$ and treat it as the vector $[P(g_1 \mid m_j), \ldots, P(g_C \mid m_j)]^T$, where $P(g_c \mid m_j)$ denotes the probability that a customer prefers g_c given that he/she prefers m_j.

To cast the IBM method in our basic model, let variable A stand for a subclass that takes values from g_1 to g_C. Then we have:

$$P(m_j \mid u_i) = \sum_c P(A = g_c \mid u_i) P(m_j \mid A = g_c) \tag{9}$$

$$\propto \sum_c P(A = g_c \mid u_i) P(A = g_c \mid m_j), \tag{10}$$

where $P(A = g_c \mid u_i)$ and $P(A = g_c \mid m_j)$ can be looked up in $\mathbf{u}_i$ and $\mathbf{m}_j$. Rewriting (9) to (10) is justified by assuming a uniform distribution of g_c, which implies that $P(g_c \mid m_j)$ is proportional to $P(m_j \mid g_c)$. Obviously, this assumption implicitly made by the IBM method is not valid because subclass preference is very skewed and far from uniform.

4.3. Discussion

We have reviewed four well-known collaborative filtering recommenders: the aspect model (Hofmann, 1999), GroupLens (Resnick et al., 1994), PD (Pennock, Horvitz, and Giles, 2000) and the IBM method (Lawrence et al., 2001) from a probabilistic viewpoint.

Previously, an algorithmic framework for collaborative filtering has been proposed in the work of Herlocker et al. (1999) for similarity-based approaches, but no probabilistic framework has been proposed. From this review, we conclude that, regardless whether they are based on ratings or for shopping recommendation, basically, these recommenders can all be cast as instances of the three-node probabilistic graphical model as depicted in figures 3 and 5. An exception is PD, for which we need to extend our basic model with a naive Bayes cluster model to model user groups.

The conclusion implies that the difference among these recommenders is how they estimate the parameter probabilities. How these recommenders treat the relation type, represented by node A in the middle of the model, plays an important role in differentiating them. The aspect model treats A as a latent variable and learns the probabilities using the

EM algorithm (Dempster, Laird, & Rubin, 1977). GroupLens uses correlation and treats A as the group of the users who have rated a given item. PD uses a cluster model to model user groups, but A plays no role in recommendation. The IBM method also uses correlation and treats A as given (i.e., subclasses). Among them only the IBM method is designed for shopping recommendation.

Different treatments of A not only affect the predictive accuracies but also the time required to use a recommender. The more the possible states that A has, the longer the required time. Therefore, GroupLens and the IBM methods may spend relatively longer time to generate recommendations.

5. Hybrid Poisson Aspect Model

Based on our comparative review of collaborative filtering, we derive a new model called HyPAM (Hybrid Poisson Aspect Model), a probabilistic model for shopping recommendation. This section describes how to provide recommendation and how to learn the parameters from transaction data for HyPAM.

5.1. Model specification

HyPAM applies the cluster model to cluster customers and the aspect model to model the relationships between customer clusters and products. Figure 6 shows the graphical representation of HyPAM. In the dotted box is the cluster model and in the dashed box the aspect model. This model is similar to the PD model of collaborative filtering in figure 4. Unlike PD, in HyPAM, A is a latent variable as in the aspect model and M represents the relative preference rather than the rating.

5.1.1. Cluster model. A customer is represented by a set of F random variables $(X_1, \ldots, X_F)$ where X_f represents the event that the customer has purchased X_f units of the f-th item in a given period of time T. T can be a basket, holidays, a long weekend, a quarter, etc., depending on the recommendation application at hand. The items considered here can be a product or a set of products. In our application in retailer stores, an item represents a subclass in a three-layer taxonomy as in the IBM method.

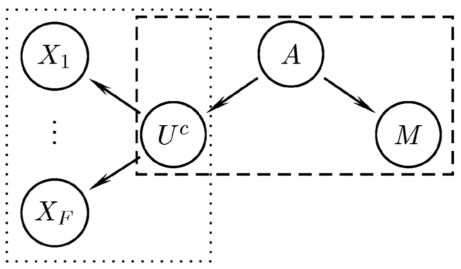


Figure 6. Graphical representation of HyPAM.

The cluster model partitions the customers into L clusters. U^c represents a customer cluster with value u_l^c denoting the event that the customer is in the l-th cluster. The cluster model provides generalization that allows HyPAM to recommend a new customer items that he/she never purchased.

Each cluster determines the distribution $P(X_f \mid u_l^c)$, which is modeled by the *Poisson distribution*. That is,

$$P\bigl(X_f \bigm| u_l^c\bigr) = P(X_f \mid \lambda_{lf}) = e^{-\lambda_{lf}} \frac{(\lambda_{lf})^{X_f}}{X_f!}, \tag{11}$$

where λ_{lf} is the only parameter for the Poisson distribution. Note that λ_{lf} is also the expectation as well as the variance. Therefore, λ_{lf} can be interpreted as the expected number that the f-th item has been purchased given that the customer belongs to cluster u_l^c. Hence, each cluster u_l^c can be described by $(\lambda_{l1}, \ldots, \lambda_{lF})$. We can also regard u_l^c as a typical customer who has bought λ_{lf} units of item f.

Now we describe how to estimate $P(u_i \mid u_l^c)$, the probability that customer u_i belongs to cluster u_l^c. Recall that $(x_{i1}, \ldots, x_{iF})$ represents a customer u_i. Since the model assumes that $X_1, \ldots X_f$ are conditionally independent given U^c (i.e., the "naive Bayes" assumption), we have

$$P\bigl(u_i \bigm| u_l^c\bigr) = \prod_{f=1}^{F} P\bigl(X_f = x_{if} \bigm| u_l^c\bigr). \tag{12}$$

Here, $x_{if} = 0$ if u_i has never purchased item f. As a result, it may appear to the model that the customer is uninterested in f, but this is seldom the case. More often this is simply because the customer has not noticed this item. A good recommender is supposed to recommend such item if there is evidence showing the potential to match the customer with the item. This is critical for a recommender to handle skewed transaction databases. Therefore, we treat the case $x_{if} = 0$ as if we have not observed that u_i purchases item f. This is realized by replacing Eq. (12) by:

$$P\bigl(u_i \bigm| u_l^c\bigr) = \prod_{\{f \mid x_{if} \neq 0\}} P\bigl(X_f = x_{if} \bigm| u_l^c\bigr). \tag{13}$$

5.1.2. Aspect model. The aspect model partitions the co-occurrence data over $\mathcal{U}^c \times \mathcal{M}$ into K aspects, denoted by $a_1, \ldots, a_K$. The aspect model is similar to (Hofmann, 1999) but we have customer clusters instead of individual customers to relate to an item to be recommended. Section 4.1.1 describes the aspect model in details.

5.2. Training HyPAM

We apply the Expectation Maximization (EM) algorithm (Dempster, Laird, & Rubin, 1977) to learn the parameters of HyPAM because HyPAM contains two hidden nodes, U^c and A. Figure 7 shows all the parameters for HyPAM.

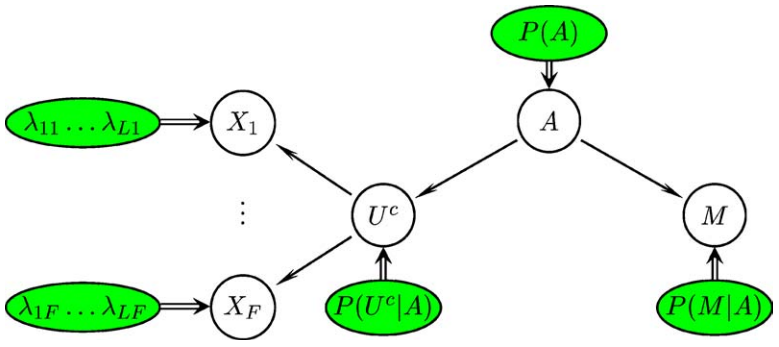

$P(A)$
$\lambda_{11}\dots\lambda_{L1}$
X_1
A
U^c
M
$\lambda_{1F}\dots\lambda_{LF}$
X_F
$P(U^c|A)$
$P(M|A)$

Figure 7. HyPAM and its parameters.

Let D be the training data set extracted from the transaction database, and $d^1, \dots, d^N$ be the instances in D. Each d^n must have the form $(\mathbf{x}^n, m^n)_{1\le n\le N}$, where N is the number of total transactions and m^n is the item. $\mathbf{x}^n = (x_1^n, \dots, x_F^n)$ indicates that customer u_n bought item 1 to item F during the data collection period, and the value of x_i^n represents the amount of purchases.

To apply the EM algorithm, suppose that we have a data set Z that contains complete training data with all the hidden values for the latent variables of HyPAM. That is, each instance in Z is of the form $z^n = \{d^n, a^n, u^{cn}\}$, for $n = 1, \dots, N$. The EM algorithm consists of two steps: E-step computes the distributions for a^n and u^{cn} and M-step re-estimate new parameters for HyPAM. Appendix gives the detailed derivation of the EM algorithm for training HyPAM.

5.3. *Using the Hybrid Poisson Aspect Model*

5.3.1. Recommendation. With the trained HyPAM model, we can solve the personalized shopping recommendation problem by estimating the preferences of the items for a given customer u, represented by his/her historical shopping list $(x_1, \dots, x_F)$. More precisely, we can provide a ranked list of items m_j to u, ordered by probability $P(m_j \mid u)$, the preference of u toward item m_j. $P(m_j \mid u)$ can be easily estimated from the trained HyPAM model:

$$\begin{aligned} P(m_j \mid u) &\propto P(m_j, u) \\ &= \sum_{k=1}^{K}\sum_{l=1}^{L} P(a_k)P(m_j \mid a_k)P\big(u_l^c \mid a_k\big)P\big(u \mid u_l^c\big) \\ &= \sum_{k=1}^{K}\sum_{l=1}^{L} \big[P(a_k)P(m_j \mid a_k)P\big(u_l^c \mid a_k\big) \\ &\quad \prod_{\{f \mid x_f \neq 0\}} P\big(X_f = x_f \mid u_l^c\big)\big]. \end{aligned} \tag{14}$$

5.3.2. Supporting marketing decision making. In addition to recommendation, we can query the HyPAM model to forecast customers' preference to support marketing decision making. Here are some examples:

– What is the distribution of the user clusters?

$$P\left(u_l^c\right) = \sum_{k=1}^{K} P\left(u_l^c \mid a_k\right) P(a_k)$$

– What are the most popular items?

$$\arg\max_j P(m_j) = \arg\max_j \sum_{k=1}^{K} P(m_j \mid a_k) P(a_k)$$

– What are the most popular items in cluster u_l^c?

$$\begin{aligned}\arg\max_j P\left(m_j \mid u_l^c\right) &= \arg\max_j \frac{P\left(m_j, u_l^c\right)}{P\left(u_l^c\right)} \\ &= \arg\max_j \frac{\sum_{k=1}^{K} P(a_k) P(m_j \mid a_k) P\left(u_l^c \mid a_k\right)}{\sum_{k=1}^{K} P(a_k) P\left(u_l^c \mid a_k\right)}\end{aligned}$$

6. Experimental evaluation

In this section, we report our experimental evaluation of HyPAM and two well-known recommenders, GroupLens and the IBM method, with real-world large transaction data sets from two different retailer stores described in Section 2.

6.1. Evaluation methodology

Given the historical shopping list of a customer, the recommenders are supposed to output a ranked list of items, sorted by the predicted preference, for the customer. We apply the *Given n* and *All but one* protocols as explained below to divide the historical shopping lists as the input to the recommenders. We then apply two widely applied metrics, *rank score* and *lift index*, to evaluate the performance of the recommenders in this experiment. These two metrics measure the quality of a given ranked list against a list of items that the customer actually purchased. Other metrics such as log-likelihood used in Cadez, Smyth, and Mannila (2001) is not applied here because of their inability to measure the quality of a ranked list. This subsection describes the metrics and protocols.

6.1.1. Protocols. In our experiment, we divide the customers into a training set and a test set. The transaction data of the customers in the training set is used to train the recommenders. Each trained recommender then generates a ranked list for each customer in the test set according to the customer's shopping list.

For each target customer in the test set, we apply two types of protocols to divide his/her shopping list into a given set and a test set of items to simulate situations with differing numbers of purchases available to the recommenders (Breese, Heckerman, & Kadie, 1998). In the first protocol, we withhold a randomly selected purchase for each target customer and the remainders are treated as given. Next we can evaluate how accurate the recommender is by how the recommender ranks the withheld item. This protocol is called *All but one*. We can also randomly preserve 2, 5, or 10 purchases from the shopping list as given and see if the recommender can accurately predict the remainder in the ranked list. These are referred to as *Given 2*, *Given 5*, and *Given 10* protocols.

The *All but one* protocol measures the performance of the algorithms when given as much information as possible about a target customer. The various *Given n* protocols examine the performance of the recommenders when we only know a small number of purchases of a customer. Note that in the cases applying *Given n* protocol, only those customers with more than *n* purchases will be selected to the test set.

6.1.2. Rank score. Breese, Heckerman, and Kadie, originally for a Web page recommendation application, define the expect utility of a ranked list of items for a customer as:

$$\mathcal{R}_i = \sum_j \frac{\delta(i, j)}{2^{(j-1)/\alpha}},$$

where j is the rank of an item in the ranked list generated by the recommender, $\delta(i, j)$ is 1 if user i accessed item j in the test set and 0 otherwise, and α is the *viewing half-life*, which is the rank of an item such that it has a 50% chance of being viewed or purchased. We use a constant $\alpha = 10$ in our experiment. We can normalize the utilities with $\mathcal{R}_i^{\max}$, the maximum possible utility obtained when all items that user i has accessed appear at the top of the ranked list:

$$\mathcal{R}'_i = \frac{\mathcal{R}_i}{\mathcal{R}_i^{\max}}.$$

Note that if the recommender considers that the customer has the same preference for more than one items, then these items will have the same rank in the ranked list.

6.1.3. Lift index. Another evaluation metric is *lift index* (Ling & Li, 1998). After the recommender generates the ranked list, we divide the list into ten equal deciles, and see how the items accessed by the user in the test data set distribute in the ten deciles. Clearly, the more items appear in the top deciles, the better this ranked list is. Note that in different protocols, a different number of items will be withheld and the withheld items are not in the ranked list. As a result, the lengths of rank lists for different users or different protocols are different and items with the same rank in different ranked list may be in different deciles.

The lift index is defined as the weighted sum of the number of accessed items that appear in the ten deciles. In our work, an accessed item means a purchased product. Let $S_1, S_2, \ldots, S_{10}$ be the number of the accessed items in each decile (ordered). The *lift index* is defined by

$$S_{\text{lift}} = \frac{1 \times S_1 + 0.9 \times S_2 + \cdots + 0.1 \times S_{10}}{\sum_{i=1}^{10} S_i}.$$

6.2. *Comparison*

This subsection reports the experimental results of three recommenders, GroupLens, the IBM method, and HyPAM. For each data set and each protocol, we randomly selected 15,000 customers from both data sets to train the recommenders and a disjoint set of 1,000 customers as the test set. The performance is then evaluated by the two metrics under different protocols.

6.2.1. Preprocessing. To train the recommenders, we converted the records in the transaction databases into the form $((x_1^n, \ldots, x_F^n), m^n)_{1 \leq n \leq N}$ as specified in Section 5.2. In this case, we have $F = 2,012$ subclasses for Ta-Feng data set and 577 for B&Q, respectively (see Table 1). This data preprocessing step can be accomplished using some simple SQL commands to the transaction database.

HyPAM is tested with four combinations of the possible states L and K for its latent variables U^c, the customer clusters, and A, the aspects. We use Ll_Kk to denote a combination. For example, L20_K10 denotes a HyPAM model using 20 clusters and 10 aspects.

Since GroupLens is originally designed for recommendation based on ratings, we assume that the more a customer buys the products in a subclass, the more he/she likes it. However, if the products in a subclass has never been bought, we regard the subclass as unrated. For others, their ratings are obtained from the transaction data, ranging from 1 to 10. Our rule is: if a customer purchases r products in the same subclass, the rating is r for $0 < r \leq 10$ and 10 if r exceeds ten. The rule is determined heuristically by observing the transaction data where the amount of purchases in a subclass is usually less than ten. The IBM method requires association rules mined from the data sets in advance. The support and confidence thresholds (Agrawal & Srikant, 1994) for Ta-Feng data set are 50 and 0.2, respectively and for B&Q data set are 25 and 0.1, respectively. The thresholds were selected empirically such that a sufficient but not exceedingly large set of association rules can be mined from the data sets.

We also report the results of "the default method," which always recommends the best selling items to all customers. More precisely, the default method always generates a ranked list that is actually a list of items ordered by their historical records of the items sold. As the transaction data sets are very skewed, as shown in figure 1, it is likely that a customer will purchase many items at the top of the list. Therefore, the performance of the default method can serve as the baseline of a personalized shopping recommender.

Table 2. Experimental result—Ta-Feng data set.

Algorithms	Given 2	Given 5	Given 10	All but 1
(a) Rank score, Training set size: 15000, Test set size: 1000				
GroupLens	0.0284	0.0295	0.0302	0.0240
IBM	0.0984	0.0985	0.0802	0.0715
Default	0.2567	0.2406	0.2272	0.1937
L10_K10	0.2637	0.2467	0.2352	0.2958
L10_K20	0.2637	0.2470	0.2354	0.2939
L20_K10	0.2637	0.2467	0.2355	0.2949
L20_K20	0.2635	0.2467	0.2351	0.2981
(b) Lift index, Training set size: 15000, Test set size: 1000				
GroupLens	0.8590	0.8392	0.8239	0.8726
IBM	0.5056	0.6658	0.6804	0.7297
Default	0.9353	0.9253	0.9173	0.9442
L10_K10	0.9355	0.9261	0.9177	0.9886
L10_K20	0.9355	0.9260	0.9176	0.9884
L20_K10	0.9356	0.9260	0.9178	0.9884
L20_K20	0.9356	0.9261	0.9178	0.9884

6.2.2. The results. Table 2 shows the experimental results of rank scores and lift index scores for Ta-Feng data set. HyPAM outperforms the other two recommenders significantly in all combinations of experimental setting by a large margin. Table 3 shows the results for B&Q data set. HyPAM also outperforms the other two recommenders by a large margin. The sparsity of B&Q data set affects performance especially for GroupLens and the IBM method. It is not surprising that the default method performs well because of the skewness of the data set. HyPAM is the only recommender in this experiment that can outperform the default method. This shows that HyPAM not only recommends popular items but also identifies potential customers of unpopular items.

In the results for B&Q, the rank scores of HyPAM with *All but one* protocol are significantly lower than that for *Given n*. This may be due to the fact that in the cases applying *Given n* protocol, only those customers with more than n purchases are selected to the test set. Since for B&Q the average purchasing is very low (=5.26), the test set of *All but one* protocol contains many customers who purchased only two items, while the customers in the test sets for *Given n* purchased at least three items. The short shopping lists also affect the default method. Also remarkable is that the lift index scores of IBM are very low because not many association rules between subclasses can be found in B&Q data set. Consequently, many items are predicted as impossible to buy. We have assigned low support and confidence thresholds for this data set but without much improvement.

Both rank scores and lift index scores are not sensitive to the number of clusters and aspects of HyPAM, with almost identical results for all combinations. Experiments with a wider range of combinations may be required to determine their impact.

6.2.3. Rank plots. The IBM method achieves better rank scores but worse lift index scores than GroupLens. Examining the ranked lists by the different methods reveals that the IBM method can hit one or two items higher than the best hit by GroupLens but rank other items at low deciles. To remedy this discrepancy, we proposed a new visualization method called *rank plots* to visually compare the ranked lists as scatter plots. Figure 8 shows the rank plots of the experimental results described in this section. Each actual purchase (i.e., hit) is presented by a spot at its position in the ranked list. The x-axis represents customer ID and the y-axis represents rank. The more the spots concentrate at the top, the better the ranked lists. We also colored the spots at the top twenty red, the next forty orange, etc., to indicate which intervals they are in. The colored rank plots for all experimental results are available on the World Wide Web: `http://chunnan.iis.sinica.edu.tw/hypam/visualization/`. As we can see, most of actually purchased items are ranked in top ten percent of all items by HyPAM, but their rankings by the other two recommenders spread all over the plots. Scatter plots for the experiments under other protocols look similar to these ones. To sum up, the rank plots clearly reveal the superiority of HyPAM over the other two recommenders.

6.2.4. Miscellanies. Table 4 shows the ranked lists for two randomly picked customers of Ta-Feng and B&Q, respectively. The hit positions of the test items, that is, the actual purchases, are provided. The first example is the ranked lists by three recommenders for a customer in Ta-Feng data set. This customer has purchased nine items. The example shows

Table 3. Experimental result—B&Q data set.

Algorithm	Given 2	Given 5	Given 10	All but 1
(a) Rank score, Training set size: 15000, Test set size: 1000				
GroupLens	0.0162	0.0147	0.0150	0.0123
IBM	0.0708	0.0914	0.0905	0.1007
Default	0.2147	0.2245	0.2413	0.1840
L10_K10	0.2503	0.2551	0.2697	0.1882
L10_K20	0.2502	0.2548	0.2694	0.1882
L20_K10	0.2501	0.2545	0.2691	0.1885
L20_K20	0.2503	0.2547	0.2695	0.1881
(b) Lift index, Training set size: 15000, Test set size: 1000				
GroupLens	0.7438	0.7482	0.7479	0.7220
IBM	0.1874	0.2557	0.3376	0.2843
Default	0.8804	0.8869	0.8784	0.8811
L10_K10	0.8953	0.9002	0.8916	0.8894
L10_K20	0.8954	0.9004	0.8916	0.8896
L20_K10	0.8957	0.9005	0.8920	0.8894
L20_K20	0.8956	0.9005	0.8918	0.8892

the results of *Given 5* protocol. The second example is the *Given 2* results for B&Q data set. In this case, the IBM method suffered from sparse data, and thus, failed to rank five items. GroupLens also failed to rank one item because no other customer who has purchased this item also purchased any withheld one in this case. In contrast, HyPAM provided better ranked lists, even for the items not ranked by other two methods.

In terms of the computational cost, Table 5 reports the time spent for training and testing by the recommenders for both data sets. The training time of HyPAM differs according to

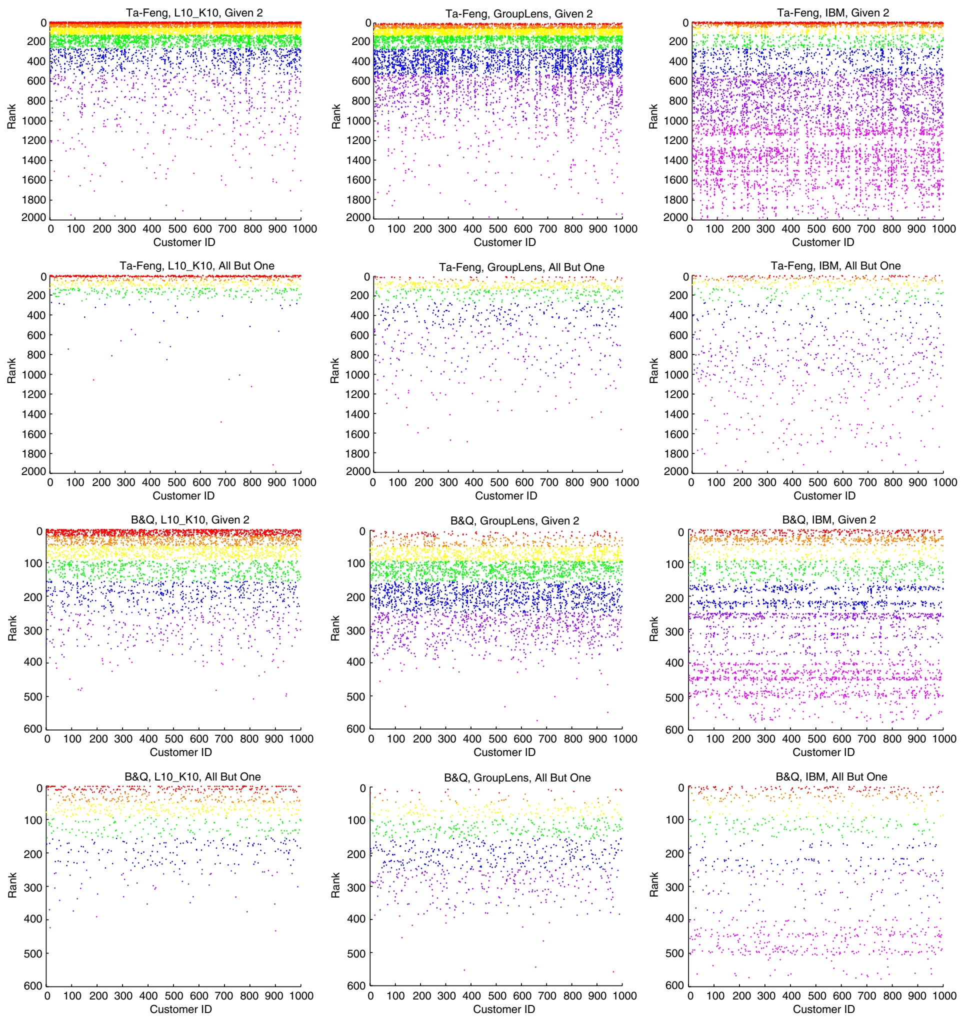


Figure 8. Rank plots of HyPAM L10_K10 (left), GroupLens (center) and IBM (right) for Ta-Feng data set (upper two rows) and B&Q data set (bottom two rows) under protocols Given 2 (first and third rows) and All-but-one (second and fourth rows).

Table 4. Examples of ranked lists for individuals.

Item	GroupLens	IBM	L10_K10
(a) Data set: Ta-Feng, Customer: 000xxxx69, Total Buy: 9, Protocol: Given 5: chewing gum, cookies, nori(dry seaweed), candy and salad oil			
Rice	105	1029	7
Soybean source	238	708	17
Sweet rice balls	47	922	26
Gourmet gift set	294	983	252
Rank Score	0.012	1.44e-22	0.32
Lift Index	0.9	0.6	0.95
(b) Data set: B&Q, Customer: A10xxxxx38, Total Buy: 8, Protocol: Given 2: exterior paint and filler/putty			
Paint brushes	110	12	4
Tap/pipe repair accessory	179	n/a	11
Insect repellant	227	n/a	70
Compost	262	n/a	82
Hammer	221	n/a	130
Outdoor furniture accessory	n/a	n/a	401
Rank Score	0.0001	0.09	0.26
Lift Index	0.62	0.25	0.85

Table 5. Training and testing time.

	Data set	GroupLens	IBM	HyPAM
Training	Ta-Feng	0	20 min	15–40 min
	B&Q	0	40 min	5–10 min
Testing	Ta-Feng	2.5 sec	3.0 sec	0.3 sec
	B&Q	0.3 sec	0.5 sec	0.08 sec

the possible states of the latent variables. It takes less time to train a L10_K10 HyPAM but more for L20_K20. The IBM method can be trained in about 30 minutes including the time spent for association rule mining. GroupLens is a memory-based method that requires no training. However, both GroupLens and the IBM method require more time to generate a ranked list for a customer while it takes less than 0.3 second for HyPAM. This is because implicitly, both GroupLens and the IBM method have a large number of possible states for their latent variables, as discussed in Section 4.3. The experiment was performed on a Pentium 600 MHz notebook PC with 192 MB of memory. We have tried our best to optimize our implementation of these recommenders, but room for improvement is always there.

7. Lessons learned

In this paper, we reported our experience in data mining skewed and sparse transaction data sets to predict individual customers' shopping preferences for two large retailer stores. We learned from our experiments that the collaborative filtering methods such as GroupLens and the association-rule based method such as the IBM method can generally be applied to this problem but rarely produce satisfying results. They perform worse than the default method that simply uses the list of best selling items as its prediction. We also learned the lessons from the implicit assumptions made by GroupLens and the IBM method. These assumptions are revealed when we cast them in a probabilistic framework. The lessons led us to derive HyPAM, a probabilistic graphical model to address the issues of data skewness and sparsity.

We summarize the lessons learned as follows.

- GroupLens:
  - GroupLens recommends items based on ratings that can distinguish whether a user's preference toward an item is high, low or unknown. However, this is not readily available in transaction data and translating the transactions to "implicit" ratings is rarely fruitful.
  - In the probabilistic model of GroupLens, node A represents the group of the users who purchased or rated the item for which we intend to predict a user's preference. Due to data skewness, this group may not be sufficiently large for most items and could be empty in many cases. Also, node A has as many possible states as the number of items, slowing down the efficiency of prediction (see Section 4.3 and 6.2.4).
- IBM SmartPad:
  - The IBM method depends on association rules but we can hardly mine sufficient association rules from a sparse data set and the mined rules are mostly among a small portion of popular items due to the skewness of the data (see Section 6.2.1).
  - The IBM method divides the degree of association into four values and thus provides limited generalization. As a result, the IBM method considers a large number of items as impossible to buy for many customers (see Section 4.2).
  - The IBM method implicitly assumes that customers' preferences for different subclasses of items are uniformly distributed, which is not valid for skewed transaction data (see Eq. (10)).
- HyPAM:
  - HyPAM combines the cluster model and the aspect model to provide generalization that allows HyPAM to recommend a new customer items that he/she never purchases (see figure 6).
  - HyPAM uses the Poisson distribution to model $P(X_f \mid u_l^c)$, the distribution of the purchases for an item given a customer cluster (see Eq. (11)). The Poisson distribution

is an appropriate choice because shopping behavior of an individual customer appears to be governed by a Poisson process. In fact, the Poisson distribution has been applied to model shopping behavior in marketing research for years (see, e.g., Wedel & Kamakura, 1999).

- To deal with data sparsity, instead of modelling items never purchased as uninterest, HyPAM models those items as "unseen" (see Eq. (13)). As a result, it is less likely that HyPAM will consider an item as impossible to buy and a small set of purchase records can be sufficient for HyPAM to identify potential customers.

– Evaluation:

- We found that widely applied metrics for recommenders *rank scores* and *lift index* may produce discrepant results and proposed to use *rank plots* to visually evaluate performance of recommenders (see Section 6.2.3).

The ultimate metric of a recommender is whether it can boost sales. Ability to accurately predict customers' preference is critical for achieving the goal, but a well-crafted campaign strategy with the recommender is also critical. It is also unknown that how accurate a recommender must achieve is sufficient for practical use. These questions remain to be answered by close collaboration between data mining and marketing teams.

Appendix

This appendix presents the detailed derivation of the EM algorithm for training HyPAM. Refer to Section 5.2 and figure 7 for the notation.

– *E-step*: Let Θ be the current parameter set, and Θ' the set for the next iteration. E-step is to compute the probability $P(a^n = a_k, u^{cn} = u_l \mid \mathbf{x}^n, m^n; \Theta)$. Let P_Θ denote the distributions based on Θ. We have

$$\begin{aligned}
&P\big(a^n = a_k, u^{cn} = u_l^c \,\big|\, \mathbf{x}^n, m^n; \Theta\big)\\
&\quad = \frac{P_\Theta(a_k, u_l, \mathbf{x}^n, m^n)}{P_\Theta(\mathbf{x}^n, m^n)}\\
&\quad = \frac{P_\Theta(a_k)P_\Theta\big(u_l^c \,\big|\, a_k\big)P_\Theta(m^n \mid a_k)P_\Theta(\mathbf{x}^n \mid u_l)}{\sum_{k'=1}^{K}\sum_{l'=1}^{L} P_\Theta(a_{k'})P_\Theta\big(u_{l'}^c \,\big|\, a_{k'}\big)P_\Theta(m^n \mid a_{k'})P_\Theta(\mathbf{x}^n \mid u_{l'})},
\end{aligned}$$

where $P_\Theta(\mathbf{x}^n \mid u_l)$ can be estimated by Eq. (13).

For the sake of conciseness, in the following derivation, we use p_{kl}^n to abbreviate $P(a^n = a_k, u^{cn} = u_l \mid \mathbf{x}^n, m^n; \Theta)$, p_k^n for $P(a^n = a_k, \mid \mathbf{x}^n, m^n; \Theta) = \sum_{l=1}^{L} p_{kl}^n$, and p_l^n for $P(u^{cn} = u_l \mid \mathbf{x}^n, m^n; \Theta) = \sum_{k=1}^{K} p_{kl}^n$.

– *M-step*: M-step maximizes $Q(\Theta'; \Theta)$ to obtain the new parameters. $Q(\Theta'; \Theta)$ can be derived as follows:

$$\begin{aligned} Q(\Theta'; \Theta) &= E[\log P(Z; \Theta') \mid D; \Theta] \\ &= \sum_{n=1}^{N} E[\log P(z^n; \Theta') \mid d^n; \Theta] \\ &= \sum_{n=1}^{N} \sum_{k=1}^{K} \sum_{l=1}^{L} p_{kg}^n \log P_{\Theta'}\big(\mathbf{x}^n \mid u_l^c\big) P_{\Theta'}\big(u_l^c \mid a_k\big) P(a_k) P(m^n \mid a_k) \\ &= \sum_{n=1}^{N} \sum_{k=1}^{K} \sum_{l=1}^{L} p_{kg}^n \big[\log P_{\Theta'}\big(\mathbf{x}^n \mid u_l^c\big) + \log P_{\Theta'}\big(u_l^c \mid a_k\big) \\ &\quad + \log P(a_k) + \log P(m^n \mid a_k) \big]. \end{aligned}$$

To obtain Θ', we compute the derivatives of every parameter. In the following equations, Θ' is omitted because all the distributions are based on it.

– λ_{lf}: recall that $P(X_f \mid U_l^c) = e^{-\lambda_{lf}} \frac{\lambda_{lf}^{X_f}}{X_f!}$, and we substitute $P_{\Theta'}(\mathbf{x}^n \mid u_l^c)$ with Eq. (13):

$$\begin{aligned} \frac{\partial Q(\Theta'; \Theta)}{\partial \lambda_{lf}} &= \sum_{\{n \mid x_f^n > 0\}} \sum_{k=1}^{K} p_{kl}^n \left(-1 + \frac{x_f^n}{\lambda_{lf}} \right) = 0 \\ \Longrightarrow \lambda_{lf} &= \frac{\sum_{\{n \mid x_f^n > 0\}} p_l^n x_f^n}{\sum_{\{n \mid x_f^n > 0\}} p_l^n}. \end{aligned} \tag{15}$$

– $P(a_k)$: add a Lagrange multiplier α in the derivative:

$$\begin{aligned} &\frac{\partial}{\partial P(a_k)} \left[Q(\Theta'; \Theta) + \alpha \left(\sum_{k'=1}^{K} P(a_{k'}) - 1 \right) \right] = 0 \\ &\Longrightarrow P(a_k) = \frac{1}{N} \sum_{n=1}^{N} \tilde{q}_k^n. \end{aligned} \tag{16}$$

– $P(u_l^c \mid a_k)$: add a Lagrange multiplier β:

$$\begin{aligned} &\frac{\partial}{\partial P\big(u_l^c \mid a_k\big)} \left[Q(\Theta'; \Theta) + \beta \left(\sum_{l'=1}^{L} P\big(u_{l'}^c \mid a_k\big) - 1 \right) \right] = 0 \\ &\Longrightarrow P\big(u_l^c \mid a_k\big) = \frac{\sum_{n=1}^{N} p_{kl}^n}{\sum_{n=1}^{N} p_k^n}. \end{aligned} \tag{17}$$

– $P(m_j \mid a_k)$: add a Lagrange multiplier γ:

$$\frac{\partial}{\partial P(m_j \mid a_k)} \left[Q(\Theta'; \Theta) + \gamma \left(\sum_{j'} P(m_{j'} \mid a_k) - 1 \right) \right] = 0$$

$$\Longrightarrow P(m_{j'} \mid a_k) = \frac{\sum_{\{n \mid m^n = m_j\}} p_k^n}{\sum_{n=1}^{N} p_k^n}. \tag{18}$$

Acknowledgments

This research was supported in part by the National Science Council in Taiwan under Grant No. 91-2213-E-001-033. We wish to thank Mr. Ta-Feng Kao, Victor Lin and Eddie Chien for providing the data sets and valuable comments. We also wish to thank anonymous reviewers for their valuable comments.

References

Agrawal, R., & Srikant, R. (1994). Fast algorithms for mining association rules. In *Proceedings of the 20th International Conference on Very Large Data Bases* (pp. 487–499).

Apte, C., Liu, B., Pednault, E. P. D., & Smyth, P. (2002). Business applications of data mining. *Communications of the ACM, 45:8*, 49–53.

Billsus, D., & Pazzani, M. J. (1998). Learning collaborative information filters. In *Proceedings of the Fifteenth International Conference on Machine Learning* (pp. 46–54).

Breese, J. S., Heckerman, D., & Kadie, C. (1998). Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence* (pp. 43–52).

Brijs, T., Goethals, B., Swinnen, G., Vanhoof, K., & Wets, G. (2000). A data mining framework for optimal product selection in retail supermarket data: The generalized PROFSET model. In *Proceedings of the 6th ACM International Conference on Knowledge Discovery and Data Mining* (pp. 300–304).

Cadez, I. V., Smyth, P., & Mannila, H. (2001). Probabilistic modeling of transaction data with applications to profiling, visualization, and prediction. In *Proceedings of the 7th ACM International Conference on Knowledge Discovery and Data Mining* (pp. 37–46).

Dempster, A., Laird, N., & Rubin, D. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, B39*, 1–37.

Goldberg, D., Nichols, D. Oki, B. M., & Terry, D. (1992). Using collaborative filtering to weave an information tapestry. *Communications of the ACM, 35:12*, 61–70.

Herlocker, J. L., Konstan, J. A., Borchers, A., & Riedl, J. (1999). An algorithmic framework for performing collaborative filtering. In *Proceedings of the Conference on Research and Development in Information Retrieval* (pp. 230–237).

Hofmann, T. (1999). Probabilistic latent semantic analysis. In *Proceedings of the 15th Conference on Uncertainty in Artificial Intelligence* (pp. 289–296).

Lawrence, R. D., Almasi, G. S., Kotlyar, V., Viveros, M. S., & Duri, S. (2001). Personalization of supermarket product recommendations. *Data Mining and Knowledge Discovery, 5*, 11–32.

Ling, C., & Li, C. (1998). Data mining for direct marketing: Problems and solutions. In *Proceedings of the 4th International Conference on Knowledge Discovery and Data Mining* (pp. 73–79).

Pennock, D. M., Horvitz, E., & Giles, C. L. (2000). Collaborative filtering by personality diagnosis: A Hybrid memory- and model-based approach. In *Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence* (pp. 473–480).

Popescul, A., Ungar, L., Pennock, D., & Lawrence, S. (2001). Probabilistic models for unified collaborative and content-based recommendation in sparse-data environments. In *Proceedings of the Seventeenth Conference on Uncertainty in Artificial Intelligence* (pp. 437–444).

Resnick, P., Iacovou, N., Suchak, M., Bergstorm, P., & Riedl, J. (1994). GroupLens: An open architecture for collaborative filtering of netnews. In *Proceedings of ACM Conference on Computer Supported Cooperative Work* (pp. 175–186).

Shardanand, U., & Maes, P. (1995). Social information filtering: Algorithms for automating "Word of Mouth". In *Proceedings of ACM Conference on Human Factors in Computing Systems* (pp. 210–217).

Wedel, M., & Kamakura, W. A. (1999). *Market segmentation: Conceptual and methodological foundations*. Kluwer Academic Publishers.

Received April 7, 2003
Accepted April 8, 2004
Final manuscript April 11, 2004