Towards Maximizing the Area Under the ROC Curve for Multi-Class Classification Problems

Ke Tang[*1] **Rui Wang**[1] **Tianshi Chen**[2]
[1] Nature Inspired Computation and Applications Laboratory (NICAL), School of Computer Science and Technology, University of Science and Technology of China, Hefei 230027, China
[2] Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190, China
ketang@ustc.edu.cn wrui1108@mail.ustc.edu.cn chentianshi@ict.ac.cn

Abstract

The Area Under the ROC Curve (AUC) metric has achieved a big success in binary classification problems since they measure the performance of classifiers without making any specific assumptions about the class distribution and misclassification costs. This is desirable because the class distribution and misclassification costs may be unknown during training process or even change in environment. MAUC, the extension of AUC to multi-class problems, has also attracted a lot of attention. However, despite the emergence of approaches for training classifiers with large AUC, little has been done for MAUC. This paper analyzes MAUC in-depth, and reveals that the maximization of MAUC can be achieved by decomposing the multi-class problem into a number of independent sub-problems. These sub-problems are formulated in the form of a "learning to rank" problem, for which well-established methods already exist. Based on the analysis, a method that employs RankBoost algorithm as the sub-problem solver is proposed to achieve classification systems with maximum MAUC. Empirical studies have shown the advantages of the proposed method over other eight relevant methods. Due to the importance of MAUC to multi-class cost-sensitive learning and class imbalanced learning problems, the proposed method is a general technique for both problems. It can also be generalized to accommodate other learning algorithms as the sub-problem solvers.

Introduction

The performance of classification systems are traditionally measured by accuracy, which is the ratio of correctly labeled instances to all testing instances. However, using accuracy implicitly presumes that the class distribution of the data set is approximately balanced and the misclassification costs are equal, while these conditions rarely hold in practices. That is, for many real-world problems, the class distribution is imbalanced (He and Garcia 2009) and different types of misclassifications usually lead to unequal costs (Elkan 2001). Moreover, it is also likely that the misclassification costs is unknown during the training process or both the class distribution and misclassification costs can change in environment (Provost and Fawcett 2001).

In all the above-mentioned circumstances, seeking a system with the maximum classification accuracy is inappropriate (Fawcett 2006), and thus a number of new learning targets have emerged during the past ten years. For problems involving unequal misclassification costs (i.e., the cost-sensitive learning problem), the natural learning target is minimizing the total classification cost for a given misclassification cost matrix. Since traditional classification methods tend to perform poorly in cost-sensitive problems, new methods taken the cost information into consideration during classifier construction have been developed (Elkan 2001). When the class distribution is imbalanced (i.e., the class imbalanced learning problem), the main difficulty is that traditional classifiers are easily overwhelmed by instances from majority classes while the minority classes instances are usually ignored (He and Garcia 2009). There are generally two different learning targets for imbalanced learning problem. The first is, setting aside the misclassification costs, obtain enhanced recognition rate of minority classes instances while the error rate of majority classes are not rise or rise in accepted range (Kubat and Holte 1998). However, in most imbalanced learning problems, misclassifying a minority class instance is more expensive than misclassifying a majority class instance. It is unrealistic to ignore the misclassification cost completely. In other words, the imbalanced learning problem and the cost-sensitive learning problem may happen simultaneously. More important, it is also very hard if not impossible to determine numerically the misclassification costs during training process (Maloof 2003). Therefore, the second learning target of imbalanced learning problem is to obtain classification system which could, in general, work well for all possible class distribution and misclassification costs. This issue was successfully addressed in binary problems using ROC analysis and the Area Under the ROC Curve (AUC) metric (Fawcett 2006).

The properties and merits of ROC curve and AUC are widely known to the machine learning and data mining com-

[*]This work was supported by the National Natural Science Foundation of China under Grant 60802036.

munities (Fawcett 2006; Huang and Ling 2005). Generally speaking, a ROC curve explicitly shows the tradeoff between the true positive rate and the false positive rate of a binary classification system on different operating points, putting the class distribution and the misclassification costs out of the evaluation of classifiers' performance. Once the class distribution and misclassification cost matrix are given, a categorical classifier with the least total cost can be determined by threshold moving (Lachiche and Flach 2003). However, it is commonly the case that the ROC curves of two classifiers cross over at some point in the ROC space, and thus ROC curve is not a suitable approach for identifying the best classifier in practice. Alternatively, a standard metric used by researchers to indicate the quality of ROC curve is AUC (Fawcett 2006). Larger AUC corresponds to good ROC curve and implies, in general, the classifier can achieve smaller total cost by moving the threshold given a specific class distribution and misclassification cost matrix.

The extension of ROC curve to multi-class problem is the ROC hyper-surface, which inherits all the desirable properties of ROC curve. For example, it has been shown that a classifier with good ROC hyper-surface can lead to classifiers suitable for various class distribution and misclassification costs via simple re-optimization the its output matrix (Srinivasan 1999; Bourke et al. 2008). However, it is, again, unlikely that the ROC hyper-surface of a classifier covers that of another classifier entirely. Furthermore, due to the increase of the dimensionality of the ROC space, achieving the optimal ROC hyper-surface is even more difficult than achieving the optimal ROC curve. Hence, researchers have extended the AUC to multi-class problem to establish tractable learning targets and to facilitate the comparison between classifiers. A straightforward generalization of AUC is the Volume Under the ROC hyper-Surface (VUS) (Ferri, Hernández-orallo, and Salido 2003), but the computation of VUS is complicated and elusive (Landgrebe and Duin 2008). Moreover, (Edwards, Metz, and Nishikawa 2005) has pointed it out theoretically that the VUS value of a "near guessing" classifier will get close to a "near perfect" classifier when the number of class $c > 2$. Therefore, a simpler generalization of AUC for multi-class problems, namely MAUC (Hand 2001), has been much more widely used in recent work (Zhou and Liu 2006; Cerquides and Mantaras 2005; Sikonja 2004).

During the past decade, a number approaches have been developed to directly train classifiers with large AUC (Ferri, Flach, and Hernandez-Orallo 2002; Herschtal and Raskutti 2004; Yan et al. 2003). However, few of them can be easily extended to multi-class problem. In fact, research on related topics such as imbalanced learning problems is also highly focused on binary class problem, while progress on multi-class problems is limited (He and Garcia 2009). For those related work on multi-class problems, MAUC was merely employed as the core indicator (sometimes the only indicator) for evaluating the performance of a classifier, but not as the learning target. To the best of our knowledge, the only work that explicitly aims to seek classifiers with large MAUC appeared very recently, namely Evolutionary AUC Maximization (EAM) (Lu, Ke, and Yao 2010). EAM employs an evolutionary algorithm to search for the optimal weights for a neural network, so that the MAUC of the neural network is maximized. The computational cost of EAM is reported to be high while the performance in terms of MAUC is mixed, as will be shown by our experimental study. In this work, we explore how to obtain maximum MAUC and propose a method named MAUC Decomposition based Classification method (MDC). Briefly speaking, our analysis of MAUC revealed that the maximization of MAUC can be decomposed in to a batch of independent sub-problems. Each sub-problem can be formulated in such a way that an existing technique, the RankBoost algorithm (Freund et al. 2003), can solve it nicely. Hence, the MDC method first decomposes the MAUC into sub-problems. Then the RankBoost algorithm is employed to solve each sub-problem independently. Finally, classifiers obtained on all sub-problems are combined together to form the final classification system. Extensive empirical studies have been carried out to compare MDC with other 8 relevant methods, and the results clearly demonstrate the advantages of MDC.

The rest of the paper starts with the analysis of MAUC. Then, the MDC method is described in detail in Section 3. In Section 4, experimental studies are presented, followed by the conclusions in Section 5.

Analysis of MAUC

MAUC: A Multi-class Extension of AUC

Given a set of instances $S = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$, where $x_i \in \mathbb{R}^m$ is the feature vector of the i-th instance, and $y_i \in \{0, 1\}$ is the label of the i-th instance. A classifier $H(x_i) \rightarrow \mathbb{R}$ output a numerical score[1] that indicates the confidence of x_i belonging to class 0. The AUC of this classifier can be calculated by,

$$AUC = \frac{\sum_{x_i \in class(0);\, x_j \in class(1)} s(x_i, x_j)}{n^0 \times n^1} \tag{1}$$

where n^0 and n^1 denote the number of instances in class 0 and class 1 respectively, and $s(x_i, x_j)$ is defined as:

$$s(x_i, x_j) = \begin{cases} 1, & \text{if } H(x_i) > H(x_j); \\ 0.5, & \text{if } H(x_i) = H(x_j); \\ 0, & \text{if } H(x_i) < H(x_j). \end{cases} \tag{2}$$

For multi-class problems (i.e., $y_i \in \{1, \ldots, c\}$), a classifier will provide a $n \times c$ matrix M. The element m_{ij} indicates the confidence that the i-th instance in S belongs to class j. The MAUC is defined as,

$$MAUC = \frac{2}{c \times (c-1)} \sum_{i<j} \frac{A_{ij} + A_{ji}}{2} \tag{3}$$

where A_{ij} is the AUC between class *i* and class *j* calculated from the *i*-th column of M. Note that for multi-class problems, A_{ij} may not equal to A_{ji}, and thus both of them need to be involved in the calculation of MAUC (Hand 2001).

[1]Although some types of classifiers were originally designed to output the class label directly, most of them have been extended to produce numerical scores. One typical example is the probabilistic decision tree (Provost and Domingos 2003).

Decomposition of MAUC

Suppose that a multi-class problem is decomposed into $\frac{c\times(c-1)}{2}$ binary sub-problems in the one-versus-one manner. Each sub-problem involves two of the c classes. It can be observed from Eq. (3) that MAUC is actually the average of the AUC values of all the binary sub-problems. Each AUC value in this case is calculated twice for corresponding sub-problem, based on two columns of the matrix M. Hence, maximizing MAUC involves adjusting the columns of M inter-dependently, which is very difficult.

Fortunately, the MAUC can also be analyzed in a different way by re-writing Eq. (3) as Eq. (4),

$$MAUC = \frac{1}{c}\sum_{i=1}^{c}\frac{1}{c-1}\sum_{j=1;j\neq i}^{c} A_{ij} \tag{4}$$

From Eq. (4), it can be observed that MAUC can be regarded as the average of Eq. (5) over all columns of M.

$$A_i = \frac{1}{c-1}\sum_{j=1;j\neq i}^{c} A_{ij}. \tag{5}$$

Hence, MAUC can be maximized as long as each A_i $(1 \leq i \leq c)$ in Eq. (5) is maximized. Note that each A_i is calculated solely based on the i-th column of M. Thus, the optimization of different A_i is not necessarily inter-dependent and can be done separately. Having this observation in mind, we may obtain another implication for building classification systems with maximum MAUC. To be specific, many classification techniques, such as neural network, Naive Bayes and decision tree, are capable of providing the matrix M in a single run. This is a desirable property in many cases. However, in the context of MAUC, training classifiers in this way means optimizing all A_i as a whole (i.e., make the optimization of A_i dependent with one another), and increase the difficulty of obtaining a good classifier. Instead, it may be much easier to train c classifiers (following the formulation given in Eqs. (4) and (5)), each of which only aims to achieve one maximum A_i. The ensemble of these classifiers will serve as the final classification system that provides a desirable MAUC value.

According to the above analysis, we propose to seek a classification system with large MAUC by solving for each class i the following sub-problem:

$$\begin{aligned}\max A_i &= \frac{1}{c-1}\sum_{j=1;j\neq i}^{c} A_{ij}\\ &=\frac{1}{c-1}\sum_{j=1;j\neq i}^{c}\left(\sum_{\substack{x\in class(i);\\ x'\in class(j)}}\frac{s(x,x')}{n^i\times n^j}\right)\\ &=\frac{1}{c-1}\times\frac{1}{n^i}\sum_{j=1;j\neq i}^{c}\left(\frac{1}{n^j}\sum_{\substack{x\in class(i);\\ x'\in class(j)}} s(x,x')\right)\end{aligned} \tag{6}$$

where n^i and n^j are number of instances in class *i* and class *j* respectively, $s(x,x')$ is defined the same as Eq. (2).

The MDC Method

For the sub-problem defined by Eq. (6), a classifier needs to be trained so that the instances from class i is ranked above instances from the other classes. If an instance x from class i is ranked above (i.e., having larger score than) an instance from class j, the classifier will get a reward of $\frac{1}{(c-1)\times n^i\times n^j}$. Assuming that the scores of two different instances are also different, Eq. (6) can be further presented in the following form:

$$\max\sum_{x;x'} D(x,x')[[H(x) > H(x')]] \tag{7}$$

x is an instance in class i, x' is an instance in class j $(j \neq i)$. $[[H(x) > H(x')]]$ equals to 1 (0) when $H(x) > H(x')$ is true (false). $D(x,x')$ is the reward for the two instances being ranked correctly, which is

$$D(x,x') = \frac{1}{(c-1)\times n^i\times n^j} \tag{8}$$

By summing up the $D(x,x')$ of all pairs of x and x', we have

$$\sum_{x;x'} D(x,x') = 1 \tag{9}$$

Eq. (7) is in the form of a typical "learning to rank" problem. Moreover, with the additional condition given in Eq. (8), the problem is actually the "bipartite feedback" problem investigated in a previous work (Freund et al. 2003), and can be nicely addressed by the RankBoost algorithm proposed therein. RankBoost is a boosting type algorithm which works by combining a batch of weak learners. In each iteration, a weak classifier is trained. Over the iterations, the focus will be concentrated on the pairs of instances that are difficult to rank correctly. Specifically, the RankBoost.B algorithm (Freund et al. 2003) fit our problem perfectly. Therefore, it is directly employed in this work as the sub-problem solver and decision stump is employed as the weak learner. The only modification we made to the RankBoost.B algorithm is the initial weights assigned to each instance, which is determined using Eq. (8). Readers are referred to the original publication of the RankBoost algorithm for its implementation details.

By now, all details of the proposed MDC method have been presented. To summarize, MDC first decompose a multi-class problem into c sub-problems. Then, RankBoost.B algorithm is applied to solve the sub-problems separately. After that, the final classification system is composed of the classifiers trained for all sub-problems. The pseudocode of the MDC method is presented in Algorithm 1.

Experiments

Data Sets

16 data sets were used in our experiments, 14 of them (except phoneme data set and washington data set) are from UCI Machine Learning Repository (Asuncion and Newman 2007), washington data set is hyperspectral imaginary data set (Neher and Srivastava 2005), and phoneme (Hastie, Tibshirani, and Friedman 2001) data set is from handwritten digits recognition problem. Information of these data sets are summarized in Table 1.

Algorithm 1 Pseudo-code of MDC

```
Input: data set consist of $c$ classes
for $i = 1$ to $c$ do
    $v_1(x) = \begin{cases} \frac{1}{(c-1)\times n^i}, & \text{if } x \in class(i); \\ \frac{1}{n^j}, & \text{if } x \in class(j); j \neq i. \end{cases}$
    for $t = 1$ to $T$ do
        train weak learner $h_t$ using distribution $D(x, x') = v_t(x) \times v_t(x')$
        choose $\alpha_t \in \mathbb{R}$
        update:
        $v_{t+1}(x) = \begin{cases} \frac{v_t(x)\exp(-\alpha_t h_t(x))}{Z_t}, & \text{if } x \in class(i); \\ \frac{v_t(x)\exp(\alpha_t h_t(x))}{Z_t'}, & \text{if } x \in class(j); j \neq i. \end{cases}$
        where:
        $Z_t = \sum_{x \in class(i)} v_t(x)\exp(-\alpha_t h_t(x))$
        $Z_t^{'} = \sum_{x \notin class(i)} v_t(x)\exp(\alpha_t h_t(x))$
    end for
    $H_i(x) = \sum_{t=1}^{T} \alpha_t h_t(x)$
end for
$H = \{H_1, H_2, \ldots, H_c\}$
```

Experimental setup

We compared MDC with other 8 classification methods: Rescale (Zhou and Liu 2006), Multi-class RankBoost, EAM (Lu, Ke, and Yao 2010) , Naive Bayes, 1-Nearest Neighbor (1NN), C4.5, Artificial Neural Network (ANN), and Support Vector Machines (SVM) with RBF kernel. Rescale is an instances weighting method (using C4.5 as the base classifier) which was designed for multi-class cost-sensitive learning and imbalanced learning. It has been reported to be capable of achieving large MAUC on multi-class problems. Since MDC employs RankBoost as the sub-problem solver, it is interesting to examine whether RankBoost can be directly extended to maximize MAUC. Hence, we modified the RankBoost, which was originally designed for binary class problems, to a multi-class version and obtained the Multi-class RankBoost algorithm (we call it M-RankBoost hereafter). The extension is implemented by decomposing the multi-class problem into a number of binary class problems in one-versus-all manner. As mentioned before, EAM might be the first attempt that explicitly seeks classifiers with large MAUC, thus it is included into the experiments. The remaining 5 methods are all mainstream machine learning techniques. They not only provide a baseline for the empirical studies, but may also tell whether designing a specific approach with respect to MAUC is worthwhile.

All the compared methods were implemented on WEKA (Witten and Frank 2000) platform. The number of weak learners was set to 50 for both MDC and M-RankBoost. The parameter c and γ of SVM were set to be the values which maximize the average MAUC in a 3-fold cross-validation on the training data set of a 5-fold cross-validation. The search range of c is $2^{-5}, 2^{-3}, \ldots, 2^{15}$, and the search range of γ is $2^{-15}, 2^{-13}, \ldots, 2^{3}$. The generation number of EAM was set to 500, and all the other parameters were set according to the original publication. For each data set, 20 times 5-fold cross-validation were performed. The average MAUC of each method on every data set is reported in Table 2. To check whether the differences between MDC and other methods are significant, Wilcoxon signed rank statistical test with 95% confidence level has been conducted. The results of these statistical tests is also given in Table 2. Besides, the best average MAUC value on each data set was highlighted in boldface.

Table 1: Summary of the data sets used in the experiments

Data set	No. of features	No. of instances	No. of classes
abalone	8	4139	18
arrhythmia	279	416	7
contraceptive	9	1473	3
glass	9	192	4
hayes-roth	3	160	3
isolet	617	7797	26
mfeat	649	2000	10
page-blocks	10	5473	5
phoneme	256	4509	5
shuttle	9	57977	5
splice	60	3190	3
thyroid-allhypo	28	3770	3
thyroid-allrep	28	3772	4
washington	210	11200	7
waveform40	40	5000	3
yeast	8	1479	9

Results

From Table 2, it can be observed that MDC achieved the best results on 12 out of 16 data sets. The advantage of MDC over all the other 8 compared methods is statistically significant on 8 data sets. Concretely, MDC performed significantly worse than ANN on the abalone, isolet and waveform40 data sets, and was outperformed by EAM on the waveform40 data set. Besides, MDC is significantly better than Rescale, 1NN, Naive Bayes, C4.5 and SVM on almost all data sets, with merely two exceptions for comparisons with Rescale and C4.5 (where the difference of MDC and the compared method is insignificant), respectively. M-RankBoost was significantly outperformed by MDC on 11 data sets. On the remaining 5 data sets, difference between the two methods is insignificant. To summarize, MDC overall performed the best among all the 9 methods involved in the experiments. The superiority of MDC to M-RankBoost showed that the proposed MAUC decomposition strategy is crucial to the appealing performance of MDC, since it is the only difference between MDC and M-RankBoost. As a recently proposed method, EAM did not show good performance in the experiment. One possible reason might be that the generation number was set too small to make EAM converge to good solutions. However, as can be found in Table 3, the runtime of EAM is the longest, and is at least ten times longer than that of MDC in the current setting. Hence, we feel it impractical to evaluate EAM with a larger generation

Table 2: The MAUC achieved by the 9 compared methods. The MAUC value of each method is averaged over 20 independent runs of 5-fold cross-validation. For each data set, Wilcoxon signed-rank test with 95% confidence level is employed to compare the 8 other methods with MDC. The methods that performed significantly worse (better) than MDC is highlighted with $\dagger$($\ddagger$). The largest MAUC on each data set is in boldface.

	MDC	Rescale	M-RankBoost	EAM	Naive Bayes	1NN	C4.5	SVM	ANN
abalone	0.7847	$0.7361^\dagger$	$0.7773^\dagger$	$0.7491^\dagger$	$0.7564^\dagger$	$0.5486^\dagger$	$0.5972^\dagger$	$0.5632^\dagger$	**0.7978**‡
arrhythmia	**0.8623**	$0.7539^\dagger$	$0.8541^\dagger$	$0.7262^\dagger$	$0.8144^\dagger$	$0.6452^\dagger$	$0.7675^\dagger$	$0.7465^\dagger$	$0.8404^\dagger$
contraceptive	**0.7311**	$0.6434^\dagger$	$0.7280^\dagger$	$0.7090^\dagger$	$0.6876^\dagger$	$0.5603^\dagger$	$0.6597^\dagger$	$0.6427^\dagger$	$0.7149^\dagger$
glass	**0.8868**	$0.7837^\dagger$	0.8853	$0.8157^\dagger$	$0.8130^\dagger$	$0.7659^\dagger$	$0.7924^\dagger$	$0.7491^\dagger$	$0.8344^\dagger$
hayes-roth	**0.9564**	0.9515	$0.9391^\dagger$	$0.8633^\dagger$	$0.9059^\dagger$	$0.9077^\dagger$	0.9494	$0.8979^\dagger$	$0.8317^\dagger$
isolet	0.9955	$0.9265^\dagger$	0.9955	$0.8802^\dagger$	$0.9840^\dagger$	$0.9422^\dagger$	$0.9264^\dagger$	$0.9858^\dagger$	**0.9983**‡
mfeat	**0.9979**	$0.9710^\dagger$	**0.9979**	$0.8430^\dagger$	$0.9904^\dagger$	$0.9885^\dagger$	$0.9710^\dagger$	$0.9844^\dagger$	0.9978
page-blocks	**0.9869**	$0.9343^\dagger$	$0.9758^\dagger$	$0.9458^\dagger$	$0.9503^\dagger$	$0.8490^\dagger$	$0.9386^\dagger$	$0.7884^\dagger$	$0.9161^\dagger$
phoneme	**0.9853**	$0.9333^\dagger$	$0.9852^\dagger$	$0.9294^\dagger$	$0.9737^\dagger$	$0.9206^\dagger$	$0.9264^\dagger$	$0.9510^\dagger$	$0.9838^\dagger$
shuttle	**0.9998**	$0.9998^\dagger$	$0.9995^\dagger$	$0.9250^\dagger$	$0.9655^\dagger$	$0.9830^\dagger$	$0.9958^\dagger$	$0.9883^\dagger$	$0.9011^\dagger$
splice	**0.9869**	$0.9561^\dagger$	$0.9866^\dagger$	$0.8930^\dagger$	$0.9749^\dagger$	$0.7509^\dagger$	$0.9410^\dagger$	$0.8940^\dagger$	$0.9550^\dagger$
thyroid-allhypo	**0.9983**	$0.9905^\dagger$	$0.9971^\dagger$	$0.9629^\dagger$	$0.9274^\dagger$	$0.7003^\dagger$	$0.9893^\dagger$	$0.8826^\dagger$	$0.9094^\dagger$
thyroid-allrep	**0.9622**	$0.9382^\dagger$	$0.9563^\dagger$	$0.8309^\dagger$	$0.9126^\dagger$	$0.6612^\dagger$	$0.9182^\dagger$	$0.8214^\dagger$	$0.9053^\dagger$
washington	**0.9855**	$0.9314^\dagger$	**0.9855**	$0.9070^\dagger$	$0.9662^\dagger$	$0.8857^\dagger$	$0.9314^\dagger$	$0.9452^\dagger$	$0.9011^\dagger$
waveform40	0.9583	$0.8283^\dagger$	0.9583	**0.9669**‡	$0.9562^\dagger$	$0.8003^\dagger$	$0.8299^\dagger$	$0.9006^\dagger$	$0.9625^\ddagger$
yeast	**0.8547**	$0.7492^\dagger$	$0.8340^\dagger$	0.8523	0.8539	$0.7064^\dagger$	$0.7420^\dagger$	$0.7280^\dagger$	$0.8229^\dagger$

number.

Table 3 summarizes the runtime of the compared methods. It can be observed that MDC is computationally less expensive than ANN and EAM. The runtime of it is comparable to that of M-RankBoost and SVM , while is much longer than that of Rescale, Naive Bayes, 1NN and C4.5.

Conclusions

For multi-class classification problem, a classifier with large MAUC can be easily adapted to various scenarios in which the class distribution is imbalanced and/or misclassification costs is unequal, and to the situations that the class distribution and misclassification costs change over time. Therefore, MAUC is an important intermediate learning target for tackling real-world problems in uncertain environments. In spite of this, few techniques have been developed for maximizing the MAUC of a classification system. This paper analyzes the MAUC and shows that MAUC can be decomposed to facilitate the maximization of MAUC. Based on the analysis, a novel method is proposed to build classification systems with maximum MAUC. It is shown that the proposed MDC method outperformed a number of traditional mainstream machine learning techniques as well as recent methods that were claimed to favor MAUC. Since large MAUC is desirable in the context of class imbalanced learning and cost-sensitive learning, the proposed method is a general technique that is applicable to both types of problems. Furthermore, the analysis of MAUC is applicable for any type of learning algorithms that are capable of handling the subproblem defined in Eq. (5). Therefore, although RankBoost algorithm is adopted in this work, the MDC method can easily accommodate other learning algorithms as well.

For the above-mentioned uncertain environments, it has been shown that the integration of multiple ROC curves (i.e., multiple classifiers) can be more robust than using a single ROC curve (Provost and Fawcett 2001). To achieve this, a technique is required to be capable of exploring classifiers that perform well in different region of the ROC space, as done in a recent work (Fawcett 2008). However, such work is still restricted to binary class problems, while extension to multi-class problems is never a trivial task. For multi-class problems, one potential approach for generating a good ensemble of classifiers is to seek classifiers with large MAUC, while exhibit diverse characteristics in the ROC space. The method proposed in this paper plays an important role in such an approach, and the remaining issues will be investigated in the future.

References

Asuncion, A., and Newman, D. 2007. UCI machine learning repository. http://www.ics.uci.edu/∼mlearn/MLRepository.html.

Bourke, C.; Deng, K.; Scott, S. D.; and Vinodchandran, R. E. S. N. V. 2008. On reoptimizing multi-class classifiers. *Machine Learning* 71(2-3):219–242.

Cerquides, J., and Mantaras, R. L. D. 2005. Robust Bayesian linear classifier ensembles. In *Proceedings of 16th European Conference on Machine Learning*, 72–83.

Edwards, D. C.; Metz, C. E.; and Nishikawa, R. M. 2005. The hypervolume under the ROC hypersurface of ”near-guessing” and ”near-perfect” observers in N-class classification tasks. *IEEE transactions on medical imaging* 24(3):293–239.

Elkan, C. 2001. The Foundations of Cost-Sensitive Learning. In *Proceedings of 17th International Joint Conference on Artificial Intelligence*, 973–978.

Fawcett, T. 2006. An introduction to ROC analysis. *Pattern Recognition Letters* 27:861–874.

Table 3: Average runtime (in seconds) of the compared methods on the 16 data sets.

	MDC	Rescale	M-RankBoost	EAM	Naive Bayes	1NN	C4.5	SVM	ANN
abalone	4.437	0.147	4.419	3262.416	0.094	1.194	0.556	191.044	45.166
arrhythmia	3.775	0.394	3.816	533.641	0.103	0.369	0.438	0.819	396.262
contraceptive	0.206	0.084	0.200	714.859	0.016	0.159	0.075	1.181	3.772
glass	0.050	0.013	0.050	112.609	0.006	0.003	0.006	0.047	0.550
hayes-roth	0.013	0.009	0.012	119.142	0.006	0.000	0.003	0.016	0.184
isolet	1052.337	68.463	1029.863	18638.360	13.216	288.525	68.856	99.416	35685.656
mfeat	79.662	4.522	79.497	3651.115	1.672	20.347	4.619	9.762	9564.391
page-blocks	2.100	0.313	2.078	3811.868	0.069	2.784	0.294	3.331	19.847
phoneme	48.950	6.681	50.037	3511.208	1.134	42.078	7.209	10.931	3321.409
shuttle	20.781	4.691	21.234	149415.502	0.812	311.475	3.591	15.934	196.512
splice	2.659	0.513	2.644	1667.820	0.138	5.722	0.541	10.209	142.394
thyroid-allhypo	1.469	0.094	1.456	1540.512	0.094	3.559	0.075	2.500	43.841
thyroid-allrep	1.922	0.156	1.928	3007.865	0.103	3.400	0.109	2.559	48.234
washington	142.562	14.631	148.444	10061.219	2.834	215.253	15.281	85.600	5601.959
waveform40	4.453	1.131	4.437	1528.848	0.162	9.469	1.150	7.794	111.141
yeast	0.609	0.078	0.603	1122.179	0.028	0.147	0.081	0.456	7.012

Fawcett, T. 2008. PRIE: a system for generating rulelists to maximize ROC performance. *Data Mining and Knowledge Discovery* 17:207–224.

Ferri, C.; Flach, P.; and Hernandez-Orallo, J. 2002. Learning decision trees using the area under the ROC curve. In *Proceedings of 19th International Conference on Machine Learning*, 139–146.

Ferri, C.; Hernández-orallo, J.; and Salido, M. A. 2003. Volume under the ROC Surface for Multi-class Problems. In *Proceedings of 14th European Conference on Machine Learning*, 108–120.

Freund, Y.; Iyer, R.; Schapire, R. E.; and Singer, Y. 2003. An efficient boosting algorithm for combining preferences. *Journal of Machine Learning Research* 4(6):933–969.

Hand, D. J. 2001. A simple generalisation of the area under the ROC curve for multiple class classification problems. *Machine Learning* 45:171–186.

Hastie, T.; Tibshirani, R.; and Friedman, J. 2001. *The Elements of Statistical Learning, Springer series in statistics*. New York: Springer.

He, H. B., and Garcia, E. A. 2009. Learning from imbalanced data. *IEEE Transactions on Knowledge and Data Engineering* 21(9):1263–1284.

Herschtal, A., and Raskutti, B. 2004. Optimising area under the ROC curve using gradient descent. In *Proceedings of 21st International Conference on Machine Learning*. New York, New York, USA: ACM Press.

Huang, J., and Ling, C. X. 2005. Using AUC and accuracy in evaluating learning algorithms. *IEEE Transactions on Knowledge and Data Engineering* 17(3):299–310.

Kubat, M., and Holte, R. 1998. Machine learning for the detection of oil spills in satellite radar images. *Machine Learning* 30(2-3):195–215.

Lachiche, N., and Flach, P. 2003. Improving accuracy and cost of two-class and multi-class probabilistic classifiers using ROC curves. In *Proceedings of the 20th International Conference on Machine Learning*, 416–423.

Landgrebe, T. C. W., and Duin, R. P. W. 2008. Efficient multiclass ROC approximation by decomposition via confusion matrix perturbation analysis. *IEEE transactions on pattern analysis and machine intelligence* 30(5):810–22.

Lu, X. F.; Ke, T.; and Yao, X. 2010. Evolving Neural Networks with Maximum AUC for Imbalanced Data Classification. In *The 5th International Conference on Hybrid Artificial Intelligence Systems (HAIS2010)*, 335–342.

Maloof, M. A. 2003. Learning when data sets are imbalanced and when costs are unequal and unknown. In *Working Notes of the ICML'03 Workshop on Learning from Imbalanced Data Sets*.

Neher, R., and Srivastava, a. 2005. A Bayesian MRF framework for labeling terrain using hyperspectral imaging. *IEEE Transactions on Geoscience and Remote Sensing* 43(6):1363–1374.

Provost, F., and Domingos, P. 2003. Tree induction for probability-based ranking. *Machine Learning* 5:199–215.

Provost, F., and Fawcett, T. 2001. Robust classification for imprecise environments. *Machine Learning* 42:203–231.

Sikonja, M. R. 2004. Improving Random Forests. In *Proceedings of 15th European Conference on Machine Learning*, 359–370.

Srinivasan, A. 1999. Note on the location of optimal classifiers in n-dimensional ROC space. Technical Report PRG-TR-2-99, Oxford University.

Witten, I., and Frank, E. 2000. *Data mining practical machine learning tools and techniques with JAVA implementations*. Morgan Kaufmann Publishers.

Yan, L.; Dodier, R.; Mozer, M. C.; and Wolniewicz, R. 2003. Optimizing Classifier Performance via an Approximation to the Wilcoxon-Mann-Whitney Statistic. In *Proceedings of the 20th International Conference on Machine Learning*, 848–855.

Zhou, Z. H., and Liu, X. Y. 2006. On Multi-Class Cost-Sensitive Learning. In *Proceedings of the 21st National Conference on Artificial Intelligence*, 567–572.