A Recommender for Targeted Advertisement of Unsought Products in E-Commerce

Koung-Lung Lin[1,2]   Jane Yung-jen Hsu[2]   Han-Shen Huang[1]   Chun-Nan Hsu[1]
[1]Institute of Information Science, Academia Sinica Taipei, Taiwan 105
[2]Department of Computer Science and Information Engineering
National Taiwan University, Taipei, Taiwan 106
lkl@iis.sinica.edu.tw yjhsu@csie.ntu.edu.tw hanshen@iis.sinica.edu.tw chunnan@iis.sinica.edu.tw

Abstract

Recommender systems are a powerful tool for promoting sales in electronic commerce. An effective shopping recommender system can help boost the retailer's sales by reminding customers to purchase additional products originally not on their shopping lists. Existing recommender systems are designed to identify the top selling items, also called hot sellers*, based on the store's sales data and customer purchase behaviors. It turns out that timely reminders for* unsought products*, which are* cold sellers *that the consumer either does not know about or does not normally think of buying, present great opportunities for significant sales growth. In this paper, we propose the framework and process of a recommender system that identifies* potential customers *of unsought products using boosting-SVM. The empirical results show that the proposed approach provides a promising solution to targeted advertisement for unsought products in an E-Commerce environment.*

1. Introduction

With the rapid development of e-commerce, it has become critical for business proprietors and managers to improve customer relationship and to boost revenue by deploying advanced information technology. Recommender systems represent an attractive application of information technology to help guide the customers in deciding what to buy. Good recommendations not only increase the possibility of cross-selling or up-selling products, but also help target the right customers with the right products or services.

The basic idea of a recommender system is to exploit the association among users and product items in order to predict the items, such as books [12], movies [13], or news [14] that a user may like. In 1994, GroupLens [14] proposed the first automated Netnews recommender system based on *collaborative filtering* to help users find articles from a huge stream of available articles. In recent years, various applications of recommender systems in E-Commerce have been proposed [15, 16, 19, 6]. One famous example is Amazon.com [10] that utilizes recommenders as a targeted marketing tool in email campaigns, such as "Your Recommendations". Recommendations are used extensively to personalize the Amazon Web site for individual customer's interests, such as "Customers who bought this book also bought".

For many businesses, the *sales distribution of items sold* follows the *Pareto's Law*, such that eighty percent of sales come from the top twenty percent grossing items. Figure 1
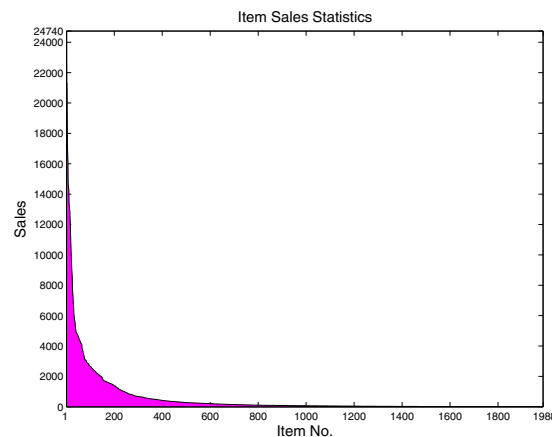

Figure 1. Sales distribution of items sold at Ta-Feng Supermarket (11/2000 to 02/2001).

shows the item sales statistics, sorted by their sales ranks, compiled from the actual transaction data from Ta-Feng Supermarket over a four-month period. The top ranking items include products that are commonly purchased by every household, such as eggs, beverages and so on. The distrib-

ution vividly illustrates the *80/20 rule* in a *power law* curve that occurs naturally in many fields of study.

Current recommender systems are designed to identify the top selling items based on the store's sales data and customer purchase behaviors. Given that sales are skewed and concentrated on a very small portion of items, a trivial recommender that simply recommends items from the front of the curve to *any* customer can predict her shopping preferences with high accuracy. Such recommendations create little added values since the items are likely on the customer's shopping list regardless of the recommendations.

It turns out to be quite difficult to improve from the trivial recommender unless a recommender can identify potential customers for items in the *tail* end of the curve. As a result, this research aims to increase the revenue of a business by boosting sales of the items beyond the top twenty percent, especially the sales of *unsought products*. An unsought product can be a consumer product of which the customer is unaware but potentially willing to buy, e.g. a new gadget or a new flavor of fruit snack. It can be something the customer does not normally think of buying, such as encyclopedias or life insurance. Unsought products tend to be cold sellers that can benefit greatly from targeted advertisement to specific buyers. Timely reminders for unsought products to targeted customers present great opportunities for significant sales growth.

Most recommender systems are *customer-triggered* in that they recommends a list of items for each customer [3]. In other words, they compare a customer's preference for different items instead of comparing the preference for a given item among different customers. In contrast, an *item-triggered* recommender system returns a list of potential customers for each cold seller [5]. Business managers may use it to create the prediction models for identifying the potential customers of a given unsought product automatically. The prediction models enable the implementation of online targeted advertisement to the mostly likely buyers.

This paper proposes the framework for an item-triggered recommender system and the associated learning process. Section 2 describes the problem of unsought product recommendation. Section 3 presents the system framework and process. To learn the model for predicting potential buyers, customers who have bought a given item serve as positive examples, while customers who did not purchase the item serve as negative examples. By focusing on recommending unsought products, negative examples significantly outnumber the positive, and it is formulated as a *rare-class classification* problem. The core of our recommender system is the Boosting-SVM algorithm, which combines boosting and SVM classifiers [5]. Section 4 shows the proposed boosting SVM algorithm and discusses the experimental results, followed by the conclusion in Section 5.

2. Item-Triggered Recommendation

In 2001, we had an opportunity to collaborate with Ta-Feng, a local supermarket in Taiwan, in developing a personalized shopping recommender system. Based on the available data and technology survey, we defined the specification of the recommender to create a ranked list of products for each customer given his/her shopping history. The goal is to use such ranked lists to support decision making of various marketing campaigns in addition to personalized recommendations.

The data set from Ta-Feng contains the transactions collected within a time span of four months, from November, 2000 to February, 2001. There are a total of 119,578 transactions involving 24,069 products and 32,266 customers in the data set[1]. Each transaction record consists of five main attributes: the transaction date, customer ID, product sub-subclass ID, product bar code, and the purchase amount. Ta-Feng adopts a common commodity classification standard consisting of a three-level product taxonomy as shown in Figure 2. Products are classified into 26 product classes,
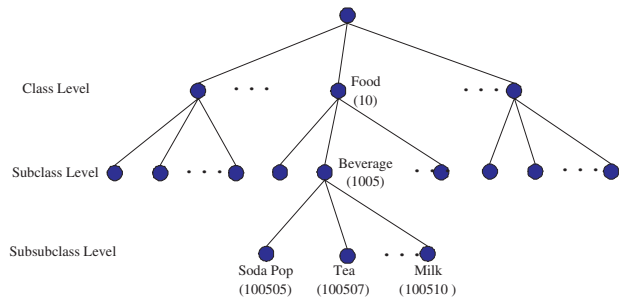

Figure 2. Product taxonomy.

201 subclasses and 2,012 sub-subclasses. Each node in the taxonomy contains a set of products.

Despite the presence of demographic data in the data set, we decided to focus on the purchase history only, because some customers provided fake personal data for privacy reasons. Transaction records with the same customer ID on any specific purchase date are considered as a single transaction. Without loss of generality, the experiments and analyses in this research considered 1,986 product sub-subclass in the product taxonomy.

In Section 1, Figure 1 showed the *skewed* sales distribution of the Ta-Feng data set. The skewness of the data makes it trivial to accurately recommend a hot seller but difficult to identify potential customers for the cold sellers. An effective recommender should promote the sales of unsought items at the tail of the curve.

2
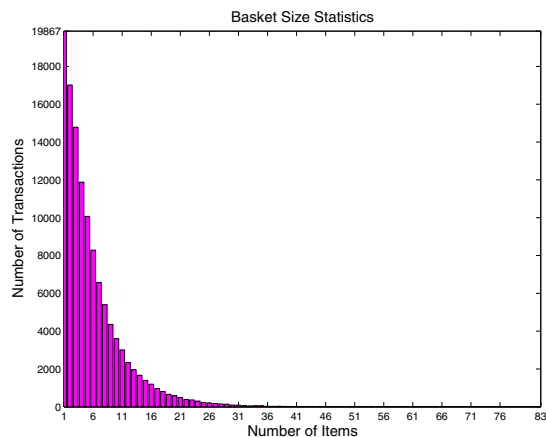
Figure 3. Sparcity of the Ta-Feng data set: distribution of transactions over basket size.

Figure 4. The diversity of recommendations between HyPAM and DEFAULT.

The Ta-Feng data set is also shown to be very *sparse*. Figure 3 plots the distribution of transactions over *basket size*, i.e., the number of different items (or product sub-subclass) in a single transaction. The number of transactions declines exponentially with increased basket size. The long right tail indicates that most transactions involve a very small number of items, so the amount of useful information available to the recommender is quite limited.

A probabilistic graphical model has been shown to be effective in handling skewed and sparse data [3]. By casting the collaborative filtering algorithm in a probabilistic framework, we derived a novel model HyPAM (Hybrid Poisson Aspect Modeling) for personalized shopping recommendation. Experimental results showed that HyPAM outperforms GroupLens [14] and the IBM method [8] by generating much more accurate predictions of what items are actually purchased by customers in the unseen test data. HyPAM also outperforms the DEFAULT method – the trivial recommender that simply recommends hot sellers over cold sellers to any customer.

However, we were surprised to find *no* obvious difference between the lists of recommended items suggested by HyPAM and DEFAULT. For ten randomly selected customers, Figure 4 compares the recommendation lists suggested by HyPAM with DEFAULT. The X-axis represents the items in ascending order of sales, and the Y-axis represents the ranking of recommendation. Each item on the recommendation list is represented by a dot, with red dots for DEFAULT and green dots for HyPAM. The straightforward recommendations by DEFAULT form a diagonal line. For the most part, the recommendations by HyPAM positively correlate with those from DEFAULT, and the similarity is especially strong for the hot sellers. Even thought HyPAM learns to generate personalized recommendations, few cold sellers can make it to the top of the list.
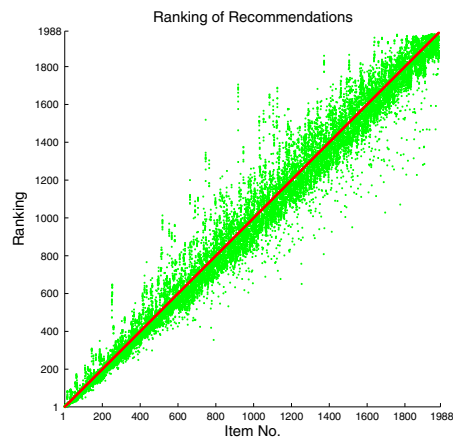
It turns out that the skewness of the data combined with the performance evaluation based on prediction accuracy imply that a perfect recommender system must recommend cold sellers less often. An accurate predictor of customer shopping preferences may improve the overall shopping experience and indirectly increase sales. However, the original problem formulation as "customer-triggered" recommendation cannot lead to the expected sales growth for the *unsought products*. It is therefore desirable for the recommender to generate a ranked list of potential customers for a given product. The "item-triggered" recommendation is proposed to promote unsought products to potential customers, which can contribute to increased sales directly and justify the investment of deploying recommender systems by the supermarket.

3. System Framework and Process

In this section, we present the design of the proposed item-triggered recommender system for targeted advertisement of unsought products in E-Commerce. Figure 5 illustrates the learning framework and process. First of all, raw data are collected by the system automatically from online *Point of Sales* (POS) system in e-commerce environments. As the famous computing axiom – "Garbage in, Garbage out" points out, erroneous data will lead to many potential problems in data mining and decision-making [7].h The *data cleaner* performs an important step in the system that removes erroneous and/or inconsistent data in order to ensure the data quality of the customer profiles. Each profile consists of the actual purchase history of the customer compiled from the transaction records in the data set. For example, the data cleaner should remove a transaction record

3
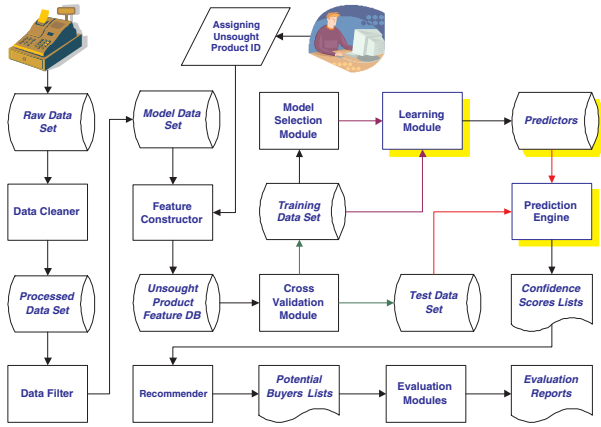
Figure 5. Item-triggered recommender system: learning framework and process.

if the corresponding products were subsequently returned. The result of data cleaning is the *processed data set*.

The next step involves sifting through the transaction records to create the *model data set*, which will be used in building the prediction model for the item-triggered recommender. For example, the transaction records associated with non-personal VIP cards should be excluded. The supermarket clerks sometimes use a temporary VIP card to grant discount prices to new customers who haven't applied for membership or existing members who forgot to bring their cards. While the POS system has no problem recording all such transactions in the temporary account, they do not represent a single personal member profile. As a result, those transactions should not be filtered out.

The unsought product recommendation is formulated as a rare-class classification problem. For any product m_j, we can learn a classifier in the feature space defined in terms of the transaction history of all customers. Suppose that there are I customers in the feature space, and each customer is represented as (b_{ij}, u_{ij}), for $i = 1 \ldots I$. The label b_{ij} indicates whether the i-th customer bought the merchandise m_j, with $b_{ij} = 1$ for positive, and $b_{ij} = 0$ otherwise. The feature vector u_{ij} of the i-th customer for item m_j can be defined as $(n_{i1}, \ldots, n_{i(j-1)}, n_{i(j+1)}, \ldots, n_{iK})$, where K is the total number of items, and n_{ik} may be either zero or one, denoting whether the i-th customer bought the k-th item; or a non-negative integer, denoting the average volume of the k-th item purchased by the i-th customer within the given period of time. Given any unsought product m_j, the corresponding feature vectors u_{ij}'s are generated by the *feature constructor* and stored in the *unsought product feature database*.

There are two steps in learning a classifier for a given unsought product. First, the feature space is randomly partitioned into a *training data set* and a *test data set*. The former is used to generate the classifier. Second, the latter is used to validate the effectiveness of the learned classifier. For an unbiased validation we suggest *10-fold cross-validation*, in which the item feature space is randomly split into 10 disjoint subsets, each of size $I/10$. The results are averaged over 10 trials, each time leaving a single segment out for independent testing, and training the recommender on the remaining $9/10$ of the data set.

To achieve the best balance between data-fitting and model complexity, *model selection* has become a critical step in machine learning. The models that can be learned by any specific machine learning algorithm often depend on a set of parameters, which need to be tuned to optimize the learned models with respect to certain criteria. This research formulates the item-triggered recommendation problem as a *two-class classification* problem where the data is sparse. In particular, the SVM classifier is chosen due to its superior ability in handling sparse data. In order to learn the classifier with the maximum predictive power, we perform model selection to choose the optimal penalty parameter C and kernel parameters for the training data before the formal learning process begins [4]. We use the LIBSVM package [1], a variation of SVM that can output the probability of its classification results [21], in our experiments.

Since we aim at identifying potential customers for unsought products, such as new packaging of cosmetic products, baby care products for expectant mothers and new ice cream flavors, the training data for the SVM will be very *imbalanced*. Namely, negative examples greatly outnumber positive examples, because only a very small portion of customers have purchased the items. This problem is known as the *rare class* problem, and SVM alone cannot handle imbalanced training data very well. We propose using a boosting algorithm to train an ensemble of SVMs to handle such imbalanced data. The intuitive idea is to train several SVM classifiers to enclose the positive examples separately from the negative ones. The process starts by training an SVM classifier with a less imbalanced *subset* of data, and then classify the entire training data set with the SVM to identify the incorrectly classified examples. The first classifier can be reinforced by training another SVM classifier for those incorrectly classified data. The process is repeated until we obtain an SVM ensemble in which each classifier tries to enhance the performance of its previous one. In this way, the combination of the SVM ensemble can provide a finer classification boundary to separate positive and negative data.

After the learning process, we obtain several base classifiers and their relative weights from the classifier learning algorithm. Each base classifier is treated as a *predictor* and executed by the *prediction engine* to predict the confidence score that shows whether a customer will buy a given prod-

4
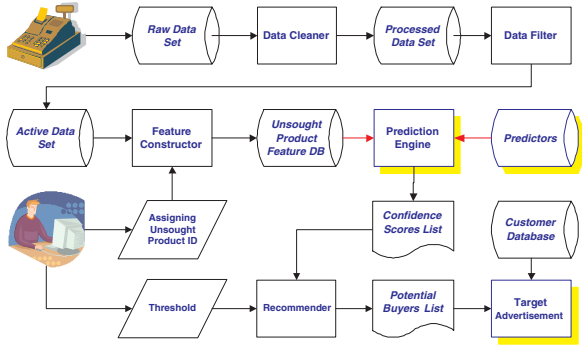
Figure 6. Online targeted advertisement: application framework and process.

uct based on his/her profile. The ensemble confidence score is generated by the weighted average of all base classifiers. Then, the *recommender* will output the *potential customer list*, which is a list of customers ordered in descending rank by the ensemble confidence score for a given item.

Herlocker et al. [2] suggested using the *Receiver Operating Characteristic* (ROC) curve as a measure to evaluate recommender system. An ROC curve is a graphical representation of the trade off between the true positive and false positive rate for every possible cutoff. The plot shows the false positive rate (false alarm) on the X-axis and the true positive rate (recall) on the Y-axis. ROC analysis can be done by tuning the ratio between the true positive rate and the false positive rate for a recommender system under various threshold. This type of analysis can also be used for cost/benefit analysis of marketing strategy. We use the *area under the ROC curve* (AUC), between 0 and 1, to measure the performance of the item-triggered recommender. An AUC of 1 indicates perfect discrimination between potential customers and non-customers, while an AUC of 0.5 or less indicates the recommender system has no power to discriminate. Note that other metrics which focus on evaluating individual recommendation, such as accuracy or absolute deviation, are not appropriate here because the score will be high if all the customers are predicted as not going to buy the unsought product.

After training the classification models (predictors) of the unsought products, we can apply them to actual online application. Figure 6 illustrates the application framework and process for online targeted advertisement in E-Commerce. Its main purpose is to automatically deliver the advertisement to the potential customers of unsought products. First, the business manager selects an unsought product, and then the transaction histories of all customers in the active data set were translated into the accessible format for the corresponding base classifiers (predictors). All base classifiers of the specific unsought product are executed by

the *prediction engine* and output a list of confidence scores of all customers in active data set according to the weighted average of their classification results. To achieve optimal performance of targeted advertisement within the limited budget, the business manager assigns a threshold based on the prior ROC analysis to divide the list of customers into two groups. A proportion of the customers from the top of the list will be viewed as potential customers for the unsought product, and the recommended list will be output by the recommender for targeted advertisement.

4. Boosting SVM

Item-triggered recommendation for unsought products is formulated as a rare-class classification problem since there are many more positive examples (customers who have bought the item) than negative examples (customers who haven't bought the item). Our approach, Boosting-SVM, learns an SVM ensemble for each item, and classifies customers as "buy" or "not buy" with probabilities showing the confidence of classification. Finally, a customer list ordered by their probability to buy the given item is returned.

This section presents the Boosting-SVM algorithm, followed by discussions on the experiments and the results.

4.1. Algorithm

Let us formally present the Boosting-SVM algorithm, which is detailed in Algorithm 1 below.

Algorithm 1 Boosting-SVM

1: Initialize $D = \{x_1, y_1, \ldots, x_N, y_N\}$, T, M, W_0 and $t = 1$;
2: Let Z_t be a normalization constant
3:    $W_t(i) = \frac{W_0}{Z_t}$ if $y_i = +1$, (positive examples)
4:    $W_t(i) = \frac{1}{Z_t}$ if $y_i = -1$; (negative examples)
5: **while** $t \leq T$ **do**
6:    Train $h_t(x)$ using D sampled according to W_t;
7:    Calculate α_t for $h_t(x)$;
8:    Calculate *err* for $h_t(x)$; (error rate)
9:    $W_{t+1}(i) = \frac{W_t(i)}{Z_{t+1}} \exp(-(1 - err))$, if $h_t(x_i) = y_i$; (correctly classified cases)
10:   $W_{t+1}(i) = \frac{W_t(i)}{Z_{t+1}} \exp(1 - err)$, if $h_t(x_i) \neq y_i$; (incorrectly classified cases)
11:   $t = t + 1$;
12: **end while**
13: Return $\{h_1, \alpha_1, \ldots, h_T, \alpha_T\}$;

Let $D = \{x_1, y_1, x_2, y_2, \ldots, x_N, y_N\}$ denote the training examples, where x_i is the feature vector of customer i with respect to the given item, and y_i indicates whether the customer bought the given item. $W_t(i)$ is the probability

that (x_i, y_i) pair will be selected to train the t-th classifier $h_t(x)$. M is the number of training examples that will be selected for $h_t(x)$ and T is the number of classifiers that will be trained in total.

We have tried a standard SVM as $h_t(x)$, since SVM can handle sparse and high dimensional data better [20]. However, we found that the resulting recall is quite low due to the skewed transaction data. In many cases, the SVM simply predicts that nobody will buy the given product. This yields a very low error rate but clearly is not desirable. Instead, we chose LIBSVM 2.6 [1] because it includes a reliable SVM variant that can output probabilities for all predictive results [21].

To predict whether a customer will or will not buy the given item, the trained classifiers will be combined linearly with $\alpha_1, \ldots, \alpha_T$, which are the weights of $h_1(x), \ldots, h_T(x)$, respectively. That is, the likelihood that customer i will buy the item m_j is estimated as:

$$P(y_i = +1|x_i, m_j) = \sum_{t=1}^{T} \alpha_t P(h_t(x_i) = +1). \quad (1)$$

4.2. Experiments

There are a total of $17,290$ valid customers and $F = 1,986$ product sub-subclasses from the Ta-Feng data set. Using 10-fold cross-validation, each training set contains $15,561$ customers, with a corresponding test set of $1,729$ customers. We selected 381 out of 1,986 items as cold sellers, which have been sold to at least 100 customers but less than 500 customers. Note that the corresponding percentages of buyers range from $0.64\%(100/15,561)$ to $3.2\%(499/15,561)$ for the cold sellers. Eight items are identified from the cold sellers to carry out the unsought products experiments. They are 712102 (electronic heater), 510104 (fondue pot), 120302 (ice cream), 120305 (ice candy), 560311 (baby teether), 560334 (baby feeding bottle), 300604 (lipstick) and 300701 (toning lotion). We did not consider those items which were purchased less than 100 times because they might not worth being recommended to customers.

For each item, we applied the following three different algorithms to produce the ranked lists of potential customers. Table 1 reports the experimental results of the algorithms for each of the items. The performance is measured by the average AUC scores based on 10-fold cross-validation.

DEFAULT : This is the trivial algorithm that outputs a customer list sorted by the shopping frequency of each customer. Basically, if a customer comes more often, the prior probability that the customer will purchase

Table 1. Unsought products: average AUC scores based on 10-fold cross-validation.

Item	DEFAULT	SVM15561	P+eSVM+ROC
712102	**0.643**	0.553	0.616
510104	0.583	0.495	**0.657**
120302	0.656	0.571	**0.715**
120305	0.626	0.574	**0.743**
560311	0.589	0.742	**0.874**
560334	0.592	0.748	**0.867**
300604	0.693	0.727	**0.829**
300701	0.688	0.704	**0.825**

an unsought product is higher. Note that this algorithm generates the same customer list for any item. The performance of DEFAULT is treated as the baseline of a qualified recommendation algorithm.

SVM15561 : SVM is trained using all the training data. This algorithm represents the strategy that we do not consider the rare class problem.

P+eSVM+ROC : This one is the instantiation of the Boosting-SVM algorithm as defined in Algorithm 1 with higher initial sampling probability for all positive examples (i.e., $W_0 = 100$ in Algorithm 1) in training data set. In other words, we balance the positive/negative ratio in sampling data set, where the number of sampling examples is twice as many as the positve ones in training data set. "eSVM" is the abbreviation of "ensemble SVM". Therefore, all the base classifiers will be combined linearly with their weights. The "ROC" represent that the weight α_t is calculated using the AUC scores of the base classifier against the training data set.

4.3. Results and Discussion

All items selected for the experiments were cold sellers. We divided them into two groups: non-unsought products and unsought products. People prefer to buy the heater and to have fondue in the winter than in other season because of the cold weather. Hence, we selected item 712102 (electronic heater) and item 510104 (fondue pot) as non-unsought products. Generally, people will eat more ice cream in the summer months than in the winter months. Hence, the consumers' needs for ice cream decrease in the winter. For stimulating the sales figures in winters, the ice cream makers usually push new flavors at that time. As the data set was collected in the winter, we selected item

6

120302 (ice cream) and item 120305 (ice candy) as un-sought products. Besides, the baby care products can be viewed as unsought products for some customers, such as expectant mothers and fathers. They do not have urgent needs to buy the baby goods because the baby is not yet born, that is why we selected item 560311 (baby teether) and item 560334 (baby feeding bottle) as unsought products. The cosmetics departments usually promote products with new packaging to get the prospect's attention. We selected item 300604 (lipstick) and item 300701 (toning lotion) as new unsought products because in these two sub-subclasses there are usually products with new styles or packing that potential customers do not know about this yet. Table 1 reports the experiment results of the algorithms for each item. The results show that our approach (P+eSVM+ROC) outperform DEFAULT except the item 712102; note that there is no very obvious performance difference in item 510104. The result means that our approach is superior to DEFAULT and SVM15561 to identify the potential customers of unsought products. For the non-unsought products, however, its potential customers can be easily predicted according to customers' shopping frequency.

The ROC curve allows us to see how many buyers will be identified by different algorithms from their corresponding recall scores. Figure 7 shows the ROC curves of DEFAULT and P+eSVM+ROC for the item 560311. Each curve consists of 100 data points, representing 100 cutoff points to divide the customers ranked by the predicted probability that they will buy the product. Suppose that we are recommending an item to the customers at the top 10% of the ranked list, since the recall values of DEFAULT and P+eSVM+ROC are 21% and 65% at that points, respectively, we can expect that P+eSVM+ROC will help us to identify three times as many buyers as those by DEFAULT.

With the ranked list of potential customers, marketing staffs can design a campaign strategy targeting a certain percentage of customers at the top of the list. The optimal percentage can be determined by maximizing a utility function that takes many factors into account, such as resource available for the campaign, supply and stock status of the item, etc..

In real application, we can use the ratio of one item's total revenue to its advertisement cost as the evaluation metric. For example, given one cold seller the business proprietors and managers can compare the performances of two different item-triggered recommender systems by the ratios of individual revenue earn by the two different recommenders to the same advertisement budget. If the cost to send a message is one dollar and they prepare ten thousand advertisement budget for each recommender then each recommender can send messages to their top ten thousand potential customers. Higher revenue represents advertise-
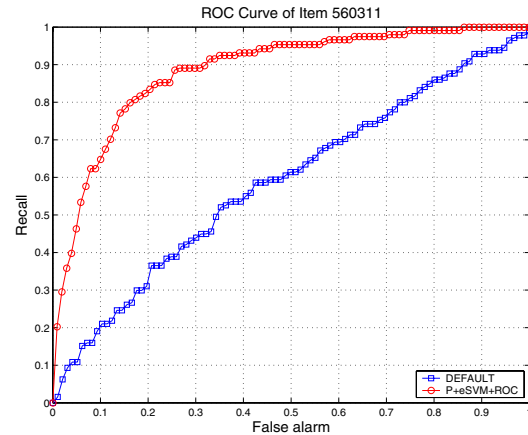
Figure 7. Comparison of the ROC curves of DEFAULT and P+eSVM+ROC.

ment result which has higher recall and lower false alarm. On the contrary, lower revenue represents advertisement result which has lower recall and higher false alarm. Accordingly, the recommender with the higher ratio of revenue to advertisement cost is better than another.

4.4. Related Work

This paper presented our experiments of combining SVM and boosting to train the proposed item-triggered recommender system.

4.4.1 Support Vector Machines

Originally proposed by Vapnik [20], SVM learns a hyperplane to separate positive and negative examples. The hyperplane is oriented from the maximal margin between positive and negative classes so that the risk of misclassification is minimized.

One of the approaches to solving the rare-class classification problem for SVM is sample balancing, hierarchical SVM framework [22]. In this work, negative examples are divided uniformly and combined with all the positive ones to train a set of SVM classifiers. A top level SVM then takes their classification results as the input to produce the final classification result. Unlike their approach, we apply boosting and linear combination to combine the ensemble of SVM classifiers.

4.4.2 Boosting

Boosting [17] uses re-sampling techniques to learn a set of classifiers, and linearly combines them to predict the class of input data. The probability that an example is chosen to

train a classifier is determined by whether its true class can be correctly predicted or not by the classifier learned in the previous iteration.

Many boosting methods have been proposed for general or specific purposes. One of the most well-known algorithm is AdaBoost [18], which minimizes the error rate of the entire training data set without imposing any restriction to the training data. When the training data is imbalanced, AdaUBoost, a variant of AdaBoost, suggests that minority examples be initialized with higher weights and lower updating rate when they can be correctly classified [18, 9]. DataBoost-IM, another method to deal with the rare-class problem, is to synthesize more positive examples using the previously learned classifiers [11]. Currently, we adopt the method similar to AdaUBoost to initialize weights of training examples.

5. Conclusion

This paper has presented our approach to targeted advertisement for unsought products in E-Commerce. In this research, item-triggered recommendation is formulated as a rare-class classification problem, which is usually handled by an ensemble of SVM classifiers. With Boosting-SVM, we can obtain potential customer lists sorted by their probability to purchase the given items to support targeted advertisement for the unsought products. The experimental results show that our approach is superior to the shopping frequency and single SVM methods, and is a promising solution to identifying potential customers for cold sellers. While the current experiments are based on the transaction data collected by a supermarket rather than an online store, such recommendations in online retail scenarios should prove to be even more effective. To further validate the proposed solution, we plan to seek cooperation with some E-Commerce business in the near future.

References

[1] C.-C. Chang and C.-J. Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at http://www.csie.ntu.edu.tw/ cjlin/libsvm.

[2] J. L. Herlocker, J. A. Konstan, L. G. Terveen, and J. T. Riedl. Evaluating collaborative filtering recommender systems. *ACM Trans. Inf. Syst.*, 22(1):5–53, 2004.

[3] C.-N. Hsu, H.-H. Chung, and H.-S. Huang. Mining skewed and sparse transaction data for personalized shopping recommendation. *Machine Learning*, 57(1-2):35–59, 2004.

[4] C.-W. Hsu, C.-C. Chang, and C.-J. Lin. A practical guide to support vector classification. Technical report, Department of Computer Science and Information Engineering, National Taiwan University, July 2003.

[5] H.-S. Huang, K.-L. Lin, J. Y. jen Hsu, and C.-N. Hsu. Item-triggered recommendation for identifying potential cus-tomers of cold sellers in supermarkets. In *Beyond Personalization 2005, A Workshop on the Next Stage of Recommender Systems Research, In conjunction with the IUI 2005*.

[6] Z. Huang, W. Chung, and H. Chen. A graph model for e-commerce recommender systems. *J. Am. Soc. Inf. Sci. Technol.*, 55(3):259–274, 2004.

[7] R. Kohavi, L. Mason, R. Parekh, and Z. Zheng. Lessons and challenges from mining retail e-commerce data. *Machine Learning*, 57(1-2):83–113, 2004.

[8] R. D. Lawrence, G. S. Almasi, V. Kotlyar, M. S. Viveros, and S. S. Duri. Personalization of supermarket product recommendations. *Data Mining and Knowledge Discovery*, 5(1-2):11–32, 2001.

[9] J. Leskovec and J. Shawe-Taylor. Linear programming boosting for uneven datasets. In *Proceedings of ICML-2003*, pages 351–358, 2003.

[10] G. Linden, B. Smith, and J. York. Amazon.com recommendations: item-to-item collaborative filtering. *Internet Computing, IEEE*, 7(1):76–80, 2003.

[11] M. A. Maloof. Learning when data sets are imbalanced and when costs are unequal and unknown. In *ICML-2003 Workshop on Learning from Imbalanced Data Sets II*, 2003.

[12] R. J. Mooney and L. Roy. Content-based book recommending using learning for text categorization. In *Proceedings of DL-00, 5th ACM Conference on Digital Libraries*, pages 195–204, San Antonio, US, 2000. ACM Press.

[13] R. Mukherjee, G. Jonsdottir, S. Sen, and P. Sarathi. Movies2go: an online voting based movie recommender system. In *Proceedings of the 5th international conference on Autonomous agents*, pages 114–115. ACM Press, 2001.

[14] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl. GroupLens: an open architecture for collaborative filtering of netnews. In *CSCW '94: Proceedings of the 1994 ACM conference on Computer supported cooperative work*, pages 175–186. ACM Press, 1994.

[15] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Analysis of recommendation algorithms for e-commerce. In *Proceedings of the 2nd ACM conference on Electronic commerce*, pages 158–167. ACM Press, 2000.

[16] J. B. Schafer, J. A. Konstan, and J. Riedl. E-commerce recommendation applications. *Data Mining and Knowledge Discovery*, 5(1-2):115–153, 2001.

[17] R. E. Schapire. The strength of weak learnability. *Machine Learning*, 5(2):197–227, June 1990.

[18] R. E. Schapire and Y. Singer. Improved boosting algorithms using confidence-rated predictions. *Machine Learning*, 37(3):297–336, 1999.

[19] T. Sun and A. Trudel. An implemented e-commerce shopping system which makes personal recommendations. In *Proceedings of the Internet and Multimedia Systems and Applications*, pages 58–62. IASTED/ACTA Press, Aug 2002.

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

[21] T.-F. Wu, C.-J. Lin, and R. C. Weng. Probability estimates for multi-class classification by pairwise coupling. *J. Mach. Learn. Res.*, 5:975–1005, 2004.

[22] R. Yan, Y. Liu, R. Jin, and A. Hauptmann. On predicting rare classes with SVM ensembles in scene classification. In *IEEE ICASSP2003*, volume 3, pages 21–24, 2003.

IEEE
COMPUTER
SOCIETY