GreedyBoost: An Accurate, Efficient and Flexible Ensemble Method for B2B Recommendations

Weipeng Zhang
IBM Research - China
javawebzwp@gmail.com

Tobias Enders
IBM Chief Analytics Office
tobias.enders@us.ibm.com

Dongsheng Li
IBM Research - China
ldsli@cn.ibm.com

Abstract

Recommender systems have achieved great success in finding relevant products and services for individual customers, e.g. in B2C markets, during recent years. However, due to the diversity of enterprise clients' requirements it is still an open question on how to successfully apply existing recommendation techniques in the B2B domain. This paper presents GreedyBoost — an accurate, efficient and flexible ensemble method for product and service recommendations in the B2B domain. Given a set of base models, GreedyBoost can sequentially add base models to the ensemble by a linear approach to minimize training error, so that the ensemble process is efficient. Meanwhile, GreedyBoost does not have any special requirement on base models and evaluation metrics, so that any kind of client requirements and sale & distribution purposes can be adapted. Experimental results on real-world B2B data demonstrate that GreedyBoost can achieve higher recommendation accuracy compared with two popular ensemble methods.

1. Introduction

With an increase in the overall competition in the marketplace, companies across all industries are searching for new and innovative ways to grow their business. Growth can be defined in a number of ways. For some this means an increase in profit or revenue, others define it as growing their customer base. One of the main challenges in achieving this goal is to better understand the customers as well as to offer products and services that are demand – at the right time and place.

For decades, enterprises have explored many ways to analyze consumer behavior and trends, for example through the use of focus groups, interviews and surveys [1]. These insights are used for creating personas, which can be targeted in advertising and other promotional campaigns. With a steady increase in the number of customers to be served and the volume of data available, companies are looking for a more automated way to better understand the needs of their clients.

In recent years, recommender systems have become more and more popular to process vast amounts of client data and to make specific product and service recommendations for individuals [2], [3]. This technique has been used very successfully in the B2C space — Amazon [4] and Alibaba [5] are just two of the many examples. However, the number of use cases in the B2B world is very limited so far.

Main challenges of recommender systems in B2B include the limited availability of data - especially for new clients with no previous business interaction, offering complexity as well as multiple data type integration and scalability [6]. The shortage of data creates the necessity for models to utilize information that is not directly related to a particular client but can be inferred. An additional challenge is that in contrast to making recommendations for an individual person (B2C), a targeted organization can be very diverse in its structure and needs. Each of the divisions and, in larger enterprises, each geographic region, may require a different set of recommendations.

Our analysis has shown that currently available base and ensemble models do not achieve a high level of accuracy when it comes to product and service recommendations. With this problem statement in mind, we explored different ways to improve the level of accuracy and precision of the ensemble model. The outcome of this work is presented in this paper along with a way to implement and evaluate the algorithm. We refer to the developed algorithm and the application of the same as GreedyBoost.

The rest of this paper is structured into 5 sections: related work, algorithm design, analysis, experiments and conclusion. Section 2 gives a brief overview of the existing research around recommender systems and discuss limitations of the current work. Section 3 first presents three classic collaborative filtering algorithm, which are adopted as base models in the ensemble, and then presents the proposed GreedyBoost methods in detail. Algorithm design is followed by Section 4, which examines the two key characteristics of of GreedyBoost: (1) convergence and (2) computation complexity. Section 5 evaluates the proposed method in the context of real-world data sets. Finally, we conclude the paper in Section 6.

2. Related Work

Collaborative filtering (CF) has become one of the most popular recommendation algorithms during recent years [7] due to its high accuracy and efficiency. Collaborative filtering-based recommender systems have been applied in many domains, e.g., product recommendation [8], news recommendation [9], [10], video recommendation [11], music recommen-

URI: http://hdl.handle.net/10125/41342
ISBN: 978-0-9981331-0-2

dation [12] etc. Generally, collaborative filtering methods can be classified into two main categories [7], [13]: memory-based collaborative filtering and model-based collaborative filtering. Memory-based CF methods can build some correlations of clients/products based on clients' historical records and then adopt such correlations to predict their future interests [14], [8], [15], [16], [17]. On the contrary, model-based CF methods first train a model based on clients' historical records, and then predict the future interests of clients based on the model [18], [19], [20], [21]. Most of existing model-based collaborative filtering methods rely on both positive ratings and negative ratings of clients on products to train the model, which are not directly applicable to our scenario because there is no negative ratings in B2B markets. Therefore, the base models in this paper are selected from memory-based collaborative filtering methods, which do not suffer from the "missing negative rating" issue.

Ensemble methods have been proved to be more accurate than single models, and two of the most popular ensemble methods are boosting and bagging [22]. Boosting methods can integrate the power of a set of "weak" learners to achieve a learner with better performance [23]. Gradient boosting [24] is one of popular boosting method which can construct additive learning models by sequentially fitting a simple parameterized function (base learner) to minimize residuals by some loss function at each iteration. Another popular method is AdaBoosting [25], which can set higher weights to wrongly predicted examples during iterations to help choose base learners to minimize the overall training error. The above boosting methods can achieve good performance if the base learners can be trained or easily obtained. However, this cannot be easily guaranteed in top-N recommendation application, in which no negative examples are available for model learning. Different from boosting, bagging method [26] can build base learners by changing the set of training examples and then use the average outputs of all the base learners, so that ensemble results can achieve better generalization performance. Bagging can achieve good performance if the base learners are not very stable [27], e.g., decision-tree or neural network. However, memory-based collaborative filtering methods are similar to kNN methods, which have been proved to be stable. Therefore, bagging cannot substantially improve recommendation accuracy as in other applications.

In addition, some of the existing ensemble methods have special requirements about the base learners. For instance, gradient boosting decision tree [28] can only deal with the case that all the base learners are decision trees. On the contrary, the proposed GreedyBoost method can be adopted to integrate any kinds of base learners, e.g., user-based collaborative filtering, item-based collaborative filtering and Naïve Bayes recommendation in this work. This flexibility is important because many real-world problem cannot be easily solved by one kind of methods but a variety of different kinds of methods.

3. Algorithm Design

This section first presents the three collaborative filtering models that are adopted as base models in GreedyBoost, i.e., user-based collaborative filtering (UBCF) [14], item-based collaborative filtering (IBCF) [15] and Naïve Bayes recommendation algorithm (NBR) [29]. Note that, the GreedyBoost method is orthogonal to base models, so that any other base model can be adopted by GreedyBoost. Then, the technique details of GreedyBoost algorithm is presented.

3.1. Base Models

3.1.1. User-based Collaborative Filtering. User-based collaborative filtering (UBCF) method is based on the idea that users with certain interests in the past will be likely to have similar interests in the future. Thus, we can observe the similarities among users in the past, then predict the future interest of a target user based on the decision of its similar neighbors. There are two key steps in the above process: 1) measure the similarity among different users and 2) compute the recommendation score based on the decision of neighbors.

Similarity computation. There are a variety of methods for computing similarities, and we adopt two of the most popular ones for top-N recommendation, i.e., Cosine similarity and Jaccard similarity. Given two users u and v, let $r_{u,i}$ be the rating of user u on item i and n be the number of items. Note that, the value of $r_{u,i}$ is binary in top-N recommendation, i.e., $r_{u,i} = 1$ if u purchased i before and $r_{u,i} = 0$ otherwise. Then, the Cosine similarity between u and v can be computed as follows:

$$cos(u,v) = \frac{\sum_{i=1}^{n} r_{u,i} r_{v,i}}{\sqrt{\sum_{i=1}^{n} r_{u,i}^2} \sqrt{\sum_{i=1}^{n} r_{v,i}^2}} \tag{1}$$

The Jaccard similarity between u and v can be computed as follows:

$$jaccard(u,v) = \frac{|I_u \cap I_v|}{|I_u \cup I_v|} \tag{2}$$

where I_u (I_v) is the set of items that are rated by u (v) before.

Rating computation. Given a target user u and the similarities between u and all other users, we can select the set of most similar users to u as "neighbors", and then predict the rating of u on unrated items based on the opinions of the neighbors by a weighted average as follows [14]:

$$\hat{r}_{u,i} = \frac{\sum_{v \in N_u} sim(u,v) r_{v,i}}{\sum_{v \in N_u} sim(u,v)} \tag{3}$$

where N_u is the set of neighbors of u. $sim(x,y)$ is a user similarity measure, which can be computed by Equation 1 or Equation 2.

After obtaining the predicted rating $\hat{r}_{u,i}$ for each of the unrated items of user u, we can rank all the unrated items by the predicted rating in a descending order and recommend the top k items to the user.

3.1.2. Item-based Collaborative Filtering. Item-based collaborative filtering method works based on the idea that users will be interested in items that are similar to the items they have purchased before. Thus, we can observe the similarities among items, then predict the future interest of a target user based on the set of items it purchased. Similarly, there are also two key steps in the above process: 1) measure the similarity among different items and 2) compute the recommendation score based on the purchase history of the target user.

Similarity computation. Item similarities can be computed just like user similarities. Given two items i and j, the Cosine similarity between i and j can be computed as follows:

$$cos(i,j) = \frac{\sum_{u=1}^{m} r_{u,i} r_{u,j}}{\sqrt{\sum_{u=1}^{m} r_{u,i}^2} \sqrt{\sum_{u=1}^{m} r_{u,j}^2}} \tag{4}$$

where m is the number of users. The Jaccard similarity between i and j can be computed as follows:

$$jaccard(i,j) = \frac{|U_i \cap U_j|}{|U_i \cup U_j|} \tag{5}$$

where U_i (U_j) is the set of users that rated by i (j) before.

Rating computation. Given a target user u and the similarities among all items, we can predict the rating of u on unrated item i based on the similarities between i and the set of items that u rated before by a weighted average as follows [15]:

$$\hat{r}_{u,i} = \frac{\sum_{j \in I_u} sim(i,j) r_{u,j}}{\sum_{j \in I_u} sim(i,j)} \tag{6}$$

where I_u is the set of items that are rated by u. $sim(x,y)$ is an item similarity measure, which can be computed by Equation 4 or Equation 5.

Similar to UBCF, after obtaining all the predicted ratings, we can rank all the unrated items by the predicted rating in a descending order and recommend the top k items to the user.

3.1.3. Naïve-Bayes Recommendation. Naïve Bayes recommendation algorithm is a kind of content-based method, which adopts user/item features to generate the recommendations, e.g., based on firmographics information of companies. Given a target user u with a set of content features $F_u = \{f_1, ..., f_t\}$, the recommendation score for an unrated item i can be computed as follows [29]:

$$\hat{r}_{u,i} = \Pr(i|F) = \frac{\Pr(i)\Pr(F|i)}{\Pr(F)} \tag{7}$$

Then, based on the independence assumption of each feature in F, we have

$$\hat{r}_{u,i} = \frac{\Pr(i)\prod_{f \in F}\Pr(f|i)}{\prod_{f \in F}\Pr(f)} \tag{8}$$

$\Pr(i)$ is the probability of item i being purchased in the past, which is defined as follows:

$$\Pr(i) = \frac{N_1(i) + \delta}{\sum_j N_1(j) + n\delta} \tag{9}$$

where $N_1(i)$ is the number of times that item i was purchased in the past and δ is the Laplacian smoothing parameter to avoid the "0"-probability issue.

$\Pr(f)$ is the probability that feature f appeared among all the users, which is defined as follows:

$$\Pr(f) = \frac{N_2(f) + \delta}{\sum_{f'} N_2(f') + t\delta} \tag{10}$$

where $N_2(f)$ is the number of times that feature f appeared among all users.

$\Pr(f|i)$ is the probability of the feature f among the users who purchased i, which is defined as follows:

$$\Pr(f|i) = \frac{N_3(i,f) + \delta}{\sum_{f'} N_3(i,f') + t\delta} \tag{11}$$

where $N_3(i,f)$ is the number of times that feature f appears among the users who purchased i.

3.2. GreedyBoost Method

Related works [28], [26], [22], [21], [30] have theoretically and empirically proved that ensemble methods can significantly improve model performance compared with base methods, because the generalization error of ensemble model can be reduced by combining the advantages of different base models. To this end, we adopt the ensemble idea in B2B recommendation and propose an accurate, efficient and flexible ensemble method, namely GreedyBoost, which can greedily combine the outputs of a number of base models by a weight optimization algorithm.

Given a set of base models $M = \{m_1, m_2, ..., m_k\}$, where k is the number of base models, the goal of the proposed GreedyBoost method is to find the optimal weights $\{w_1, ..., w_k\}$ to combine the outputs of all base models in M. Therefore, the output of the ensemble model can be defined as follows:

$$m^* = \sum_{i=1}^{k} w_i * m_i \tag{12}$$

Following the idea of boosting, the proposed GreedyBoost method iteratively add new base models in the ensemble while ensuring that the weight of newly added base model can minimize the training error. As illustrated in Figure 1, the proposed GreedyBoost method consists of three key steps:

1) Let $m^* = m_1$ (assuming that m_1 is the base model with lowest training error);
2) Update m^* by adding the best base model $m' \in M$ ensuring that m' can minimize the residual error of m^*, and then compute the corresponding weight w by optimize the following problem $w = \arg\min_{w'} E((1 - w)m^* + wm)$ ($E(\cdot)$ is an evaluation function to measure how accurate a model is);
3) Repeat step 2 until the error of m^* converges or the maximum number of iterations reaches.

Algorithm 1 formally describes the detailed procedure of the proposed GreedyBoost algorithm. For each step, a base model

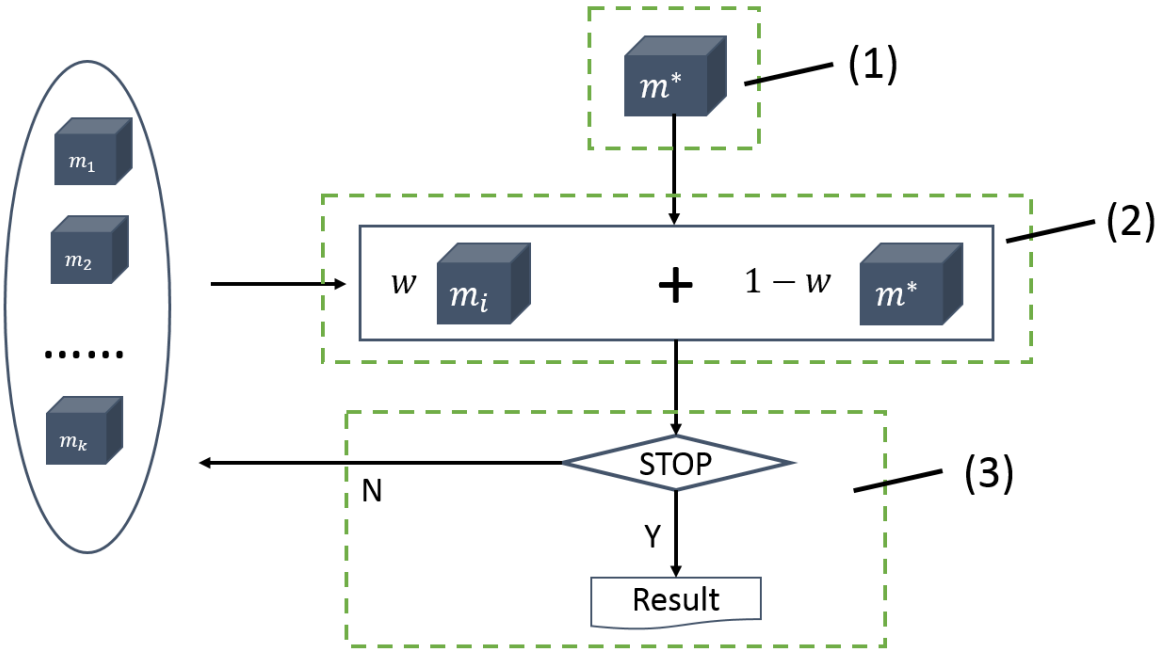

Fig. 1. **High-level overview of the proposed GreedyBoost method**

is added to the ensemble by a weighted average to integrate the base model with the current ensemble model, which makes it very fast to add a base model. Moreover, the evaluation function $E(\cdot)$ does not need to be specified in GreedyBoost, which allows us to optimize the ensemble model with any kind of evaluation metrics, i.e., the GreedyBoost method is flexible and can be adopted in any kind of top-N recommendation problems.

4. Analysis

4.1. Generalization Performance

This section proves that the proposed GreedyBoost method can generalize well with sufficient number of base models, i.e., the good performance of the ensemble model on training data can be preserved on test data. Here, we first prove the case that all the base models are of equal weight in the ensemble.

Theorem 1 *Let the errors of k different base models $m_1, ..., m_k$ be k independent random variables, $a_i \leq X_i \leq b_i$, the expectation of the errors be m and $\bar{m} = \frac{1}{k}\sum_{i=1}^{k} m_i$. For any $\epsilon, \delta > 0$, if we choose $k \geq \sqrt{\frac{\sum_{i=1}^{k}(b_i-a_i)^2 \log \delta/2}{-2\epsilon^2}}$ we have*

$$\Pr[|m - \bar{m}| < \epsilon] \geq 1 - \delta. \tag{13}$$

Proof: Based on Hoeffding's inequality, we have

$$\Pr[m - \bar{m} \geq \epsilon] \leq \exp\{\frac{-2k^2\epsilon^2}{\sum_{i=1}^{k}(b_i - a_i)^2}\} \tag{14}$$

Similarly, we have

$$\Pr[\bar{m} - m \geq \epsilon] \leq \exp\{\frac{-2k^2\epsilon^2}{\sum_{i=1}^{k}(b_i - a_i)^2}\} \tag{15}$$

Algorithm 1 GreedyBoost

Require: A set of base models $M = \{m_1, m_2, ..., m_k\}$, evaluation metric $E(m_i), i \in \{1, ..., k\}$
Ensure: Final model m^*

```
choose m_0 ∈ M as the base model with best performance evaluated by E;
S* = E(m_0);
while S* does not converge do
    Let ST = ∅ be the data structure to store the intermediate recommendation accuracy during training;
    Let TM = ∅ be the data structure to store the intermediate weight and the selected base model pairs during training;
    for each i ∈ [1, k] do
        w = arg min_{w'} {w' * m* + (1 − w') * m_i};
        st_i = E(w * m* + (1 − w) * m_i);
        put st_i into ST;
        put (m_i, w) into TM;
    end for
    st* = max_{st∈ST} st;
    if st* > S* then
        S* = st*;
        m* = w * m* + (1 − w) * m_i, where (m_i, w) is the corresponding model and weight
    else
        break;
    end if
end while
```

Then, combining the above two inequalities, we have

$$\Pr[|m - \bar{m}| \geq \epsilon] \leq 2\exp\{\frac{-2k^2\epsilon^2}{\sum_{i=1}^{k}(b_i - a_i)^2}\} \quad (16)$$

This is equivalent to

$$\Pr[|m - \bar{m}| < \epsilon] \geq 1 - 2\exp\{\frac{-2k^2\epsilon^2}{\sum_{i=1}^{k}(b_i - a_i)^2}\} \quad (17)$$

To ensure that $\Pr[|m - \bar{m}| < \epsilon] \geq 1 - \delta$, we have $2\exp\{\frac{-2k^2\epsilon^2}{\sum_{i=1}^{k}(b_i-a_i)^2}\} \leq \delta$. Solving the inequality, we have

$$k \geq \sqrt{\frac{\sum_{i=1}^{k}(b_i - a_i)^2 \log \delta/2}{-2\epsilon^2}} \quad (18)$$

This proves the conclusion in the theorem. □

Note that, Theorem 1 is based on the bounded error of all base models. This means that if the errors of all base models are bounded, then the error of their ensemble is bounded. Otherwise, if any of the base models has an unbounded error, the above theorem does not hold. If the base models run on datasets with limited examples, the errors of the base models are always bounded. Therefore, we can claim that the above theorem will hold in real-world problems in which datasets always have limited examples.

The above Theorem 1 states that if we can choose enough independent base models then the error of the averaged outputs of all base models can be bounded. If we take the averaged outputs of all base models as a kind of ensemble, then the above Theorem 1 also indicates that the error of the averaged ensemble model is also bounded. Moreover, the above proof actually does not require each base model to be of equal weight, i.e., unequal weights in the averaging can also derive similar results. Therefore, we can conclude that the error of the proposed GreedyBoost method is bounded. Then, based on the argument that $\hat{m} \leq \frac{1}{k}\sum_{i=1}^{k} m_i$ [30] ($\hat{m}$ is the generalization error of the ensemble), we know that the generalization error of ensemble cannot be larger than the average error of all base models, i.e., the generalization error of the ensemble is bounded. Therefore, we can conclude that the generalization error of GreedyBoost method is also bounded. This indicates that, if the ensemble model can achieve good performance on training data, it can also achieve similarly good performance on test data.

4.2. Complexity Analysis

Following, we only analyze the the computation complexity of the proposed GreedyBoost method. This is because GreedyBoost only requires the outputs of all base models, so that all base models can be trained offline. The training of the GreedyBoost method is efficient, because adding a base model to the ensemble model only requires a linear combination of the output of the previous ensemble model and the output of the selected base model. Therefore, the computation of training in the GreedyBoost method is $O(L)$ per iteration, where L is the number of examples in the cross-validation data. Suppose that there are k different base models in M, steps 5 to 10 in Algorithm 1 take k times per iteration. If the GreedyBoost method runs T iterations in total before termination, the total computation cost of training the ensemble model is only $O(LKT)$ (assuming that the evaluation metric can be computed in $O(L)$). If the computation cost of evaluation metric is more complex than $O(L)$, e.g., computation of AUC value requires sorting and thus is $O(LlogL)$), the computation cost of GreedyBoost is $O(LlogLKT)$.

4.3. Reference to B2C Recommendations

It should be noted that the proposed GreedyBoost method is more suitable to B2B recommendations than B2C recommendations, due to the often different application scenarios in the B2C and B2B context. There are three aspects that summarize the key differences: 1) Accuracy. B2B data are sparse so that complex models may easily overfit due to incomplete and noisy training data [31], [32]. However, GreedyBoost adopts a linear model to integrate the base models and achieves good generalization performance as analyzed in Section 4.1. This guarantees that the recommendations from GreedyBoost can be reliable in real-world B2B sales and markets as it does not easily overfit towards the limited training data. 2) Efficiency. Users are not changing with a high frequency in the B2B domain, i.e., on a weekly or even monthly basis. Therefore, we can pre-train a set of base models, and then GreedyBoost can serve different goals based on the same pre-trained base models. This is different for B2C recommendation, in which user interests will often update/change on a daily basis. 3) Flexibility. Requirements are very dynamic in B2B sales and marketing campaigns, so that the method should provide recommendations based on different optimization goals and scenarios, e.g. for precision, coverage, revenue. GreedyBoost can achieve any optimization goal by optimizing towards those goals based on the same base models. In contrast, for B2C applications, recommendations will mostly have one optimization goal so that no such flexibility is required. Moreover, some existing ensemble methods, e.g. gradient boosting, cannot support various kinds of optimization goals due to the limitation of the algorithms. For instance, gradient boosting cannot optimize towards loss functions which are not differentiable, e.g. area under ROC curve. In conclusion, we can say that although the proposed GreedyBoost method can be applied in the B2C domain as well, it is more suitable to application areas in the B2B space.

5. Experiments

5.1. Experimental Setup

5.1.1. Dataset Description. In this section, we evaluate the proposed GreedyBoost method on a real-world dataset, which is collected from a commercial B2B company. The company develops, manufactures and markets computer hardware and software, and also provides IT-related services. Table 1

describes the detailed information of the dataset. Table 2 describes customer features, i.e., country, industry and region, which are adopted in the Naïve Bayes recommendation algorithm. We select customer purchase records from 2011 to 2014 as training data, and use the data in 2015 as testing. During the training of the ensemble model, we randomly select 20% data from training set as validation set, which is used to determine the optimal weight of each base model.

TABLE 1. **Dataset Description (purchase records)**

#records	#customers	#products	period
$\sim$ 1.3 million	$\sim$ 180k	627	2011 - present

TABLE 2. **Dataset Description (customer features)**

#countries	#regions	#industries
165	7	24

5.1.2. Evaluation Metrics. In top-N recommendation scenario, we can regard the recommendation as binary classification problem, because the label of each client-product pair in the test set is 1 or 0. i.e., purchased or not. And the recommendation scores can be used to predict the probability of clients purchasing products. To this end, we use Receiver Operator Characteristic (ROC) curves, which shows how the number of correctly classified positive examples varies with the number of incorrectly classified negative examples. However, ROC curves can present an overly optimistic view of an algorithm's performance if there is a large skew in the class distribution. Therefore, area under ROC curves (AUC) can be used to evaluate the overall performance of each algorithm.

Meanwhile, we also evaluate the proposed GreedyBoost method using precision and recall metrics, which are two commonly used in evaluating top-N recommendation algorithms. Precision-Recall curves can show the trend of recall varying with precision, and the area under PR curves (AUPR) can be used to evaluate the overall performance of each algorithm. For both the two evaluation metrics, higher value means better accuracy. Precision and Recall can be computed as follows:

$$Precision = \frac{|I_r \cap I_u|}{|I_u|} \quad (19)$$

$$Recall = \frac{|I_r \cap I_u|}{|I_r|} \quad (20)$$

where I_r is the set of recommended items and I_u is the set of items that are liked by user u.

The ROC curve represents the relationship between sensitivity and specificity, in which sensitivity is the same as recall but specificity is slightly different from precision. As pointed out by Davis and Goadrich [33], Precision-Recall curves provide more informative pictures of algorithm performance when dealing with highly skewed datasets. Note that, GreedyBoost can optimize towards AUC and AUPR independently. Therefore, if the targeted dataset is highly skewed, we can let GreedyBoost optimize the ensemble model towards AUPR. Otherwise, we can let GreedyBoost optimize towards AUC.

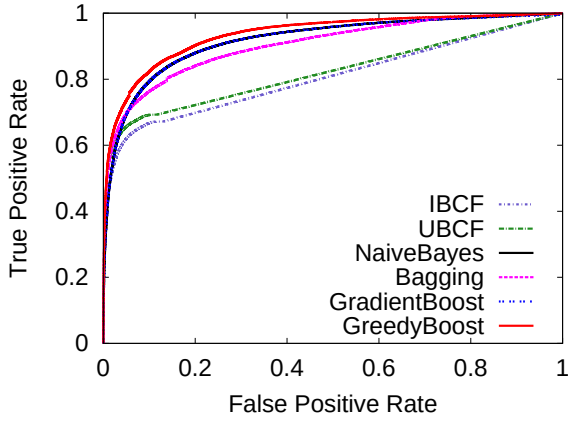

Fig. 2. **ROC curve comparison**

5.2. Base Model Selection

The performance of the proposed GreedyBoost method is determined by the selection of base models. In this section, we adopt the three kinds of base models as described in Section 3.1. For UBCF and IBCF, we vary the number of neighbors to form diverse base models. In particular, we adopt 100, 200 and 300 as the numbers of neighbors in UBCF to form three different UBCF-based models and adopt 50, 100, and 150 as the numbers of neighbors in IBCF to form three different IBCF-based models. In total, we have seven base models in GreedyBoost: three UBCF-based models, three IBCF-based models and a NaïveBayes model.

5.3. Accuracy Comparison

In this section, we compare GreedyBoost with two classic ensemble methods, i.e., Bagging [26] and Gradient Boosting [28], as well as all the base models adopted in the ensemble on the two aforementioned evaluation metrics AUC and AUPR. Table 3 shows the AUC scores of three base models and three different ensemble methods. We can see from the results that GreedyBoost achieves better performance than all the other compared methods. Figure 2 shows the ROC curve for all these models, and the same trend can be observed.

TABLE 3. **Comparison of Area under ROC Curve (AUC)**

Model	AUC Score
User-based Collaborative Filtering	0.8172
Item-based Collaborative Filtering	0.8050
Naïve-Bayes	0.9181
Bagging	0.9044
Gradient Boosting	0.9178
GreedyBoost	**0.9361**

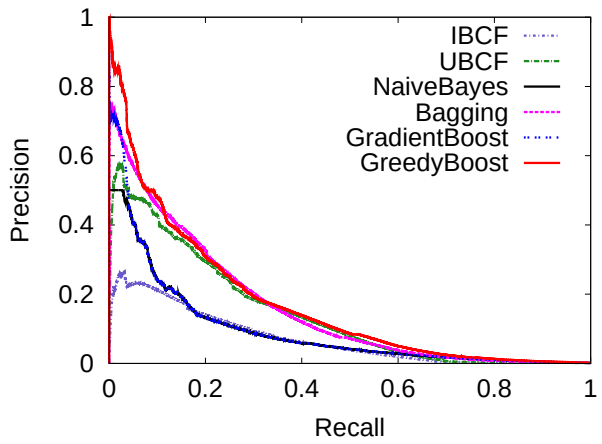

Fig. 3. **PR curve comparison**

Table 4 shows the area under PR curve (AUPR) of all three ensemble models and three base models. Note that, the precision-recall curves show the results averaged over all the customers, so that the trends are slightly different from those in the ROC curve. However, the proposed GreedyBoost method still outperforms all the other compared methods. Figure 3 shows the PR curves of all the compared methods.

TABLE 4. **Comparison of Area under PR Curve**

Model	Area under PR
User-based Collaborative Filtering	0.1415
Item-based Collaborative Filtering	0.0687
Naïve-Bayes	0.0902
Bagging	0.1557
Gradient Boosting	0.0971
GreedyBoost	**0.1703**

Overall, the proposed GreedyBoost method outperforms the other five methods by 2.0% - 16.3% relatively in terms of AUC and 9.4% - 147.9% relatively in terms of AUPR, respectively. The main reasons that the proposed GreedyBoost method can achieve better accuracy are as follows. First, the proposed method can directly optimize towards the given evaluation metric. On the contrary, the bagging method does not have such property because it tries to improve the model accuracy by preventing base models from overfitting using randomization. Gradient boosting method can minimize training error, but lower training error, e.g., lower mean absolute error, does not mean higher AUC or AUPR because they are not directly correlated [34]. Secondly, the proposed method has good generalization performance, so that it will not easily overfit as in gradient boosting [28].

5.4. Efficiency Analysis

To further evaluate the efficiency of the proposed GreedyBoost method, we compare the computation time of GreedyBoost with Bagging and Gradient Boosting. In this experiments, the maximum number of iteration is set to 200 for GreedyBoost and Gradient Boosting and convergence threshold is set to 0.0001. For all the three compared methods, we set the number of base models to 7. The numbers reported in Table 5 are the average results over 5 separate runs. Note that computation times for the base model generation process are not included in Table 5 because they can be pre-trained in B2B recommendations. We can see from Table 5 that the computation time for GreedyBoost and Gradient Boosting are much higher than Bagging, which is because training is required to learn model parameters for GreedyBoost and Gradient Boosting. And for Bagging, only a simple average is required to do the ensemble. However, the accuracy of Bagging converges with the number of base models, which can typically be as large as several hundreds [22]. Since the training time of base models is much longer than the training time of ensemble, the proposed GreedyBoost method can achieve a much lower overall computation overhead compared to Bagging. GreedyBoost and Gradient Boosting are of similar computation complexity. But compared with Gradient Boosting, GreedyBoost can reduce computation time by approximately 1/4 in the experiments, which is mainly due to faster convergence speed. In conclusion, GreedyBoost can achieve a good level of efficiency compared with popular ensemble methods.

TABLE 5. **Efficiency Comparison**

Model	Computation Time (seconds)
Bagging	13.1
Gradient Boosting	1,206.9
GreedyBoost	902.3

6. Conclusion

Recommender systems have been proven to be a valuable support in the sales and customer interaction process. They allow enterprises to process vast amounts of client information and to make meaningful recommendations. This work addresses one of the main challenges of recommender systems, which is to optimize the level of accuracy and precision. The analysis and experiments in this paper proves that the proposed implementation method for this algorithm — GreedyBoost — is a more accurate, efficient and flexible way for integrating different kinds of recommendation base models in the B2B domain.

Experimental results on real-world business data demonstrate that the proposed method can achieve higher accuracy and efficiency than two well-known ensemble methods. In addition, the GreedyBoost model performs significantly better than each of the base models. Meanwhile, the proposed method is orthogonal to base learners, which shows high flexibility by making it possible to integrate any kind of recommendation methods into the ensemble.

The future work for improving the proposed GreedyBoost method will focus on the following two aspects. First, the GreedyBoost method does not consider model-based collabo-

rative filtering methods due to low compatibility. Therefore, incorporating model-based collaborative filtering methods, e.g., matrix approximation-based method, to improve the model accuracy will be one of the possible extensions. Secondly, the weights of base models in GreedyBoost are globally optimized, which may be optimal in terms of all users but may not be optimal in terms of individual users. Therefore, another extension will be to give each user a unique set of optimal weights in the ensemble model.

References

[1] G. Zaltman, *How customers think: essential insights into the mind of the market*. Harvard Business School Press, 2003.

[2] Q. Zhao, Y. Zhang, D. Friedman, and F. Tan, "E-commerce recommendation with personalized promotion," in *Proceedings of the 9th ACM Conference on Recommender Systems*, ser. RecSys '15. ACM, 2015, pp. 219–226.

[3] J. Wang and Y. Zhang, "Opportunity model for e-commerce recommendation: right product; right time," in *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval*, ser. SIGIR '13. ACM, 2013, pp. 303–312.

[4] J. Mangalindan. (2012) Amazon's recommendation secret. [Online]. Available: http://fortune.com/2012/07/30/amazons-recommendation-secret/

[5] Alibaba. (2016) Financials and metrics. [Online]. Available: http://www.alibabagroup.com/en/ir/financial/

[6] X. Zhang and H. Wang, "Study on recommender systems for business-to-business electronic commerce,," *Communications of the IIMA*, vol. 5, no. 4, 2005.

[7] G. Adomavicius and A. Tuzhilin, "Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions," *IEEE Transactions on Knowledge and Data Engeering*, vol. 17, no. 6, pp. 734–749, 2005.

[8] G. Linden, B. Smith, and J. York, "Amazon.com recommendations: item-to-item collaborative filtering," *IEEE Internet Computing*, vol. 7, no. 1, pp. 76–80, 2003.

[9] J. A. Konstan, B. N. Miller, D. Maltz, J. L. Herlocker, L. R. Gordon, and J. Riedl, "Grouplens: Applying collaborative filtering to usenet news," *Commun. ACM*, vol. 40, no. 3, pp. 77–87, 1997.

[10] A. S. Das, M. Datar, A. Garg, and S. Rajaram, "Google news personalization: Scalable online collaborative filtering," in *Proceedings of the 16th International Conference on World Wide Web*, ser. WWW '07. ACM, 2007, pp. 271–280.

[11] J. Davidson, B. Liebald, J. Liu, P. Nandy, T. Van Vleet, U. Gargi, S. Gupta, Y. He, M. Lambert, B. Livingston, and D. Sampath, "The youtube video recommendation system," in *Proceedings of the Fourth ACM Conference on Recommender Systems*, ser. RecSys '10. ACM, 2010, pp. 293–296.

[12] N. Koenigstein, G. Dror, and Y. Koren, "Yahoo! music recommendations: Modeling music ratings with temporal dynamics and item taxonomy," in *Proceedings of the Fifth ACM Conference on Recommender Systems*, ser. RecSys '11. ACM, 2011, pp. 165–172.

[13] X. Su and T. M. Khoshgoftaar, "A survey of collaborative filtering techniques," *Advances in Artificial Intelligence*, vol. 2009, Jan. 2009.

[14] J. L. Herlocker, J. A. Konstan, A. Borchers, and J. Riedl, "An algorithmic framework for performing collaborative filtering," in *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, ser. SIGIR '99, 1999, pp. 230–237.

[15] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Item-based collaborative filtering recommendation algorithms," in *Proceedings of the 10th International Conference on World Wide Web*. ACM, 2001, pp. 285–295.

[16] J. Wang, A. P. de Vries, and M. J. T. Reinders, "Unifying user-based and item-based collaborative filtering approaches by similarity fusion," in *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, ser. SIGIR '06. ACM, 2006, pp. 501–508.

[17] H. Ma, I. King, and M. R. Lyu, "Effective missing data prediction for collaborative filtering," in *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, ser. SIGIR '07. ACM, 2007, pp. 39–46.

[18] T. Hofmann, "Latent semantic models for collaborative filtering," *ACM Transactions on Information Systems*, vol. 22, no. 1, pp. 89–115, 2004.

[19] A. Paterek, "Improving regularized singular value decomposition for collaborative filtering," in *Proceedings of KDD cup and workshop*, vol. 2007, 2007, pp. 5–8.

[20] Y. Koren, "Factorization meets the neighborhood: a multifaceted collaborative filtering model," in *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2008, pp. 426–434.

[21] C. Chen, D. Li, Y. Zhao, Q. Lv, and L. Shang, "WEMAREC: Accurate and scalable recommendation through weighted and ensemble matrix approximation," in *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2015, pp. 303–312.

[22] Z.-H. Zhou, *Ensemble Methods: Foundations and Algorithms*. Chapman & Hall/CRC, 2012.

[23] Y. Freund and R. Schapire, "A short introduction to boosting," *Journal-Japanese Society For Artificial Intelligence*, vol. 14, no. 771-780, p. 1612, 1999.

[24] J. H. Friedman, "Stochastic gradient boosting," *Computational Statistics & Data Analysis*, vol. 38, no. 4, pp. 367 – 378, 2002, nonlinear Methods and Data Mining.

[25] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *Journal of Computer and System Sciences*, vol. 55, no. 1, pp. 119 – 139, 1997.

[26] L. Breiman, "Bagging predictors," *Machine learning*, vol. 24, no. 2, pp. 123–140, 1996.

[27] E. Bauer and R. Kohavi, "An empirical comparison of voting classification algorithms: Bagging, boosting, and variants," *Machine learning*, vol. 36, no. 1, p. 2, 1998.

[28] J. H. Friedman, "Greedy function approximation: A gradient boosting machine." *The Annals of Statistics*, vol. 29, no. 5, pp. 1189–1232, 10 2001.

[29] K. Miyahara and M. J. Pazzani, "Collaborative filtering with the simple bayesian classifier," in *PRICAI 2000 Topics in Artificial Intelligence*. Springer, 2000, pp. 679–689.

[30] A. Krogh, J. Vedelsby *et al.*, "Neural network ensembles, cross validation, and active learning," *Advances in neural information processing systems*, vol. 7, pp. 231–238, 1995.

[31] D. Li, C. Chen, Q. Lv, J. Yan, L. Shang, and S. Chu, "Low-rank matrix approximation with stability," in *Proceedings of The 33rd International Conference on Machine Learning (ICML '16)*, 2016, pp. 295–303.

[32] C. Chen, D. Li, Y. Zhao, Q. Lv, and L. Shang, "Wemarec: Accurate and scalable recommendation through weighted and ensemble matrix approximation," in *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 2015, pp. 303–312.

[33] J. Davis and M. Goadrich, "The relationship between precision-recall and roc curves," in *Proceedings of the 23rd International Conference on Machine Learning*, ser. ICML '06. New York, NY, USA: ACM, 2006, pp. 233–240. [Online]. Available: http://doi.acm.org/10.1145/1143844.1143874

[34] G. Shani and A. Gunawardana, "Evaluating recommender systems," Tech. Rep. MSR-TR-2009-159, November 2009. [Online]. Available: http://research.microsoft.com/apps/pubs/default.aspx?id=115396