Item-Triggered Recommendation for Identifying Potential Customers of Cold Sellers in Supermarkets

Han-Shen Huang[1] **Koung-Lung Lin**[1,2] **Jane Yung-jen Hsu**[2] **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

ABSTRACT

Recommendation has achieved successful results in many applications. However, for supermarkets, since the transaction data is extremely skewed in the sense that a large portion of sales is concentrated in a small number of hot seller items, collaborative filtering recommenders usually recommend hot sellers while rarely recommend cold sellers. But recommenders are supposed to provide better campaigns for cold sellers to increase sales. In this paper, we propose an alternative "item-triggered" recommendation to identify potential customers of cold sellers. In item-triggered recommendation, the recommender system will return a ranked list of customers who are willing to buy a given item. This problem can be formulated as a problem of classifier learning, but again, due to the skewed distribution of the transaction data, we need to solve the rare class problem, where the number of negative examples is much larger than the positive ones. We present a boosting algorithm to train an ensemble of SVM classifiers to solve the rare class problem and compare the algorithm with its variants. We apply our algorithm to a real-world supermarket database and use the area under the ROC curve (AUC) metric to evaluate the quality of the output ranked lists. Experimental results show that our Boosting-SVM algorithm can improve from a baseline approach by about twenty-five percent for cold sellers that as low as 0.7% of customers have ever purchased.

Keywords

Recommendation, Cold Seller, SVM, Boosting, ROC

INTRODUCTION

Recommender systems have achieved successful results in many applications. We can see them work in our daily lives. A good recommender system can help increasing sales at online merchants as well as brick-and-mortar retailer stores.

In this paper, we address the problem of applying collaborative filtering recommender systems in brick-and-mortar supermarkets. This problem is challenging for current recommender systems in that the transaction data is extremely skewed in the sense that a large portion of sales is concentrated in a small number of items. Figure 1 shows the sales distribution of items sold in a supermarket. In the figure, items are sorted and re-numbered by the order of their sales figures in a given period of time. The sales figure here is the amount of the items purchased by the customers, not the revenue or other quantities. The plot shows that the sales distribution is skewed and concentrated on a very small portion of product items. This is typical to retailer stores and is an example of "*the 80-20 rule*" known in business management. The curve is also known as the *power law* curve that occurs naturally in many fields of study. A trivial recommender that always recommends hot seller items to any customer can achieve pretty accurate prediction of the customers' shopping preference. In fact, it is difficult to improve from the trivial recommender because it is difficult for a recommender to identify potential customers for the items in the tail of the curve. From the point of view of the supermarket which pays to deploy a recommender system, it is much more useful for a recommender system to recommend those *cold sellers* accurately than recommend hot sellers accurately.

In this paper, we propose *item-triggered* recommendation, an alternative view of recommendation. Previous recommender systems are *customer-triggered* in that they return a list of items as the recommendation for each customers. In contrast, item-triggered recommender system will return a list potential customers for each cold seller item. A given proportion of the customers from the top of the list will then receive the recommendation to buy that cold seller item. If this can be done accurately, the recommender system will help increasing sales of those cold sellers, which are in need of better campaigns. Previously, Sarwar et al. [10] proposed an item-based approach to recommendation. Item-based recommendation is not item-triggered because it is basically customer-triggered and still aims at returning a list of items for each customer.

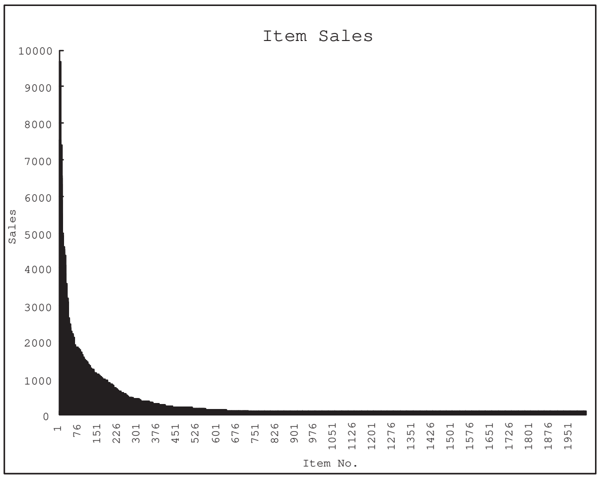

Figure 1: Sales distribution of items in Ta-Feng retailer store

The problem specification of item-triggered recommendation is, given a cold seller item, estimating the probability that a customer will buy that item given a set of features of that customer. We can apply a binary classifier that returns a confidence score of its classification result (i.e., will buy or will not buy) to solve the problem. Among many classifiers, we chose SVM because SVM can handle sparse data better than other classifiers [13]. Transcation databases in supermarkets are very sparse in that each customer only purchased a very small subset of the entire set of available items. Therefore, if we use transaction data as the features, we will need a classifier that can handle sparse data. A variation of SVM in LIBSVM [1] can output the probability of its classification results [14]. We will use that version of SVM from LIBSVM to solve our problem. Since we aim at identifying potential customers for cold sellers, the training data for the SVM will be very *imbalanced* — by definition of the cold seller, only a very small portion of customers have purchased the item. As a result, the ratio of positive data (customers who have purchased the item) and negative data (customers who did not purchase the item) will be very small. This problem is known as the *rare class* problem in machine learning. SVM alone cannot handle imbalanced training data. We propose a boosting algorithm to train an ensemble of SVMs to handle imbalanced data. The idea is to extract a subset of the training data such that the subset is less imbalanced and has more incorrectly classified data so that each SVM in the ensemble is trained using different combination of positive and negative data. In this way, the combination of the SVM ensemble can provide a finer classification boundary to separate positive and negative data.

We will use the area under the ROC curve metric (AUC) to measure the quality of the output list of potential customers by our new recommender system. See [4] for its use in recommender systems and [2] for its use in machine learning. The AUC metric takes into account both false positives and false negatives and is suitable to measure the quality of a ranked list. Other metrics that 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 cold sellers.

In our experiments, the SVM ensembles were compared with sorting customer lists by their shopping frequency. The latter serves as a baseline for item-triggered recommendation. The results show that the SVM ensemble outperforms the baseline one by increasing the AUC of the latter by 25%.

This paper is organized as follows. First, we describe the background motivation and the problem definition of item-triggered recommendation. Next, we present the boosting-SVM algorithm and experimental results. At last, we discuss related work and conclusion.

ITEM-TRIGGERED RECOMMENDATION

In 2001, we had a chance of collaboratation to develop a personalized shopping recommender with Ta-Feng, a large retailer store in Taiwan that sells a wide range of merchandise, from food and grocery to office supplies and furniture. After surveying a variety of technologies, 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 the customer's preference, given his/her historical shopping record. This specification is to use accurate ranked lists for individual customers to support various marketing strategies in addition to personalized recommendation.

The transaction data set from Ta-Feng contains the transactions collected in a time span of four months, from November, 2000 to February, 2001[1]. 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. Ta-Feng adopts a common commodity classification standard that consists of a three-level product taxonomy. Products are classified into 201 product classes and 2012 sub-classes.

Figure 1 shows the sales distribution plot of the items sold at Ta-Feng according to the transaction data set described above. The plot shows that the sales figure is skewed and concentrated on a very small portion of product items. We plotted the sales distributions for other supermarkets and found that they all have the similar curve, which shows that this is typical for supermarkets. The skewness of the data makes it easy to accurately recommend a hot seller item but difficult to identify potential customers for the items in the tail of the curve, that is, the cold sellers.

Our previous work shows that a probabilistic graphical model can be effective in handling skewed and sparse data [5]. By casting collaborative filtering algorithms in

[1]The data set is available for download at the following URL: http://chunnan.iis.sinica.edu.tw/hypam/HyPAM.html.

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 [9] and the IBM method [7] by generating much more accurate predictions of what items a customer will actually purchase in the unseen test data. HyPAM also outperforms the "default" method — the trivial recommender that always recommends the best sellers to any customer. However, when we compared the items recommended by HyPAM and the trivial recommender, we found that the difference is not that obvious. HyPAM can tailor to a customer's need by recommending some cold sellers but most of the times hot sellers are at the top of the recommendation. In fact, if we evaluate a recommender system's performance by comparing its recommendation with the unseen data in the transaction data set, given the skewness of the data, a perfect recommender system must recommend cold sellers less often. This implies that our original problem formulation, the "customer-triggered" recommendation, and the evaluation metrics, will not lead to large sales increasing for cold sellers, but cold sellers provide a wide-open opportunity of large sales increasing.

This is why we propose an "item-triggered" recommendation approach. Rather than recommending a list of items to each customer, the item-triggered recommender outputs a customer list ordered by the probability that the customers are willing to buy a given item. An accurate predictor of customers' shopping preference may improving customers' shopping experience and indirectly increase the sales, but increasing sales of cold sellers can contribute directly and justify the investment of deploying a recommender system by the supermarkets.

BOOSTING SVM

As we described in the introduction section, we can formulate the problem of item-triggered recommendation as a classifier learning problem, but we will need to face the "rare class" problem. This section presents our preliminary solution to item-triggered recommendation. Our goal is to develop a classifier for each cold seller item to accurately classify whether a customer will or will not buy the item with a probability. Ordered by the probability, the customers constitute a ranked list in the order of the likelihood that they will buy the item.

The training data of the classifier is the data of customers that we already know whether they bought or did not buy the given item. A customer who bought the item will be treated as a positive example and negative otherwise. Clearly, the training data will be very imbalanced because for cold sellers, there will be many negative examples and very few positive examples. In our experiment, the ratio of positive and negative examples is about as low as 2%. There are really "cold" sellers in our transaction data with an extremely low ratio. If the ratio for an item is lower than 0.7%, that is equivalent to having less than 100 customers in four months, then we will not consider the item here because it is too difficult to derive anything from the transaction database for this item and the item might not worth being recommended to customers at all.

We will evaluate the trained classifier with a test data set of customers. This test data set is disjoint with the training data set. Given a threshold of probability, we can divide the output customer list by the classifier into two sets: one set contains customers with the probability to buy the item higher than the threshold, and the other set contains customers with the probability lower than the threshold. Customers in the former set is predicted positive (i.e., will purchase the item) and the latter set is predicted negative (i.e., will not buy the item). Then we can compare the positive and negative sets with the real class label in the test data set and calculate recall and precision. By adjusting the threshold, the classifier will yield different recall and precision and we can apply the area under the ROC curve (AUC) metric [2] to evaluate the quality of the predicted customer list. A classifier will have a high AUC score if those customers who actually buy the item are ranked at the top of the list. A perfect classifier will have AUC score equal to one while random guess will yield 0.5 AUC score. The baseline for our performance evaluation is shopping frequency. For any item, if a customer visits and shops at the supermarket more often, then the customer is predicted as more likely to buy the item. This baseline strategy can yield a better AUC score than random guess.

We have tried a standard SVM as the classifier, since SVM can handle sparse and high dimensional data better [13]. 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 item. 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 classification probability [14]. With the probability, SVM can return a ranked list of potential customers. Also, by adjusting the threshold, at least we will have some customers predicted positive.

Though LIBSVM can reduce the impact of imbalanced data for SVM, its performance is barely better than shopping frequency. We altered the ratio of positive and negative examples in the training data and found that training data with a higher ratio tended to help improving the performance of LIBSVM. This led to the idea that we may apply *boosting* [11], a well-known technique in machine learning, to improve the performance by controlled re-sampling. The basic idea of boosting is to sample a subset of training data to train a "weak learner," in this case, SVM. Two parameters determine how sampling will be performed: the ratio of positive and negative examples and the classification results by the classifier trained in the previous iteration. The sampling iterates a constant number of times to yield as many SVM classifiers that constitute a classifier ensemble.

We now formally present our Boosting-SVM algorithm. Let $D = \{x_1, y_1, \ldots, x_N, y_N\}$ denote the training examples, where x_i is the feature vector of customer i and y_i indicates whether the customer bought the given item. $W_t(i)$ is the probability that (x_i, y_i) will be selected to train the t-th classifier $h_t(x)$. M is the number of examples that will be selected for $h_t(x)$ and T is the number of classifiers that will be trained in total. The Boosting-SVM algorithm is defined as follows:

Algorithm 1 Boosting-SVM

```
Initialize D = {x_1, y_1, ..., x_N, y_N}, T, M, W_0 and t = 1;
Let Z_t be a normalization constant
    W_t(i) = W_0 / Z_t if y_i = +1, (positive examples)
    W_t(i) = 1 / Z_t if y_i = -1; (negative examples)
while t <= T do
    Train h_t(x) using D sampled according to W_t;
    Calculate alpha_t for h_t(x) based on D;
    Calculate err for h_t(x) based on D; (error rate)
    W_{t+1}(i) = W_t(i) / Z_{t+1} exp(-(1 - err)), if h_t(x_i) = y_i; (correctly classified cases)
    W_{t+1}(i) = W_t(i) / Z_{t+1} exp(-(1 - err)), if h_t(x_i) != y_i; (incorrectly classified cases)
    t = t + 1;
end while
Return {h_1, alpha_1, ..., h_T, alpha_T};
```

To classify a customer, the trained classifiers will be combined linearly with $\alpha_1, \ldots, \alpha_T$, the weights of $h_1(x), \ldots, h_T(x)$, respectively. That is, the probability that customer i will buy the item is estimated as:

$$P(y_i = +1) = \sum_t \alpha_t P(h_t(x_i) = +1). \tag{1}$$

There are many possible ways to calculate the weights α_t. We have tried three methods to calculate the weights and varied constant W_0 to adjust the initial sampling probability. This yields different variants of Algorithm 1 so that we can empirically determine their impact on the performance.

EXPERIMENTAL RESULTS

This section reports the experimental evaluation of our item-triggered recommendation approach. We randomly selected 15,000 customers from Ta-Feng data set to train the recommenders and a disjoint set of 1,000 customers as the test set. There are $F = 2,012$ product subclasses in the Ta-Feng transaction database. They consitute the items in our recommendation task. For a selected item, we converted the records of customer i in the transaction database into the form (x_i, y_i). The shopping records of the selected item were only used to assign y_i, and omitted when we generated x_i. Each x_i is a feature vector of length F and $x_i(j)$, the j-th feature, is the average purchases of item j within a month. These data preprocessing steps can be accomplished using some simple SQL commands to the transaction database.

For each item, we applied the following variants of SVM and the boosting algorithms to produce the ranked lists.

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 a cold seller 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.

SVM15000 : SVM is trained by all the training data. This algorithm represents the strategy that we do not consider the rare class problem.

P+SVM : SVM is trained by all the positive examples and randomly selected negative examples, where the number of negative examples is twice as many as the positive ones. In other words, P+SVM will use only 2.1% to 8.1% of the training examples, but the distribution of positive and negative examples is different from the original training data set. This algorithm represents another strategy: we balance the positive/negative ratio and train a single classifier to rank customers.

U+SVM : This algorithm is the same as P+SVM except that the sample distribution is uniform. Therefore, the training examples of U+SVM is a subset of the original training data set with the same imbalanced distribution of positive and negative examples.

U+E, U+U : These two are the instantiations of the Boosting-SVM algorithm as defined in Algorithm 1 with uniform initial sampling probability (i.e., $W_0 = 1$ in Algorithm 1) for the whole training data set. In addition, the weight of a classifier is calculated differently. U+E calculates the weights using $1/2 \ln(\text{Accuracy/Error Rate})$, the Adaboost's formula [12], and U+U uses uniform weights for all classifiers.

U+ROC : This algorithm uses the same initialization of the sample distribution as U+E and U+U, but calculates the weights using the AUC scores of the classifiers against training data.

P+E, P+U, P+ROC : These three algorithms assign a high sampling probability for positive examples. More specifically, we set $W_0 = 100$ in Algorithm 1. Their weight calculation methods are the same as U+E, U+U and U+ROC, respectively.

We divided the cold seller items into four sets according to the number of customers who have purchased them in the training data: (A) 100–149, (B) 150–199, (C) 200–299, and

(D) 300–399. Since there are a total of 15,000 customers, the corresponding percentages of buyers range from 0.7% to 2.7%. There are 120 items in the item set (A), 77 in (B), 88 in (C), and 59 in (D). Table 1 reports the experimental results of the algorithms for each of the item sets. The performance is measured by the average AUC scores. The results show that all algorithms outperform DEFAULT except U+SVM, which used less than 8.1% of the training examples. Since we fetched the positive and negative examples with the same probability, U+SVM is trained by a small and imbalanced data set with the same ratio of positive and negative examples. For SVM15000, we used the whole data set for training and obtained larger AUC scores. This shows that LIBSVM can outperform DEFAULT when the training data set is sufficiently large.

Then, we discuss the influence of imbalanced positive and negative examples. SVM15000 and P+SVM apply the same learning and predicting algorithms. The only difference is that P+SVM used only a small portion of the negative examples. The whole training data size for P+SVM is the same as U+SVM. From the experimental results, we can see that the AUC scores of P+SVM are larger than U+SVM and averagely about 0.06 more than those of SVM15000. The results show that using a small but more balanced data set is better than using a large but extremely imbalanced data set.

The experimental results also show that adopting ensembles of classifiers can enhance the performance of weak classifiers. For U+ROC, U+E, and U+U, the classifier learned in the first iteration is the same as the classifier learned by U+SVM. After several iterations, the weights of positive examples will be increased so that the subsequent data sets will become more balanced and therefore, U+ROC, U+E, and U+U can outperform U+SVM. P+ROC, P+E, and P+U perform better than U+ROC, U+E, and U+U because their initial training data set is more balanced. Consequently, the best results for all item sets are produced by P+ROC, P+E, and P+U, which improve from DEFAULT by 25% in terms of the AUC scores.

The ROC curve allows us to see how many buyers will be identified by different algorithms from their corresponding recall scores. Figure 2 shows the averaged ROC curves of DEFAULT and P+ROC for the items in the item set (A), the "coldest" among the item sets of the cold sellers. 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 items. 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+ROC are 20.9% and 42.7% at that point, respectively, we can expect that P+ROC will help us identifying twice as many potential 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 into account many factors such as resource available for the campaign, supply and stock status of the item, etc. The ranked list can also complement recommendation made by collaborative filtering recommenders by recommending more cold sellers to further increase sales.

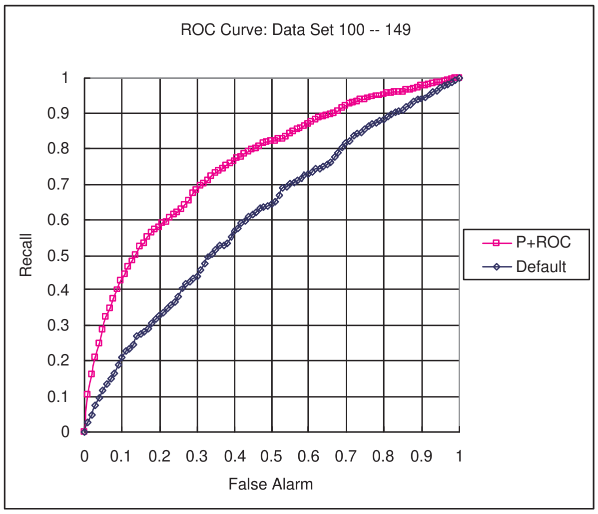

Figure 2: Comparison of the ROC curves of DEFAULT and P+ROC.

RELATED WORK

Support Vector Machines

In this paper, we combine support vector machines (SVM) and boosting to train the recommender system. Originally proposed by Vapnik [13], 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 the rare class problem for SVM is sample balancing, hierarchical SVM framework [15]. 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.

Boosting

Boosting [11] uses resampling techniques to learn a set of classifiers, and linearly combine 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 [12], which minimizes the error rate of the whole

Table 1: Average AUC scores of the nine approaches in the four unsought item classes. Best results in a data set are bolded.

Data set	DEFAULT	SVM15000	U+SVM	P+SVM	U+U	U+E	U+ROC	P+U	P+E	P+ROC
100–149	0.609	0.645	0.531	0.694	0.638	0.638	0.657	0.757	0.757	**0.758**
150–199	0.612	0.639	0.540	0.676	0.666	0.666	0.681	0.756	**0.757**	**0.757**
200–299	0.613	0.633	0.581	0.691	0.726	0.726	0.731	**0.763**	0.763	**0.763**
300–399	0.616	0.641	0.611	0.690	0.728	0.728	0.729	**0.753**	0.752	**0.753**

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 [6, 8]. DataBoost-IM, another method to deal with the rare class problem, is to synthesize more positive examples using the previously learned classifiers [3]. Currently, we adopt the method similar to AdaUBoost to initialize weights of training examples.

CONCLUSION

We have presented our item-triggered recommendation approach that predicts customer lists for cold sellers. Each customer list contains customers sorted by their probability to purchase the corresponding item. We believe that item-triggered recommendation can complement collaborative filtering-based customer-triggered recommendation by recommending cold sellers to further increase sales. For cold sellers, we will need to deal with the rare class problem to train the recommender system. Experimental results show that our approach, combination of SVM and boosting, seems promising for this problem. From the experimental results, we conclude that in terms of the AUC scores, (1) SVM outperforms shopping frequency; (2) using balanced positive and negative examples is better than using imbalanced ones for SVM; (3) SVM ensembles perform better than the variants with a single SVM.

Our future work includes to further enhance the training algorithm for cold seller recommendation. Besides, we will try to include other information such as demographical data and content-based attributes in the feature set to represent a customer to improve the recommendation.

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. P. Flach. The many faces of ROC analysis in machine learning. Tutorial in the Twenty-first International Conference on Machine Learning, Banff, Alberta, Canada., 2004.
3. H. Guo and H. L. Viktor. Learning from imbalanced data sets with boosting and data generation: the databoost-im approach. *SIGKDD Explor. Newsl.*, 6(1):30–39, 2004.
4. 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.
5. C.-N. Hsu, H.-H. Chung, and H.-S. Huang. Mining skewed and sparse transaction data from personalized shopping recommendation. *Machine Learning*, 57:35–59, 2004.
6. G. Karakoulas and J. Shawe-Taylor. Optimizing classifiers for imbalanced training sets. In *Proceedings of Neural Information Processing Workshop, NIPS'98*, 1999.
7. R. D. Lawrence, G. S. Almasi, V. Kotlyar, M. S. Viveros, and S. Duri. Personalization of supermarket product recommendations. *Data Mining and Knowledge Discovery*, 5:11–32, 2001.
8. J. Leskovec and J. Shawe-Taylor. Linear programming boosting for uneven datasets. In *Proceedings of the Twentieth International Conference on Machine Learning*, pages 351–358, 2003.
9. P. Resnick, N. Iacovou, M. Suchak, P. Bergstorm, and J. Riedl. GroupLens: An Open Architecture for Collaborative Filtering of Netnews. In *Proceedings of ACM Conference on Computer Supported Cooperative Work*, pages 175–186, 1994.
10. B. Sarwar, G. Karypis, J. Konstan, and J. Reidl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the tenth international conference on World Wide Web*, pages 285–295. ACM Press, 2001.
11. R. E. Schapire. The strength of weak learnability. *Machine Learning*, 5(2):197–227, 1990.
12. R. E. Schapire and Y. Singer. Improved boosting algorithms using confidence-rated predictions. *Machine Learning*, 37:297–336, 1999.
13. V. Vapnik. *The Nature of Statistical Learning Theory*. Springer Verlag, New York, 1995.
14. T.-F. Wu, C.-J. Lin, and R. C. Weng. Probability estimates for multi-class classification by pairwise coupling. *Journal of Machine Learning Research*, 5:975–1005, 2004.
15. R. Yan, Y. Liu, R. Jin, and A. Hauptmann. On predicting rare classes with SVM ensembles in scene classification. In *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing(ICASSP'03)*, volume 3, pages 21–24, 2003.