Building Decision Trees for the Multi-class Imbalance Problem

T. Ryan Hoens[1], Qi Qian[2], Nitesh V. Chawla[1], and Zhi-Hua Zhou[2]

[1]Department of Computer Science and Engineering
University of Notre Dame, Notre Dame IN 46556, USA
{thoens,nchawla}@cse.nd.edu
[2]National Key Laboratory for Novel Software Technology
Nanjing University, Nanjing 210046, China
{qianq,zhouzh}@lamda.nju.edu.cn

Abstract. Learning in imbalanced datasets is a pervasive problem prevalent in a wide variety of real-world applications. In imbalanced datasets, the class of interest is generally a small fraction of the total instances, but misclassification of such instances is often expensive. While there is a significant body of research on the class imbalance problem for binary class datasets, multi-class datasets have received considerably less attention. This is partially due to the fact that the multi-class imbalance problem is often much harder than its related binary class problem, as the relative frequency and cost of each of the classes can vary widely from dataset to dataset. In this paper we study the multi-class imbalance problem as it relates to decision trees (specifically C4.4 and HDDT), and develop a new multi-class splitting criterion. From our experiments we show that multi-class Hellinger distance decision trees, when combined with decomposition techniques, outperform C4.4.

1 Introduction

One of the fundamental problems in data mining classification problems is that of class imbalance. In the typical binary class imbalance problem one class (negative class) vastly outnumbers the other (positive class). The difficulty of learning under such conditions lies in the induction bias of most learning algorithms. That is, most learning algorithms, when presented with a dataset in which there is a severely underrepresented class, ignore the minority class. This is due to the fact that one can achieve very high accuracy by always predicting the majority class, especially if the majority class represent 95+% of the dataset [3].

The multi-class classification problem is an extension of the traditional binary class problem where a dataset consists k classes instead of two. While imbalance is said to exist in the binary class imbalance problem when one class severely outnumbers the other class, extended to multiple classes the effects of imbalance are even more problematic. That is, given k classes, there are multiple ways for class imbalance to manifest itself in the dataset. One typical way is there is one "super majority" class which contains most of the instances in the dataset. Another typical example of class imbalance in multi-class datasets is the result

of a single minority class. In such instances $k-1$ instances each make up roughly $1/(k-1)$ of the dataset, and the "minority" class makes up the rest.

The multi-class imbalance problem is therefore interesting for two important reasons. First, as before, most learning algorithms do not deal with the wide variety of challenges multi-class imbalance presents. Secondly, a number of classifiers do not easily extend to the multi-class domain.

As a result, researchers have sought to exploit theoretical and empirical performance benefits of binary approaches for the multi-class problem. One common technique to do so is to decompose the multi-class problems into a set of binary class problems. This enables users to learn binary class classifiers on each of the subproblems which can then be combined into an ensemble in order to solve the multi-class problem. Such examples include "One-Versus-All" (OVA) and "Error Correcting Output Codes" (ECOC) [7].

One important distinction between an ensemble created using a decomposition technique, and a traditional ensemble in the binary class literature, is no single classifier in the decomposition ensemble can classify an instance in the multi-class domain. Thus while we use the word ensemble in this paper, we do not compare against traditional ensemble techniques (e.g., bagging [1], and AdaBoost [9]) as they are outside the scope of this paper. In order to avoid confusion, ensembles built using decomposition techniques will be known as "decomposition ensembles".

Contributions While the multi-class imbalance problem is a serious problem in data mining, there has been little study on the effectiveness of decision trees on the multi-class imbalanced learning problem. Recently Hellinger distance decision trees (HDDTs) have been proposed as a way of solving the class imbalance problem for decision trees without sampling. Building upon this method, we propose a modified HDDT algorithm which improve its performance on multi-class datasets, along with an analytic result to explain the relative weaknesses of HDDT in the multi-class domain. We then demonstrate the effectiveness of various decomposition techniques on improving the performance of decision trees (both C4.4 and HDDT). We then specifically demonstrate how these techniques exploit the nature of HDDT on binary imbalanced datasets to build decomposition ensembles of HDDT classifiers which outperform other decision tree methods on two widely used metrics. Finally, we provide recommendations for building decision trees for the multi-class imbalance problem.

2 Methods

We apply a variety of methods to better understand the performance of decision trees in the class imbalance problem. Due to space restrictions, we limit the study to two popular decomposition techniques (OVA and ECOC), as well as building single trees.

2.1 Decomposition Techniques

As previously discussed, decomposition techniques have become a powerful tool in the data mining community to transfer (less studied) multi-class problems into

(more studied) binary class problems. When considering decomposition techniques, an important factor is the size of the generated decomposition ensemble. Since one of the criteria when selecting decomposition methods to consider was the amount of computation time required, we selected two techniques which (generally) generate vastly different sized decomposition ensembles (and thus require vastly different computation time).

One-Versus-All Decomposition The OVA technique is one of the simplest and most natural techniques for decomposing the multi-class problem into multiple binary class problems. In OVA, given c classes, c classifiers are built such that each one considers one of the classes to be the "positive" class while the remainder are combined into a "negative" class. When a new instance is seen, each classifier returns a probability estimate for the instance. An overall probability estimate is then obtained by combining each of the individual probability estimates into a vector of length c, and normalizing.

One of the main advantages of the OVA technique is that it is conceptually simple. Rifkin and Klautau [16] argue that this simplicity, combined with its superior performance, make OVA a very desirable technique which should be considered over its more complicated alternatives.

Error Correcting Output Codes Decomposition ECOC is another popular method developed by Dietterich and Bakiri [7], which uses the concept of error correcting codes to learn a decomposition ensemble of classifiers. The choice of error correcting codes is a natural one as, assuming the codewords have hamming distance d, a maximum of $\lfloor \frac{d-1}{2} \rfloor$ errors can be made by the decomposition ensemble before misclassification occurs. This is a strong guarantee, and allows users to customize the size of the decomposition ensemble based on how many errors they expect versus the size of the codewords which they will allow.

More specifically, in ECOC each class is given an n-bit binary string called a "codeword". These codewords are generated such that the hamming distance between all codewords is maximized. Let c be an $m \times n$ matrix (where m is the number of classes), such that c_{ij} denote the jth bit for the codeword of class i. Given this, we can now learn a decomposition ensemble of n classifiers. For each classifier, the positive and negative classes are determined by the jth column of c. That is, if $c_{ij} = 1$, then class i is considered part of the positive class in classifier j. Otherwise, if $c_{ij} = 0$, class i is considered part of the negative class.

One of the most important considerations when building an ECOC decomposition ensemble is the length of the codewords. The maximum codeword length is $2^{m-1} - 1$. While building decomposition ensembles of this size results in the one most robust to errors, it also requires the most training time. Specifically, for 11 classes this method requires building a decomposition ensemble of size 1024. While given the computing power available today this is of reasonable size, as the number of classes grows the problem quickly becomes intractable. Since having so many classes is rare in practice, and does not in fact occur for any datasets in this paper, we build codewords of maximum size for all datasets.

2.2 Decision Trees

Decision trees are one of the fundamental learning algorithms in the data mining community. The most popular of decision tree learning algorithm is C4.5 [14]. Recently Hellinger distance decision trees (HDDTs) [4] have been proposed as an alternative method for building decision trees for binary class datasets which exhibit class imbalance.

Provost and Domingos [13] recommend a modification to C4.5 known as C4.4. In C4.4 decision trees are constructed by building unpruned and uncollapsed C4.5 decision trees which use Laplace smoothing at the leaves. These choices are due to empirical results [13] demonstrating that a fully built unpruned, uncollapsed tree with Laplace smoothing outperforms all other configurations, and thus are used in all experiments in this paper.

The important function to consider when building a decision tree is known as the *splitting criterion*. This function defines how data should be split in order to maximize performance. In C4.4 this function is gain ratio, which is a measure of purity based on entropy [14], while in HDDT this function is Hellinger distance. In the next section we motivate Hellinger distance as a splitting criterion, and then subsequently devise a strategy for improving its performance on multi-class datasets.

Hellinger Distance Splitting Criterion Hellinger distance is a distance metric between probability distributions used by Cieslak and Chawla [4] to create Hellinger distance decision trees (HDDTs). It was chosen as a splitting criterion for the binary class imbalance problem due to its property of skew insensitivity. Hellinger distance is defined as a splitting criterion as [4]:

$$d_H(X_+, X_-) = \sqrt{\sum_{j=1}^{p} \left(\sqrt{\frac{|X_{+j}|}{|X_+|}} - \sqrt{\frac{|X_{-j}|}{|X_-|}} \right)^2} \tag{1}$$

where X_+ is the set of all positive examples, X_- is the set of all negative examples and X_{+j} (X_{-j}) is the set of positive (negative) examples with the jth value (of p distinct values) of the relevant feature.

Since Hellinger distance defines the distance between probability distributions, it does not naturally extend to the multi-class problem. This is in contrast to gain ratio — which is based on entropy — which is easily extensible to any number of classes. Specifically, since Hellinger distance is a distance metric, any natural extension would be attempting to determine the distance between c probability distributions, where c is the number of classes. Since this is not a well defined problem, we propose an extension to the HDDT algorithm for the multi-class problem.

Multi-Class HDDT In order to overcome the shortcomings of Hellinger distance as a splitting criterion for the multi-class problem, we employ techniques similar to the decomposition algorithms described in Section 2.1. That is, given the set of classes C, we consider each unique pair of subsets: $C_1 \subset C$, $C_2 = C \backslash C_1$

Algorithm 1 *Calc_Multi_Class_Hellinger*

Require: Training set T, Feature f, Set of classes C

Let Hellinger $\leftarrow -1$.
Let V_f be the set of values of feature f.
for each pair of subsets of C: $C_1 \subset C$, $C_2 = C \setminus C_1$: **do**
  for each value $v \in V_f$ **do**
    Let $w \leftarrow V_f \setminus v$
    cur_value $\leftarrow$ $(\sqrt{|T_{f,v,+}|/|T_+|} - \sqrt{|T_{f,v,-}|/|T_-|})^2 + (\sqrt{|T_{f,w,+}|/|T_+|} - \sqrt{|T_{f,w,-}|/|T_-|})^2$
    if *cur_value* > Hellinger **then**
      Hellinger $\leftarrow$ *cur_value*
    end if
  end for
end for
return $\sqrt{\text{Hellinger}}$

and consider all classes in C_1 as the positive class, and all classes in C_2 as the negative class[1].

Algorithm 1 outlines the approach to incorporating Hellinger distance in learning multi-class decision trees. Let T_C indicate the subset of training set T which has its class in set C, and $T_{k,j,C}$ identifies the subset which has its class in set C and has value j for feature k.

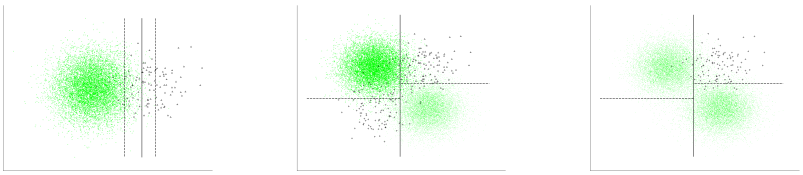
The important aspect of this version of the Hellinger distance splitting criterion is the reduction of the multiple classes into all relevant binary class possibilities. This choice enables Hellinger distance to try find the best split between all possible choices of positive and negative class, and thus any meaningful split available to it in the multi-class domain.

This distance calculator can then be used as the splitting criterion in a decision tree algorithm in order to build multi-class HDDTs (MC-HDDTs). Comparing MC-HDDTs further to HDDT shows us that for the binary class problem, exactly the same tree will be learned as the original version. *Our algorithm can therefore be recommended in lieu of HDDT, as it returns the same tree for the binary case while offering better performance on the multi-class problem.*

3 Analysis of the Splitting Criteria

One of the major research questions in this paper is why the performance of HDDT suffers in the multi-class case when compared to C4.4, especially in light of their performances on binary class imbalanced datasets. In this section we present an analytic example which demonstrates how HDDT and C4.4 behave when a binary class problem is transformed into a multi-class problem and then back again. Due to space limitations we limit ourselves to a single example which

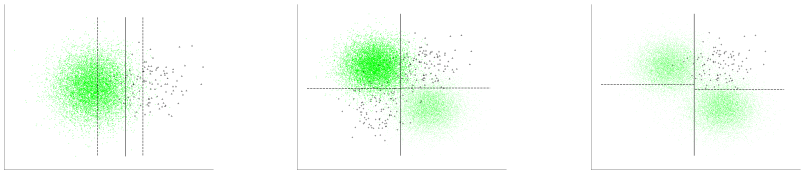
[1] Note that two pairs of subsets (e.g., (C_1, C_2) and (D_1, D_2)) are considered equal if $C_1 = D_1$ or $C_1 = D_2$



(a) Effects of 100:1 imbalance ratio.

(b) Effects of 100:1:100:1 imbalance.

(c) Effects of OVA on 100:1:100:1 imbalance.



(d) Effects of 100:1 imbalance.

(e) Effects of 100:1:100:1 imbalance.

(f) Effects of OVA on 100:1:100:1 imbalance.

Fig. 1. Comparison of the effects of various class distributions on the ability of information gain (top) and Hellinger distance (bottom) to correctly determine the class boundary which optimizes AUC.

demonstrates an example of Hellinger distance performing poorly in the multi-class case.

For our analytic example we created a simulated dataset with 4 classes, with centers on the corners of a square, such that their means were separated by 2σ. In the upper left and lower right corners, we simulated 10,000 examples, while in the lower left and upper right corners we simulated only 100 examples. This gives us a class imbalance ratio of 10,000:100:10,000:100 (C.V.: 0.98). We then decomposed the 4 class problem into a binary class problem by removing the lower half of the square (as depicted in Figure 3). In order to determine their performance, each of the experiments was run 100 times, and the (W)AUROC (defined in Section 4) computed.

Figures 1(a) and 1(d) are representative examples of the effects of gain ratio and Hellinger distance (respectively) on the binary class problem. From the splits, we see that Hellinger distance is much more aggressive when splitting into the majority class. When considering their performance, we see that, based on AUROC, HDDT wins 85 out of the 100 runs. This increase in performance is therefore an effect of Hellinger distance aggressively attempting to capture as much of the minority class as possible, while gain ratio remains very conservative.

In the multi-class case (Figures 1(b) and 1(e)) Hellinger distance once again is very aggressive in attempting to capture as much of the minority class as possible, while C4.4 is much more conservative. Due to the nature of this problem, however, the more conservative approach is better able to capture the multi-distributional aspect of the problem. This is demonstrated by the fact

that, based on WAUROCs, C4.4 wins 82 of the 100 runs. Thus, for multi-class, Hellinger distance is not able to adequately separate the two classes, instead being overwhelmed by the spurious information from the extra classes.

In order to better understand this phenomena, consider the right-most horizontal split Hellinger makes in the multi-class case. For this split, Hellinger distance considers the "top" points to be the positive class and the "bottom" points to be the negative class. As evidenced by the inaccuracy of the top left points, Hellinger distance is not able to accurately partition the space. Gain ratio, on the other hand is able to arrive at a better split point which more accurately represents the boundary for this problem.

Finally we consider the case of OVA decomposition on the dataset. Figure 1(f) shows Hellinger distance is very good at capturing the minority class. This favorable splitting is exactly what would be expected from such a binary class imbalanced dataset, and thus explains the performance increase HDDT sees over C4.4 when used in conjunction with OVA. This hypothesis is further confirmed when we note that HDDT obtains a higher AUROC in 80 of the 100 runs, thus confirming that it is the preferred classifier to use.

Given these results, we now better understand the dynamics of Hellinger distance in the binary class problem which result in inferior performance in the multi-class domain. Further research into overcoming these challenges might prove useful in developing a single decision tree approach which, without sampling, is able to outperform the others in the case of multi-class imbalance.

4 Experiments

We implemented MC-HDDT in WEKA [10], and used WEKA's built-in OVA and ECOC to train each of the classifiers. In order to make fair comparisons, we split the experiments into three separate categories, namely: single trees, OVA decomposition, and ECOC decomposition. This separation is done to highlight the difference in performance of HDDT and C4.4 under different decomposition techniques. That is, by comparing each method within a category, we are providing a fair comparison of the different decision tree techniques.

Table 1 gives the relevant simple statistics about the datasets used in this paper. One of the main goals when choosing the datasets to consider was ensuring that they were imbalanced. To measure imbalance in multi-class datasets, we use the "coefficient of variation" (C.V.) as recommended by Cieslak and Chawla [5]. Specifically, C.V. is the proportion of the deviation in the observed number of examples for each class versus the expected number of examples in each class. In this paper we consider datasets with a C.V. above 0.35 – a class ratio of 2:1 on a binary dataset – imbalanced. This leaves us with the 17 datasets listed.

4.1 Configuration

In order to ensure a fair comparison of the methods, we ran 50 iterations [15] of 2-fold cross-validation. We chose 2-fold cross-validation due to the small number of instances of some classes in the datasets. Due to space restrictions, we only consider weighted area under the receiver operating characteristic (WAUROC)

Table 1. Statistics for the datasets used in this paper. C.V. is the coefficient of variation, # Ftrs is the number of features, and # Insts is the number of instances.

Dataset	C.V.	# Ftrs	# Insts	# Classes
abalone	0.711	9	4177	4
artificial	0.594	8	5109	5
auto-mpg	0.621	8	398	3
balance-scale	0.541	5	625	3
bgp	1.260	9	24984	4
car	1.082	7	1728	4
connect-4	0.714	43	67557	3
dermatology	0.455	35	366	6
dna	0.394	180	3186	3
glass	0.761	9	214	6
page-blocks-5	1.747	10	5473	5
sat	0.372	36	6435	6
segment	0.535	20	2310	3
solar-flare-2	0.535	12	1066	6
splice	0.393	61	3190	3
vehicle	0.370	19	846	3
yeast	1.005	9	1484	9

[19]. We chjse this metrics as it is a commonly used criterion when comparing classifiers in the multi-class imbalance case.

4.2 Statistical Tests

While many different techniques have been applied to attempt to compare classifier performance across multiple datasets, Demšar suggests comparisons based on ranks. We follow this recommendation and rank the performance of each classifier by its average performance, with 1 being the best. Since we seek to determine whether or the HDDT methods are statistically significantly better than the existing methods, we use the Friedman and Bonferroni-Dunn tests as was recommended by Demšar [6].

The Friedman test is first applied to determine if there is a statistically significant difference between the rankings of the classifiers. That is, it tests to see if the rankings are not merely randomly distributed. Next, as recommended by Demšar, we preform the Bonferroni-Dunn test to compare each classifier against the control classifier.

4.3 Results

As stated previously we break the experiment into three different categories. Each of the categories corresponds to a different level of computational effort required to construct the classifier, with single trees requiring the least amount of work, and ECOC requiring the most. For the sake of space, however, the WAUROC values for each of the methods is presented in Table 2.

Table 2 also contains the results of the statistical test described in Section 4.2. A classifier receives a check mark if it is considered statistically significantly worse than the best classifier (i.e., the classifier with the lowest average rank) in its category (e.g., single tree, OVA, ECOC) at the noted confidence level.

Single Tree Performance When considering the single tree performances, C4.4 and MC-HDDT perform equivalently. This is an interesting result for multi-class imbalanced data sets, and further corroborates the intuition established

Table 2. WAUROC values for the various methods over each of the datasets. Bold numbers indicate overall best performance. The number in parenthesis indicates the rank in the category. A ✓ indicates that the method performs statistically significantly worse than the other method in its category at the relevant confidence level.

	Single Tree		OVA		ECOC	
Dataset	C4.4	MC-HDDT	C4.4	HDDT	C4.4	HDDT
abalone	0.71484 (1)	0.71007 (2)	0.73751 (2)	0.74073 (1)	**0.74869 (1)**	0.74803 (2)
artificial	0.87548 (2)	0.87932 (1)	0.87913 (2)	0.88178 (1)	0.90344 (2)	**0.90474 (1)**
auto-mpg	0.90093 (2)	0.90806 (1)	0.91153 (2)	0.91476 (1)	0.91153 (2)	0.91476 (1)
balance-scale	0.90607 (1)	0.90495 (2)	0.90486 (2)	0.90651 (1)	0.90486 (2)	**0.90651 (1)**
bgp	0.80268 (1)	0.79698 (2)	0.81378 (2)	0.81414 (1)	0.82418 (2)	**0.82446 (1)**
car	0.97662 (2)	0.98652 (1)	0.98209 (2)	0.99244 (1)	0.98262 (2)	**0.99413 (1)**
connect-4	0.87879 (1)	0.85156 (2)	0.88971 (1)	0.87900 (2)	**0.88971 (1)**	0.87900 (2)
dermatology	0.97794 (2)	0.98283 (1)	0.98357 (2)	0.99079 (1)	0.99594 (2)	**0.99682 (1)**
dna	0.97508 (1)	0.96680 (2)	**0.98436 (1)**	0.98232 (1)	0.98436 (1)	0.98232 (2)
glass	0.80585 (1)	0.79370 (2)	0.83867 (2)	0.84393 (1)	0.88161 (2)	**0.88348 (1)**
page-blocks-5	0.98104 (2)	0.98111 (1)	0.98446 (2)	0.98480 (1)	0.98731 (2)	**0.98940 (1)**
sat	0.96262 (1)	0.96124 (2)	0.97273 (2)	0.97471 (1)	0.98679 (2)	**0.98715 (1)**
segment	0.98656 (2)	0.98848 (1)	0.99437 (2)	0.99650 (1)	0.99437 (2)	0.99650 (1)
solar-flare-2	0.91886 (2)	0.92032 (1)	0.91683 (2)	0.91856 (1)	**0.92098 (1)**	0.92035 (2)
splice	0.97459 (2)	0.97476 (1)	0.98457 (1)	0.98262 (2)	0.98457 (1)	0.98262 (2)
vehicle	0.95696 (2)	0.96139 (1)	0.97164 (2)	0.97717 (1)	0.97164 (2)	0.97717 (1)
yeast	0.73156 (1)	0.71541 (2)	0.76973 (2)	0.77137 (1)	0.80843 (2)	**0.80871 (1)**
Avg. Rank	1.52941	1.47059	1.82353	1.17647	1.70588	1.29412
$\alpha = 0.05$			✓			
$\alpha = 0.10$			✓		✓	

with the illustrations in Section 3. As discussed, this is mainly due to the aggressive nature of the splits which Hellinger distance tries to create. The consequence of this analysis is further evidenced in the OVA performance.

Hellinger distance, as a criterion, is limited in capturing the multi-class divergences. Nevertheless, we recommend MC-HDDT as a decision tree classifier, as it reduces to HDDT for binary class datasets (achieving statistically significantly superior performance over C4.4 [4]), and is a competitive alternative to C4.4 for multi-class datasets (no statistically significant variation in performance).

OVA Performance When considering OVA performance, HDDT significantly outperforms C4.4. This result confirms our understanding of the binary class performances of each of the classifiers. That is, when decomposing the multi-class problem into multiple binary problems, the binary class problems obtained are (often) extremely imbalanced. This fact is further exacerbated by the fact that the multi-class dataset itself is highly imbalanced.

Thus in the OVA approach, each binary classifier in the decomposition ensemble must deal with the class imbalance problem. Since HDDT has been shown to perform statistically significantly better than C4.4 in this scenario, we expect to see HDDT outperforming C4.4 when using the OVA approach. Based on the observations obtained, we can conclude that our intuition is correct and, furthermore, that when using OVA decomposition for multi-class imbalance, HDDT are the appropriate decision tree learning to choose.

ECOC Performance When comparing the relative performance of the classifiers, we see that HDDT outperforms C4.4 almost as well as in the OVA approach. While the statistical significance is only $\alpha = 0.10$, we see that it is not statistically significant at the $\alpha = 0.05$ threshold by one dataset. As Table 2

shows, some of the performance differences were quite small. Thus it seems reasonable to believe that with more datasets we might see the same statistical significance with this method as was shown in OVA, as we would expect the same performance gains of using HDDT over C4.4 in this case as well.

This expectations of better performance of HDDT over C4.4 is due to similar reasoning as the OVA case. That is, by decomposing the problems into multiple binary problems, the class imbalance will still be a major concern. However, the ECOC approach will result in $2^{m-1} - 1$ binary datasets. Some of these will be highly imbalanced, while others may be balanced depending on the respective class distributions. Nevertheless, HDDT is able to capitalize with ECOC. It is able to achieve stronger separability on highly imbalanced combinations, and achieves comparable performance to C4.4 on the relatively balanced class combinations, and thus, as a collective, it is able to outperform C4.4.

Overall Performance When considering the overall performance of each method as given in Table 2, we see that, in general, the more computational power used, the better the performance. That is, the ECOC methods outperform the OVA methods which outperform the single tree methods.

This is an unsurprising result, as a wealth of data mining literature demonstrates that combining a large number of classifiers into an ensemble is a powerful technique for increasing performance. The decomposition ensemble techniques employed in this paper are also of particular interest, as the diversity of the classifiers created in the decomposition ensembles is quite high. That is, since the class values under consideration are changing between datasets, the classifiers are not merely learning on different permutations of the underlying instances, instead having the decision boundaries themselves change. It is well known that diversity is important to creating good ensembles [11].

5 Related work

A number of methods have been proposed to counter the class imbalance issue, however a large portion has focused on the binary class problem. Sampling methods have emerged as a de facto standard, but present numerous challenges when being extended to multiple classes. This is due to the complexity arising from the combination of multiple class imbalance types, different amounts of sampling, different sampling methods, and different cost matrices. Thus to apply any reasonable optimization criteria to discovering the optimal sampling amount is computationally prohibitive.

Rescaling [8] is a general method for cost-sensitive and class-imbalance problems which changes the distribution of the original data [20]. As there are many methods that can change the distribution, rescaling can be realized in numerous ways (e.g., by sampling, instance-weighting, threshold moving, etc.). Sampling is a widely used rescaling method to deal with the class-imbalance problem. The method balances the distribution modifying the training set to either increase the presence of the minority class (e.g., random oversampling, SMOTE [2]), or reduce the majority class (e.g., undersampling). Another popular rescaling method is instance-weighting. In this method, instead of removing or adding in-

stances, a weight is generated for each instance according to its misclassification cost, which is passed to a cost-blind classifier which uses instance weights [17]. A final common approach is threshold moving, wherein the decision threshold is modified in order to achieve the minimal cost in cost-sensitive learning [8, 21].

Cost sensitive learning methods have been developed to deal with the different costs of misclassification [18]. For example, the cost of misclassifying a cancer patient as healthy is much higher than the cost of misclassifying a healthy patient as having cancer. Given this, cost sensitive problems require the minimization of the misclassification cost rather than misclassification errors. The class-imbalance problem can thus be considered a cost-sensitive problem where the costs are unequal and unknown [12]. Most cost-sensitive learning methods are actually based on rescaling [20], and therefore it is natural that by assigning the appropriate misclassification cost for each class, cost-sensitive approaches can be used to deal with the class-imbalance problems.

6 Conclusion and Discussion

In this paper we compared different methods of building C4.4 and Hellinger distance decision trees for multi-class imbalanced datasets. Given the different amounts of computation time required by each method, we investigated the problem in three separate categories: single tree, OVA, and ECOC.

In the single tree case we found that MC-HDDT performs comparably to C4.4. While MC-HDDT does not statistically significantly outperform C4.4, it is a reasonable alternative to C4.4 for all classification problems. This is an important result as it gives practitioners another viable tool to use when confronted with a new dataset.

Alternatively, when the analysis was extended to build decomposition ensembles of binary classifiers HDDT became the clear choice. Given the skew insensitivity of Hellinger distance as a splitting criterion, coupled with the nature of skew in the resulting problems, the gains in performance become significant. With this in mind, we recommend HDDT for all multi-class imbalanced learning when used in a decomposition method.

Another important observation stems from the overall performance. Specifically we see that the more complex (and thus more computationally intensive) algorithms give real gains in performance. We can therefore revise our recommendation, this time recommending the use of HDDT in an ECOC decomposition ensemble if the user has enough computational power. Otherwise, the user should consider an OVA decomposition ensemble with HDDT, and, finally, if not enough computational power exists for such a decomposition, building MC-HDDTs. We recommend MC-HDDTs over C4.4, as even though the difference between them is not statistically significant for multi-class datasets, MC-HDDT reduces to HDDT for binary class datasets, where it has been demonstrated to be strongly skew insensitive and statistically significantly over C4.4. As a result, MC-HDDT may be considered the recommended decision tree algorithm.

Finally, Section 3 illustrated the challenges Hellinger distance faces in the multi-class domain. With this understanding further research can now explore

the problem of multi-class Hellinger distance and attempt to overcome the demonstrated difficulties to provide a robust classifiers for multi-class problems.

Acknowledgements

Work is supported in part by the NSF Grant ECCS-0926170, NSFC (60903103, 61021062) and the Notebaert Premier Fellowship.

References

1. Breiman, L.: Bagging predictors. Machine Learning 24(2), 123–140 (1996)
2. Chawla, N.V., Bowyer, K.W., Hall, L.O., Kegelmeyer, W.P.: Smote: Synthetic minority over-sampling technique. JAIR 16, 321–357 (2002)
3. Chawla, N.: Data mining for imbalanced datasets: An overview. Data Mining and Knowledge Discovery Handbook pp. 875–886 (2010)
4. Cieslak, D.A., Chawla, N.V.: Learning decision trees for unbalanced data. ECML pp. 241–256 (2008)
5. Cieslak, D.A., Chawla, N.V.: Start globally, optimize locally, predict globally: Improving performance on imbalanced data. ICDM pp. 143–152 (2008)
6. Demšar, J.: Statistical comparisons of classifiers over multiple data sets. JMLR 7, 30 (2006)
7. Dietterich, T., Bakiri, G.: Error-correcting output codes: A general method for improving multiclass inductive learning programs. In: AAAI. pp. 395–395 (1994)
8. Domingos, P.: Metacost: A general method for making classifiers cost-sensitive. KDD pp. 155–164 (1999)
9. Freund, Y., Schapire, R.: A desicion-theoretic generalization of on-line learning and an application to boosting. CoLT pp. 23–37 (1995)
10. Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.H.: The weka data mining software: an update. SIGKDD Exp. News. 11(1), 10–18 (2009)
11. Krogh, A., Vedelsby, J.: Neural network ensembles, cross validation, and active learning. NIPS pp. 231–238 (1995)
12. Maloof, M.A.: Learning when data sets are imbalanced and when costs are unequal and unknown. ICML WLIDS (2003)
13. Provost, F., Domingos, P.: Tree induction for probability-based ranking. Machine Learning 52(3), 199–215 (2003)
14. Quinlan, J.: Induction of decision trees. Machine learning 1(1), 81–106 (1986)
15. Raeder, T., Hoens, T., Chawla, N.: Consequences of Variability in Classifier Performance Estimates. ICDM pp. 421–430 (2010)
16. Rifkin, R., Klautau, A.: In defense of one-vs-all classification. JMLR 5, 101–141 (2004)
17. Ting, K.M.: An instance-weighting method to induce cost-sensitive trees. TKDE 14(3), 659–665 (2002)
18. Turney, P.D.: Types of cost in inductive concept learning. ICML pp. 15–21 (2000)
19. Van Calster, B., Van Belle, V., Condous, G., Bourne, T., Timmerman, D., Van Huffel, S.: Multi-class auc metrics and weighted alternatives. IJCNN pp. 1390–1396 (2008)
20. Zhou, Z.H., Liu, X.Y.: On multi-class cost-sensitive learning. AAAI pp. 567–572 (2006)
21. Zhou, Z.H., Liu, X.Y.: Training cost-sensitive neural networks with methods addressing the class imbalance problem. TKDE 18(1), 63–77 (2006)