Dynamic Ensemble Re-Construction for Better Ranking

Jin Huang and Charles X. Ling

Department of Computer Science
The University of Western Ontario
London, Ontario, Canada N6A 5B7
email: {jhuang33, cling}@csd.uwo.ca

Abstract. Ensemble learning has been shown to be very successful in data mining. However most work on ensemble learning concerns the task of classification. Little work has been done to construct ensembles that aim to improve ranking. In this paper, we propose an approach to re-construct new ensembles based on a given ensemble with the purpose to improve the ranking performance, which is crucial in many data mining tasks. The experiments with real-world data sets show that our new approach achieves significant improvements in ranking over the original Bagging and Adaboost ensembles.

1 Introduction

Classification is one of the fundamental tasks in knowledge discovery and data mining. The performance of a classifier is usually evaluated by predictive accuracy. However, most machine learning classifiers can also produce the probability estimation of the class prediction. Unfortunately, this probability information is ignored in the measure of accuracy.

In many real-world data mining applications, however, we often need the probability estimations or ranking. For example, in direct marketing, we often need to promote the most likely customers, or we need to deploy different promotion strategies to customers according to their likelihood of purchasing. To accomplish these tasks we need a ranking of customers according to their likelihood of purchasing. Thus ranking is often more desirable than classification in these data mining tasks.

One natural question is how to evaluate a classifier's ranking performance. In recent years, the area under the ROC (Receiver Operating Characteristics) curve, or simply AUC, is increasingly received attention in the communities of machine learning and data mining. Data mining researchers [1, 2] have shown that AUC is a good summary in measuring a classifier's overall ranking performance. Hand and Till [3] present a simple approach to calculating AUC of a classifier for binary classification.

$$\hat{A} = \frac{S_0 - n_0(n_0+1)/2}{n_0 n_1}, \tag{1}$$

where n_0 and n_1 are the numbers of positive and negative examples respectively, and $S_0 = \sum r_i$, where r_i is the rank of the i_{th} positive example in the ranked list.

Ensemble is a general approach which trains a number of classifiers and then combines their predictions in classification. Many researches [4–6] have shown that the ensemble is quite effective in improving the classification accuracy compared with a single classifier. The reason is that the prediction error of an individual classifier can be counteracted by the combination with other classifiers. Bagging [5] and Boosting [7] are two of the most popular ensemble techniques.

Most previous work of ensemble learning is focussed on classification. To our knowledge, there is little work that directly constructs ensembles to improve probability estimations or ranking. [8] compared the probability estimations (ranking) performance of different learning algorithms by using AUC as the comparison measure and demonstrated that Boosted trees and Bagged trees perform better in terms of ranking than Neural Networks and SVMs. [9] used the boosting technique on the general preference learning (ranking) problem and proposed a new ranking boosting algorithm: RankBoost.

In this paper, we propose a novel approach to improve the ranking performance over a given ensemble. The goal of this approach is to select some classifiers from the given ensemble to re-construct new ensembles. It first uses the *k*-Nearest Neighbor method to find training data subsets which are most similar to the test set, then it uses the measure SAUC (see Section 2.2) as heuristic to dynamically choose the diverse and well performed classifiers. This approach is called DERC (Dynamic Ensemble Re-Construction) algorithm. The new ensembles constructed by this approach are expected to have better ranking performance than the original ensemble.

The paper is organized as follows. In Section 2 we give detailed description for our new algorithm. In Section 3 we perform experiments on real world data sets to show the advantages of the new algorithm.

2 DERC (Dynamic Ensemble Re-Construction) Algorithm

In an ensemble, the combination of the predictions of several classifiers is only useful if they disagree to some degree. Each ensemble classifier may perform diversely during classification. Our DERC algorithm is motivated by this diversity property of ensemble. The diversity implies that each ensemble classifier performs best in probability estimation (ranking) only in a subset of training instances. Thus given a test (sub)set, if we use the *k*-Nearest Neighbor method to find some training subsets that are most similar to it, the classifiers that perform diversely and accurately on those similar training subsets are also expected to perform well on the test (sub)set. Therefore the new ensembles constructed are expected to have better ranking performance than the original ensemble.

Our DERC algorithm involves two basic steps: finding the most similar training (sub)sets, and selecting the diverse and accurate classifiers.

Now we use Figure 1 to illustrate how DERC algorithm works. Suppose that we are given an ensemble E with multiple classifiers built on a training set S, and we have an unlabeled test set T at hand. Our goal is to select some classifiers from the ensemble E to build one or more new ensembles to perform ranking on test set T.

2.1 Finding the Most Similar Training Subsets

The first step is to stratify the test set to some equal parts and find the most similar training subsets corresponding to test partitions. Since the labels of test instances are unknown, we randomly pick a classifier from ensemble E to classify the test set T to obtain the predicted labels. Assume that we want to construct 3 new ensembles. According to the predicted class labels we stratify (partition with equal class distributions) the test set T into 3 equal sized parts: T_1, T_2, and T_3. We want to select some classifiers from ensemble E to build 3 different new ensembles which are responsible for ranking T_1, T_2 and T_3 respectively.

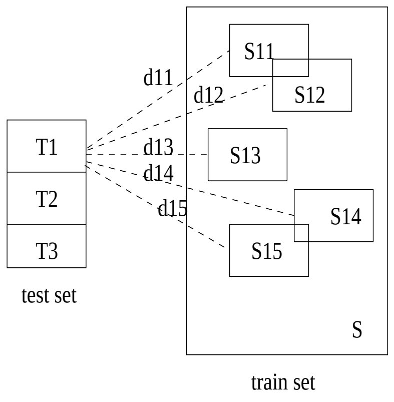

Fig. 1. An example for the similar sets.

For each stratified test subset we use the k-Nearest Neighbor method to find k subsets of training set which are most similar to that test set part. For each instance of the test subset, we compute the distances from this instance to all training instances and find the nearest k instances. We use the following method to compute the distance between two instances u and v, which are from the test subset and training dataset, respectively. Suppose that an instance has k_1 nominal attributes A_i and k_2 numerical attributes B_i. We use the simplified VDM measure proposed in [10] to compute the distance of all nominal attributes.

$$VDM(u,v) = \sum_C \sum_{i=1}^{k_1} \left(\frac{N_{A_i=a_u,C=c}}{N_{A_i=a_u}} - \frac{N_{A_i=a_v,C=c}}{N_{A_i=a_v}}\right)^2$$

where $N_{A_i=a_u}$ is the number of instances in test subset holding value a_u on attribute A_i, $N_{A_i=a_u,C=c}$ is the number of instances in test subset which are predicted belonging to class c and hold value a_u on attribute A_i. Here note that since test set is unlabeled, we use the class labels predicted in the first step.

We simply use the Euclidean distance to compute the difference of numerical attributes. $ED(u,v) = \sum_{i=1}^{k_2}(b_{u_i} - b_{v_i})^2$, where b_{u_i} is instance u' value on numerical attribute B_i.

The distance of u and v is

$$d(u,v) = VDM(u,v) + ED(u,v)$$

After the distances are computed, we randomly pick one from the k nearest instances of each test instance and use them to form a training subset. This subset is most similar to the test subset. We can use this method to find a desired number of most similar training subsets. The distance between two similar data sets is simply the average distances of each test subset instance with its corresponding nearest training instance. As shown in Figure 1, assume that S_{11}, S_{12}, S_{13}, S_{14} and S_{15} are T_1's 5 most similar training subsets. Their distances to T_1 are computed as d_{11}, d_{12}, d_{13}, d_{14} and d_{15}, respectively.

2.2 Selecting Diverse and Accurate Classifiers

After the most similar training subsets are found, we use the following strategy to select diverse and accurate classifiers from original ensemble. Instead of directly using AUC as the criterion to choose classifiers, we propose a new measure SAUC (Softened Area Under the ROC Curve) as the heuristic.

For a binary classification task, SAUC is defined as

$$SAUC(\gamma) = \frac{\sum_{i=1}^{m} \sum_{j=1}^{n} U(p_i^+ - p_j^-)^\gamma}{mn} \tag{2}$$

$$U(x) = \begin{cases} x \text{ if } x \geq 0 \\ 0 \text{ if } x < 0 \end{cases}$$

where $\gamma \geq 0$, p_i^+, p_j^- represent the predicted probabilities of being positive for the ith positive and the jth negative examples in all m positive examples and n negative examples, respectively.

We choose a series of measures $SAUC(\gamma_1)$, $SAUC(\gamma_2)$, $\cdots$, $SAUC(\gamma_n)$ as heuristics. We use *SAUC*s as heuristics for two reasons. First, SAUC with different powers γ may have different sensitivities and robustness to instance ranking variations. Thus using SAUCs with varied power γ as heuristics can more reliably select diverse classifiers in terms of ranking. Second, SAUC is a softened version of AUC and thus it is basically consistent with AUC. From Equation 2 we can see that $SAUC(0) = AUC$. Thus using SAUCs as criteria can select the classifiers with accurate ranking performance.

As shown in Figure 1 we use each classifier C_t of ensemble E to classify S_{11}, S_{12}, S_{13}, S_{14} and S_{15} to obtain the respective $SAUC(\gamma_1)$ as SA_{11}, SA_{12}, SA_{13}, SA_{14}, SA_{15}. We then compute a score for C_t, which is the weighted average of the $SAUC(\gamma_1)$ values obtained above. It is $S_t = \sum_{i=1}^{5} \frac{SA_{1i}}{d_{1i}}$. We choose the classifier with the highest score. We repeat the above step n times by using a different $SAUC(\gamma_i)$ each time to select a new classifier.

Finally we use all the classifiers selected to construct a new ensemble. This ensemble is responsible for ranking T_1. The new ensemble combination method is weighted averaging, in which a classifier's weight is its score computed above. Using the same method we can construct two other ensembles which are responsible for ranking T_2 and T_3, respectively. We give the pseudo-code of this algorithm in Table 1.

One natural question about the DERC algorithm is that how many new ensembles should be constructed to give the best ranking performance. Since the number of test set partitions equals to the number of new ensembles, this question is equivalent to how to choose an optimal number of test set partitions. Clearly, a small number of partitions generally means large partitioned test subsets, which corresponds to large similar training subsets. Thus the corresponding new ensemble may not specialize on all instances

Table 1. The pseudo code for DERC algorithm

```
DERC$(E,S,T,n)$
    Input:
    $E$ :  An ensemble with classifiers $C_1,\cdots,C_N$
    $S$ :  Training data set
    $T$ :  Test data set
    $n$ :  The number of test set partitions

    choose a classifier from $E$ to classify $T$
    stratify T into $T_1,T_2,\cdots,T_n$
    for each partition $T_i$ do
        $E_i^* \leftarrow \phi$
        find the most similar training subsets $S_{i1},S_{i2},\cdots,S_{ik}$
        compute the distances $d_{i1},d_{i2},\cdots,d_{ik}$ from $T_i$ to $S_{i1},\cdots,S_{ik}$ respectively
        for each measure $SAUC(\gamma_u)$ do
            for each classifier $C_t$ do
                run $C_t$ on $S_{i1},S_{i2},\cdots,S_{ik}$
                obtain the $SAUC(\gamma_u)$ of $C_t$ as $SA_{t_{i1}},\cdots,SA_{t_{ik}}$
                compute the ranking score for classifier $C_t$
                $r_t \leftarrow \sum_{j=1}^{k} \frac{SA_{t_{ij}}}{d_{ij}}$
            endfor
            choose the classifier CC with highest score $r_t$
            $E_i^* \leftarrow E_i^* \cup CC$
        endfor
    endfor
    return all $E_i^*$
```

of the similar training subsets. Therefore our algorithm may not perform best on a small number of partitions. On the contrary for very large number of partitions, the size of similar training subsets will be very small. In this case there is a danger of overfitting. Therefore we can claim that generally too small or too large number of partitions should be avoided. We will perform experiments in the next section to confirm this claim.

3 Experimental Evaluation

To evaluate the performance of our algorithm, we extract 16 representative binary data sets from UCI [11].

We use Bagging and Adaboost as the ensembling methods and Naive Bayes as the base learner. We choose WEKA [12] as the implementations. In order to increase the ensemble diversity, we randomly select half of the training data for each bootstrap in our Bagging process. This can guarantee that the bagging classifiers are diverse to some degree. We compare the performance of DERC with Bagging and Adaboost respectively.

In our DERC algorithm we use $SAUC(\gamma_i)$ as criteria to select classifiers. We have to determine the suitable number and scores of the powers γ_i by taking into account the tradeoff between the quality of results and computational costs. We test the SAUC with a wide ranges of powers γ by using all the 16 datasets in the our experiments. The analysis of these measures' performance shows that the power range of [0,3] is a good choice for SAUC. We choose 9 different SAUC with the powers of 0, 0.1, 0.4, 0.8, 1.0, 1.5, 2, 2.5, 3 in our experiment.

We follow the procedure below to perform our experiment:

1. We discretize the continuous attributes in all data sets using the entropy-based method described in [13].
2. We perform 5-fold cross validation on each data set. In each fold we train an ensemble with 15 classifiers using Bagging and Adaboost methods, respectively. We then run our DERC algorithm on the ensemble trained. By varying the number of test set partitions, we have a number of different DERC algorithm models.
3. We run the second step 20 times and we compute the average AUC for all the predictions.

We use a common statistic to compare the learning algorithms across all data sets. We performed two tailed paired t-test with 95% confidence level to count in how many datasets one algorithm performs significantly better, same, and worse than another algorithm respectively. We use win/draw/loss to represent this.

The experimental results are listed in Table 2 and Table 3.

Table 2. Comparing the predictive AUC of DERC algorithms with Bagging

Dataset	**Bagging**	**DERC(1)**	**DERC(2)**	**DERC(3)**	**DERC(4)**	**DERC(6)**
breast	98.84 ± 0.56	98.84 ± 0.53	98.83 ± 0.50	98.85 ± 0.59	98.86 ± 0.55	98.81 ± 0.59
cars	93.56 ± 3.0	**94.77 ± 2.2**	**94.9 ± 2.7**	**94.83 ± 2.7**	**94.87± 2.9**	**95.02 ± 2.1**
credit	92.89 ± 1.2	**93.43 ± 1.1**	**93.36 ± 1.2**	**93.32 ± 1.2**	**93.3 ± 1.4**	**93.3 ± 1.1**
echocardio	72.34 ± 8.4	72.34 ± 8.4	**74.21 ± 8.3**	**74.11 ± 8.4**	**74.11 ± 8.4**	73.09 ± 8.4
eco	99.28 ±0.84	99.34 ± 1.1	99.34 ± 1.0	99.32 ± 0.84	99.3 ± 1.0	99.33 ± 0.84
heart	85.89 ± 0.45	86.01 ± 0.5	**86.97± 0.5**	**86.81 ± 0.64**	86.06 ± 1.7	85.92 ± 2.6
hepatitis	86.73 ± 2.6	87.06 ± 2.6	87.5 ± 2.9	**89.14 ± 2.6**	**88.59 ± 2.4**	**88.2 ± 1.8**
import	97.75 ± 2.6	97.75 ± 2.6	97.59 ± 2.8	97.72 ± 2.6	97.72 ± 2.6	97.74 ± 2.6
liver	61.77 ± 1.6	61.33 ± 0.45	61.64 ± 0.6	61.4 ± 0.18	61.26 ± 0.3	61.19 ± 3.7
pima	77.27 ± 8.9	**79.33 ± 7.6**	**79.29 ± 7.7**	**79.26 ± 8.0**	**79.14 ± 8.6**	**79.22 ± 8.7**
thyroid	95.12 ± 1.7	95.19 ± 1.6	95.10 ± 1.6	95.16 ± 1.9	**95.24 ± 1.9**	**95.29 ± 1.5**
voting	96.00 ±0.36	96.08 ± 0.36	96.07 ± 0.36	96.27 ± 0.36	95.99 ± 0.36	96.01 ± 0.36
sick	96.84 ±1.56	95.20 ±2.48	•94.27 ±2.11	•94.27±3.47	•93.99±2.79	•94.08±3.02
ionosphere	94.59 ±3.21	94.80 ± 3.22	**95.96 ±3.47**	**95.85**±2.63	**95.84±2.79**	**95.84 ± 3.92**
german	84.26 ±4.02	**87.58 ± 4.33**	**87.40 ± 4.1**	**87.23 ± 4.21**	**87.44 ± 4.2**	**87.4 ± 4.17**
mushroom	99.89 ±0.04	99.79 ± 0.04	99.88 ± 0.04	99.90 ± 0.04	99.89 ± 0.04	99.89 ± 0.04
w/d/l		4/12/0	7/8/1	8/7/1	8/7/1	7/8/1

Table 2 shows the AUC values for the Bagging algorithm and the DERC algorithms with different settings on various data sets. We use DERC(i) to denote the corresponding DERC algorithm which generate a number of *i* new ensembles. Each data cell represents the average AUC value of the 20 trials of 5-fold cross validation for the corresponding algorithm and data set. The data in bold shows the corresponding algorithm performs significantly better than Bagging on the corresponding data set. The data with a "•" means it is significantly worse than that of Bagging.

From this table, we can see that DERC outperforms the original Bagging algorithm. The w/d/l statistics shows that all DERCs with different settings have much more wins than losses compared with Bagging algorithm. If we rank them according to the w/d/l number, we can see that the DERC with 3 or 4 partitions performs best, the DERC with 2 or 6 partitions the second best, while the DERC with 1 partition the worst.

We can also see how the partition numbers influences the dynamic re-construction performance. We can observe that generally the dynamic re-constructions with the partition numbers of 3 or 4 perform best. It shows that dynamic re-construction with intermediate number of partitions outperforms that with large or small number of partitions. This result confirms our discussion in the previous section.

We also compare our DERC algorithm with Adaboost and report the results in Table 3. The similar comparisons show that DERC also significantly outperforms Adaboost in terms of AUC. DERC(3) wins in 5 datasets, ties in 10 datasets on loses only in 1 dataset.

Table 3. Comparing the predictive AUC of DERC algorithms with Adaboost

Dataset	**AdaBoost**	**DERC(1)**	**DERC(2)**	**DERC(3)**	**DERC(4)**	**DERC(6)**
breast	98.99 ± 2.1	98.39± 2.4	98.41 ± 2.1	98.46± 2.1	98.51 ± 2.1	98.53 ± 2.1
cars	91.74 ± 5.0	**93.21± 5.0**	**93.14± 5.0**	**94.72± 5.0**	**93.89± 5.0**	**93.21± 5.0**
credit	92.06 ± 3.7	92.04 ± 3.7	92.06 ± 3.5	92.08± 4.8	92.10 ± 5.3	91.77 ± 4.7
echocardio	72.02 ± 4.8	**73.94 ± 4.8**	**73.94 ± 4.8**	**73.94± 4.8**	**73.94 ± 4.8**	**73.94 ± 4.8**
eco	99.30 ± 1.0	99.13 ± 1.0	99.02 ± 1.0	99.24± 1.0	99.27 ± 1.0	99.62 ± 1.0
heart	88.03 ± 0.28	88.51 ± 0.31	**90.39 ± 0.28**	89.72± 1.22	89.34 ± 1.6	89.58 ± 1.24
hepatitis	85.25 ± 8.6	•83.16 ±5.8	•83.03 ± 5.6	•83.24±8.6	•83.9±8.8	•83.84 ± 5.4
import	98.99 ± 1.7	98.90 ± 0.0	98.98 ± 3.6	98.73± 0.0	98.68 ± 5.2	98.88 ± 0.0
liver	65.45 ± 6.2	66.44 ± 4.1	66.20 ± 5.1	**67.08 ± 5.1**	**67.77 ± 5.1**	66.29 ± 5.1
pima	75.99 ± 8.3	74.92 ± 8.1	74.89 ± 7.2	**77.81± 8.3**	**77.99 ± 6.5**	**78.13± 8.4**
thyroid	95.61 ±0.35	95.55 ± 0.8	95.64 ± 0.27	95.65±0.18	95.58 ± 0.71	95.58 ± 0.35
voting	96.37 ±2.9	96.32 ±2.9	96.39 ± 3.3	96.5 ± 1.4	96.37 ± 1.4	96.37 ± 2.9
sick	97.02 ±1.56	97.08 ± 1.51	97.07 ± 1.43	97.02 ± 1.5	96.99 ± 1.24	97.08 ± 2.54
ionosphere	94.56 ±3.21	94.80 ± 3.47	95.96 ±4.37	95.85±4.26	95.84±3.97	95.84 ± 3.68
german	86.41 ±4.02	**88.24 ± 4.33**	**88.21 ± 4.1**	**88.21±4.21**	**88.19±4.2**	**88.19± 4.17**
mushroom	99.92 ±0.04	99.79 ± 0.04	99.88 ± 0.04	99.90 ± 0.04	99.89 ± 0.04	99.89 ± 0.04
w/d/l		3/12/1	4/11/1	5/10/1	5/10/1	4/11/1

4 Conclusions and Future Work

In this paper we propose a novel dynamic re-construction technique which aims to improve the ranking performance of any given ensemble. This is a generic technique which can be applied on any existing ensembles. The advantage is that it is independent of the specific ensemble construction method. The empirical experiments show that this dynamic re-construction technique can achieve significant performance improvement in term of ranking over the original Bagging and Adaboost ensembles, especially with an intermediate number of partitions .

In our current study we use Naive Bayes as the base learner. For our future work, we plan to investigate how other learning algorithms perform with the DERC technique. We also plan to explore whether DERC is also effective when it is applied on other ensemble methods.

References

1. Bradley, A.P.: The use of the area under the ROC curve in the evaluation of machine learning algorithms. Pattern Recognition **30** (1997) 1145–1159
2. Provost, F., Fawcett, T.: Analysis and visualization of classifier performance: comparison under imprecise class and cost distribution. In: Proceedings of the Third International Conference on Knowledge Discovery and Data Mining. AAAI Press (1997) 43–48
3. Hand, D.J., Till, R.J.: A simple generalisation of the area under the ROC curve for multiple class classification problems. Machine Learning **45** (2001) 171–186
4. Bauer, E., Kohavi, R.: An empirical comparison of voting classification algorithms: Bagging, boosting and variants. Machine Learning **36** (1999) 105–139
5. Breiman, L.: Bagging predictors. Machine Learning **24** (1996) 123–140
6. Quinlan, J.R.: Bagging, boosting, and C4.5. In: Proceedings of the Thirteenth National Conference on Artificial Intelligence. (1996) 725 – 730
7. Freund, Y., Schapire, R.E.: Experiments with a new boosting algorithm. In: Proceedings of the Thirteenth International Conference on Machine Learning. (1996) 148 – 156
8. Caruana, R., Niculescu-Mizil, A.: An empirical evaluation of supervised learning for ROC area. In: The First Workshop on ROC Analysis in AI. (2004)
9. Freund, Y., Iyer, R., Schapire, R., Singer, Y.: An efficient boosting algorithm for combining preferences. Journal of Machine Learning Research **4** (2003) 933–969
10. Stanfill, C., Waltz, D.: Toward memory-based reasoning. Communications of the ACM **29** (1986) 1213–1228
11. Blake, C., Merz, C.: UCI repository of machine learning databases. http://www.ics.uci.edu/∼mlearn/MLRepository.html (1998) University of California, Irvine, Dept. of Information and Computer Sciences.
12. Witten, I.H., Frank, E.: Data Mining: Practical machine learning tools with Java implementations. Morgan Kaufmann, San Francisco (2000)
13. Fayyad, U., Irani, K.: Multi-interval discretization of continuous-valued attributes for classification learning. In: Proceedings of Thirteenth International Joint Conference on Artificial Intelligence. Morgan Kaufmann (1993) 1022–1027