# A Robust Decision Tree Algorithm for Imbalanced Data Sets

Wei Liu* Sanjay Chawla* David A. Cieslak† Nitesh V. Chawla†

**Abstract**

We propose a new decision tree algorithm, Class Confidence Proportion Decision Tree (CCPDT), which is robust and insensitive to size of classes and generates rules which are statistically significant.

In order to make decision trees robust, we begin by expressing Information Gain, the metric used in C4.5, in terms of *confidence* of a rule. This allows us to immediately explain why Information Gain, like confidence, results in rules which are biased towards the majority class. To overcome this bias, we introduce a new measure, Class Confidence Proportion (CCP), which forms the basis of CCPDT. To generate rules which are statistically significant we design a novel and efficient top-down and bottom-up approach which uses Fisher's exact test to prune branches of the tree which are not statistically significant. Together these two changes yield a classifier that performs statistically better than not only traditional decision trees but also trees learned from data that has been balanced by well known sampling techniques. Our claims are confirmed through extensive experiments and comparisons against C4.5, CART, HDDT and SPARCCC.

## 1 Introduction

While there are several types of classifiers, rule-based classifiers have the distinct advantage of being easily interpretable. This is especially true in a "data mining" setting, where the high dimensionality of data often means that apriori very little is known about the underlying mechanism which generated the data.

Decision trees are perhaps the most popular form of rule-based classifiers (such as the well-known C4.5 [15]). Recently however, classifiers based on association rules have also become popular [19] which are often called associative classifiers. Associative classifiers use association rule mining to discover interesting and significant rules from the training data, and the set of rules discovered constitute the classifier. The canonical example of an associative classifier is CBA (classification based on associations) [14], which uses the minimum support and confidence framework to find rules. The accuracy of associative classifiers depends on the quality of their discovered rules. However, the success of both decision trees and associate classifiers depends on the assumption that there is *an equal amount* of information for each class contained in the training data. In binary classification problems, if there is a similar number of instances for both positive and negative classes, both C4.5 and CBA generally perform well. On the other hand, if the training data set tends to have an imbalanced class distribution, both types of classifier will have a bias towards the majority class. As it happens, an accurate prediction is typically related to the minority class – the class that is usually of greater interest.

One way of solving the imbalance class problem is to modify the class distributions in the training data by over-sampling the minority class or under-sampling the majority class. For instance, SMOTE [5] uses over-sampling to increase the number of the minority class instances, by creating synthetic samples. Further variations on SMOTE [7] have integrated boosting with sampling strategies to better model the minority class, by focusing on difficult samples that belong to both minority and majority classes.

Nonetheless, data sampling is not the only way to deal with class imbalanced problems: some specifically designed "imbalanced data oriented" algorithms can perform well on the original unmodified imbalanced data sets. For example, a variation on associative classifier called SPARCCC [19] has been shown to outperform CBA [14] and CMAR [13] on imbalanced data sets. The downside of SPARCCC is that it generates a large number of rules. This seems to be a feature of all associative classifiers and negates many of the advantages of rule-based classification.

In [8], the Hellinger distance (HDDT) was used as the decision tree splitting criterion and shown to be insensitive towards class distribution skewness. We will compare and discuss CCPDT and HDDT more extensively in Section 3.4. Here it will be suffice to state that while HDDT is based on likelihood difference, CCPDT is based on likelihood ratio.

In order to prevent trees from over-fitting the data, all decision trees use some form of pruning. Traditional pruning algorithms are based on error estimations - a node is pruned if the predicted error rate is decreased. But this pruning technique will not always perform well on imbalanced data sets. [4] has shown that pruning in C4.5 can have a detrimental effect on learning from imbalanced data

*Centre for Distributed and High Performance Computing, School of Information Technologies, the University of Sydney, Sydney NSW 2006, Australia. {weiliu, chawla}@it.usyd.edu.au

†University of Notre Dame, Notre Dame IN 46556, USA. dcieslak@cse.nd.edu, nchawla@nd.edu

sets, since lower error rates can be achieved by removing the branches that lead to minority class leaves. In contrast our pruning is based on Fisher's exact test, which checks if a path in a decision tree is statistically significant; and if not, the path will be pruned. As an added advantage, every resulting tree path (rule) will also be statistically significant.

**Main Insight** The main insight of the paper can be summarized as follows. Let $X$ be an attribute and $y$ a class. Let $X \to y$ and $\neg X \to y$ be two rules, with confidence $p$ and $q$ respectively. Then we can express Information Gain (IG) in terms of the two confidences. Abstractly,

$$IG_{C4.5} = F(p, q)$$

where *F* is an abstract function. We will show that a splitting measure based on confidence will be biased towards the majority class. Our innovation is to use Class Confidence Proportion (CCP) instead of confidence. Abstractly CCP of the $X \to y$ and $\neg X \to y$ is $r$ and $s$. We define a new splitting criterion

$$IG_{CCPDT} = F^{'}(r, s)$$

The main thrust in the paper is to show that $IG_{CCPDT}$[1] is more robust to class imbalance than $IG_{C4.5}$ and behave similarly when the classes are balanced.

The approach of replacing a conventional splitting measure by CCP is a generic mechanism for all traditional decision trees that are based on the "balanced data" assumption. It can be applied to any decision tree algorithm that checks the degree of impurity *inside a partitioned branch*, such as C4.5 and CART etc.

The rest of the paper is structured as follows. In Section 2, we analyze the factors that causes CBA and C4.5 perform poorly on imbalanced data sets. In Section 3, we introduce CCP as the measure of splitting attributes during decision tree construction. In Section 4 we present a full decision tree algorithm which details how we incorporate CCP and use Fisher's Exact Test (FET) for pruning. A wrapper framework utilizing sampling techniques is introduced in Section 5. Experiments, Results and Analysis are presented in Section 6. We conclude in Section 7 with directions for research.

## 2 Rule-based Classifiers

We analyze the metrics used by rule-based classifiers in the context of imbalanced data. We first show that the ranking of rules based on confidence is biased towards the majority class, and then express information gain and Gini index as functions of confidence and show that they also suffer from similar problems.

[1] $IG_{CCPDT}$ is not Information Gain which has a specific meaning.

Table 1: An example of notations for CBA analysis

| | $X$ | $\neg X$ | $\Sigma$ Instances |
|---|---|---|---|
| $y$ | $a$ | $b$ | $a+b$ |
| $\neg y$ | $c$ | $d$ | $c+d$ |
| $\Sigma$ Attributes | $a+c$ | $b+d$ | $n$ |

**2.1 CBA** The performance of Associative Classifiers depends on the quality of the rules it discovers during the training process. We now demonstrate that in an imbalanced setting, confidence is biased towards the majority class.

Suppose we have a training data set which consists of $n$ records, and the antecedents (denoted by $X$ and $\neg X$) and class ($y$ and $\neg y$) distributions are in the form of Table 1. The rule selection strategy in CBA is to find all rule items that have support and confidence above some predefined thresholds. For a rule $X \to y$, its confidence is defined as:

$$Conf(X \to y) = \frac{Supp(X \cup y)}{Supp(X)} = \frac{a}{a+c} \tag{2.1}$$

"*Conf*" and "*Supp*" stand for *Confidence* and *Support*. Similarly, we have:

$$Conf(X \to \neg y) = \frac{Supp(X \cup \neg y)}{Supp(X)} = \frac{c}{a+c} \tag{2.2}$$

Equation 2.1 suggests that selecting the highest confidence rules means choosing the most frequent class among all the instances that contains that antecedent (i.e. $X$ in this example). However, for imbalanced data sets, since the size of the positive class is always much smaller than the negative class, we always have: $a + b \ll c + d$ (suppose $y$ is the positive class). Given that imbalanced data do not affect the distribution of antecedents, we can, without loss of generality, assume that $X$s and $\neg X$s are nearly equally distributed. Hence when data is imbalanced, $a$ and $b$ are both small while $c$ and $d$ are both large. Even if $y$ is supposed to occur with $X$ more frequently than $\neg y$, $c$ is unlikely to be less than $a$ because the positive class size will be much smaller than the negative class size. Thus, it is not surprising that the right-side term in Equation 2.2 always tends to be lower bounded by the right-side term in Equation 2.1. It appears that even though the rule $X \to \neg y$ may not be significant, it is easy for it to have a high confidence value.

In these circumstances, it is very hard for the confidence of a "good" rule $X \to y$ to be significantly larger than that of a "bad" rule $X \to \neg y$. What is more, because of its low confidence, during the classifier building process a "good" rule may be ranked behind some other rules just because they have a higher confidence because they predict the majority class. This is a fatal error, since in an imbalanced class problem it is often the minority class that is of more interest.

**2.2 Traditional decision trees** Decision trees such as C4.5 use information gain to decide which variable to split [15]. The information gain from splitting a node *t* is defined as:

$$InfoGain_{split} = Entropy(t) - \sum_{i=1,2} \frac{n_i}{n} Entropy(i) \tag{2.3}$$

where *i* represents one of the sub-nodes after splitting (assume there are 2 sub-nodes), $n_i$ is the number of instances in subnote *i*, and *n* stands for the total number of instances. In binary-class classification, the entropy of node *t* is defined as:

$$Entropy(t) = -\sum_{j=1,2} p(j|t) \log p(j|t) \tag{2.4}$$

where *j* represents one of the two classes. For a fixed training set (or its subsets), the first term in Equation 2.3 is fixed, because the number of instances for each class (i.e. $p(j|t)$ in equation 2.4) is the same for all attributes. To this end, the challenge of maximizing information gain in Equation 2.3 reduces to maximizing the second term $-\sum_{i=1,2} \frac{n_i}{n} Entropy(i)$.

If the node $t$ is split into two subnodes with two corresponding paths: $X$ and $\neg X$, and the instances in each node have two classes denoted by $y$ and $\neg y$, Equation 2.3 can be rewritten as:

$$\begin{aligned} &InfoGain_{split} = Entropy(t) \\ &- \frac{n_1}{n} \left[-p(y|X)\log(y|X) - p(\neg y|X)\log p(\neg y|X)\right] \\ &- \frac{n_2}{n} \left[-p(y|\neg X)\log(y|\neg X) - p(\neg y|\neg X)\log p(\neg y|\neg X)\right] \end{aligned} \tag{2.5}$$

Note that the probability of $y$ given $X$ is equivalent to the confidence of $X \rightarrow y$:

$$p(y|X) = \frac{p(X \cap y)}{p(X)} = \frac{Support(X \cup y)}{Support(X)} = Conf(X \rightarrow y) \tag{2.6}$$

Then if we denote $Conf(X \rightarrow y)$ by $p$, and denote $Conf(\neg X \rightarrow y)$ by $q$ (hence $Conf(X \rightarrow \neg y) = 1 - p$ and $Conf(\neg X \rightarrow \neg y) = 1 - q$), and ignore the "fixed" terms *Entropy(t)* in equation 2.5, we can obtain the relationship in Equation 2.7.

The first approximation step in Equation 2.7 ignores the first term *Entropy(t)*, then the second approximation transforms the addition of logarithms to multiplications.

Based on Equation 2.7 the distribution of information gain as a function of $Conf(X \rightarrow y)$ and $Conf(\neg X \rightarrow y)$ is shown in Figure 1. Information gain is maximized when $Conf(X \rightarrow y)$ and $Conf(\neg X \rightarrow y)$ are both close to either 0 or 1, and is minimized when $Conf(X \rightarrow y)$ and $Conf(\neg X \rightarrow y)$ are both close to 0.5. Note that when $Conf(X \rightarrow y)$ is close to 0, $Conf(X \rightarrow \neg y)$ is close to 1; when $Conf(\neg X \rightarrow y)$ is close to 0, $Conf(\neg X \rightarrow \neg y)$ is close to 1. Therefore, information gain achieves the highest value when either $X \rightarrow y$ or $X \rightarrow \neg y$ has the highest confidence, and either $\neg X \rightarrow y$ or $\neg X \rightarrow \neg y$ also has the highest confidence.

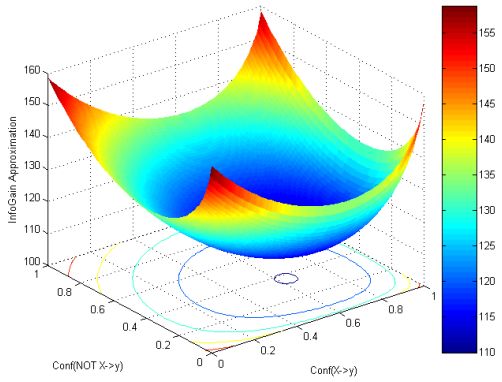



Figure 1: Approximation of information gain in the formula formed by $Conf(X \rightarrow y)$ and $Conf(\neg X \rightarrow y)$ from Equation 2.7. Information gain is the lowest when $Conf(X \rightarrow y)$ and $Conf(\neg X \rightarrow y)$ are both close to 0.5, and is the highest when both $Conf(X \rightarrow y)$ and $Conf(\neg X \rightarrow y)$ reaches 1 or 0.

$$\begin{aligned} &InfoGain_{split} = Entropy(t) + \sum_{i=1,2} \frac{n_i}{n} Entropy(i) \\ =&Entropy(t) + \frac{n_1}{n}[p\log p + (1-p)\log(1-p)] \\ &+ \frac{n_2}{n}[q\log q + (1-q)\log(1-q)] \\ \propto&\frac{n_1}{n}[p\log p + (1-p)\log(1-p)] \\ &+ \frac{n_2}{n}[q\log q + (1-q)\log(1-q)] \\ \propto&\frac{n_1}{n}\log p^p(1-p)^{1-p} + \frac{n_2}{n}\log q^q(1-q)^{1-q} \end{aligned} \tag{2.7}$$

Therefore, decision trees such as C4.5 split an attribute whose partition provides the highest confidence. This strategy is very similar to the rule-ranking mechanism of association classifiers. As we have analyzed in Section 2.1, for imbalanced data set, high confidence rules do not necessarily imply high significance in imbalanced data, and some significant rules may not yield high confidence. Thus we can assert that the splitting criteria in C4.5 is suitable for balanced but not imbalanced data sets.

We note that it is the term $p(j|t)$ in Equation 2.4 that is the cause of the poor behavior of C4.5 in imbalanced situations. However, $p(j|t)$ also appears in other decision

Table 2: Confusion Matrix for the classification of two classes

| *All instances* | *Predicted positive* | *Predicted negative* |
|---|---|---|
| *Actual positive* | true positive (tp) | false negative (fn) |
| *Actual negative* | false positive (fp) | true negative(tn) |

tree measures. For example, the Gini index defined in CART [2] can be expressed as:

$$Gini(t) = 1 - \sum_j p(j|t)^2 \tag{2.8}$$

Thus decision tree based on CART will too suffer from the imbalanced class problem. We now propose another measure which will be more robust in the imbalanced data situation.

## 3 Class Confidence Proportion and Fisher's Exact Test

Having identified the weakness of the support-confidence framework and the factor that results in the poor performance of entropy and Gini index, we are now in a position to propose new measures to address the problem.

**3.1 Class Confidence Proportion** As previously explained, the high frequency with which a particular class $y$ appears together with $X$ does not necessarily mean that $X$ "explains" the class $y$, because $y$ could be the overwhelming majority class. In such cases, it is reasonable that instead of focusing on the antecedents ($X$s), we focus only on each class and find the most significant antecedents associated with that class. In this way, all instances are partitioned according to the class they contain, and consequently instances that belong to different classes will not have an impact on each other. To this end, we define a new concept, Class Confidence (CC), to find the most interesting antecedents ($X$s) from all the classes ($y$s):

$$CC(X \to y) = \frac{Supp(X \cup y)}{Supp(y)} \tag{3.9}$$

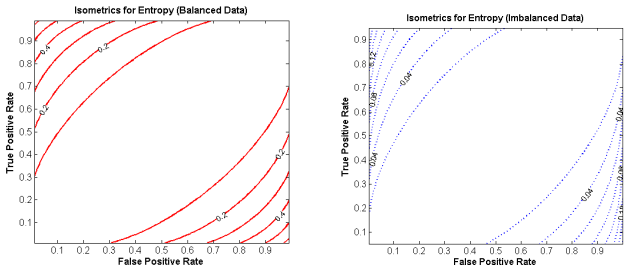
The main difference between this CC and traditional confidence is the denominator: we use $Supp(y)$ instead of $Supp(X)$ so as to focus only on each class.

In the notation of the confusion matrix (Table 2) CC can be expressed as:

$$CC(X \to y) = \frac{TruePositiveInstances}{ActualPositiveInstances} = \frac{tp}{tp + fn} \tag{3.10}$$

$$CC(X \to \neg y) = \frac{FalsePositiveInstances}{ActualNegativeInstances} = \frac{fp}{fp + tn} \tag{3.11}$$

While traditional confidence examines how many predicted positive/negative instances are actually positive/negative (the precision), CC is focused in how many actual positive/negative instances are predicted correctly (the recall). Thus, even if there are many more negative than positive instances in the data set ($tp+fn \ll fp+tn$), Equations 3.10 and 3.11 will not be affected by this imbalance. Consequently, rules with high CC will be the significant ones, regardless of whether they are discovered from balanced or imbalanced data sets.




(a) Classes are balanced (b) Classes are imbalanced (1:10)

Figure 2: Information gain from original entropy when a data set follows different class distributions. Compared with the contour lines in (a), those in (b) shift towards the top-left and bottom-right.

However, obtaining high CC rules is still insufficient for solving classification problems – it is necessary to ensure that the classes implied by those rules are not only of high confidence, but *more interesting* than their corresponding alternative classes. Therefore, we propose the *proportion* of one CC over that of all classes as our measure of how interesting the class is – what we call the *CC Proportion (CCP)*. The CCP of rule $X \to y$ is defined as:

$$CCP(X \to y) = \frac{CC(X \to y)}{CC(X \to y) + CC(X \to \neg y)} \tag{3.12}$$

A rule with high CCP means that, compared with its alternative class, the class this rule implies has higher CC, and consequently is more likely to occur together with this rule's antecedents regardless of the proportion of classes in the data set. Another benefit of taking this proportion is the ability to scale CCP between [0,1], which makes it possible to replace the traditional frequency term in entropy (the factor $p(j|t)$ in Equation 2.4) by CCP. Details of the CCP replacement in entropy is introduced in Section 4.

**3.2 Robustness of CCP** We now evaluate the robustness of CCP using ROC-based isometric plots proposed in Flach [10] and which are inherently independent of class and misclassification costs.

The 2D ROC space is spanned by false positive rate (x-axis) and the true positive rate (y-axis). The contours mark out the lines of constant value, of the splitting criterion,

conditioned on the imbalanced class ratio. Metrics which are robust to class imbalance should have similar contour plots for different class ratios.

In Figure 2, the contour plots of information gain are shown for class ratios of 1:1 and 1:10, respectively. It is clear, from the two figures, that when the class distributions become more imbalanced, the contours tend to be flatter and further away from the diagonal. Thus, given the same true positive rate and false positive rate, information gain for imbalanced data sets (Figure 2b) will be much lower than for balanced data sets (Figure 2a).

Following the model of relative impurity proposed Flach in [10], we now derive the definition for the CCP Impurity Measure. Equation 3.12 gives:

$$CCP(X \to y) = \frac{\frac{tp}{tp+fn}}{\frac{tp}{tp+fn} + \frac{fp}{fp+tn}} = \frac{tpr}{tpr + fpr} \tag{3.13}$$

where *tpr/fpr* represents true/false positive rate. For each node-split in tree construction, at least two paths will be generated, if one is $X \to y$, the other one will be $\neg X \to \neg y$ with CCP:

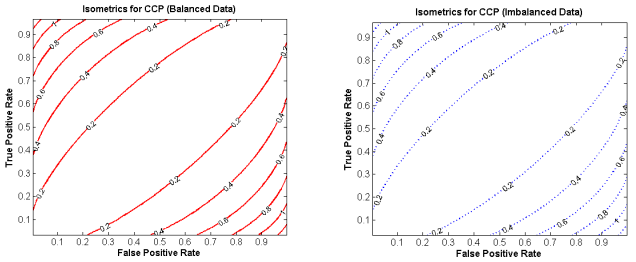
$$CCP(\neg X \to \neg y) = \frac{\frac{tn}{fp+tn}}{\frac{tn}{fp+tn} + \frac{fn}{tp+fn}} = \frac{1 - fpr}{2 - tpr - fpr} \tag{3.14}$$

The relative impurity for C4.5 proposed in [10] is:

$$\begin{aligned} InfoGain_{C4.5} =& Imp(tp + fn, fp + tn) \\ & - (tp + fp) * Imp(\frac{tp}{tp + fp}, \frac{fp}{tp + fp}) \\ & - (fn + tn) * Imp(\frac{fn}{fn + tn}, \frac{tn}{fn + tn}) \end{aligned} \tag{3.15}$$

where *Imp(p,n)*=-*p*log*p*-*n*log*n*. The first term in the right side represents the entropy of the node before splitting, while the sum of the second and third terms represents the entropy of the two subnodes after splitting. Take the second term $(tp + fp) * Imp(\frac{tp}{tp+fp}, \frac{fp}{tp+fp})$ as an example, the first frequency measure $\frac{tp}{tp+fp}$ is an alternative way of interpreting the confidence of rule $X \to y$; similarly, the second frequency measure $\frac{fp}{tp+fp}$ is equal to the confidence of rule $X \to \neg y$. We showed in Section 2.2 that both terms are inappropriate for imbalanced data learning.

To overcome the inherent weakness in traditional decision trees, we apply CCP into this impurity measure and thus rewrite the information gain definition in Equation 3.15 as the *CCP Impurity Measure*:




(a) Classes are balanced (b) Classes are imbalanced (1:10)

Figure 3: Information gain from CCP-embedded entropy when a data set follows different class distributions. No contour line shifts when data sets becomes imbalanced.

$$\begin{aligned} InfoGain_{CCP} =& Imp(tp + fn, fp + tn) \\ & - (tpr + fpr) * Imp(\frac{tpr}{tpr + fpr}, \frac{fpr}{tpr + fpr}) \\ & - (2 - tpr - fpr) * Imp(\frac{1 - tpr}{2 - tpr - fpr}, \frac{1 - fpr}{2 - tpr - fpr}) \end{aligned} \tag{3.16}$$

where *Imp(p,n)* is still "-*p*log*p*-*n*log*n*", while the original frequency term is replaced by CCP.

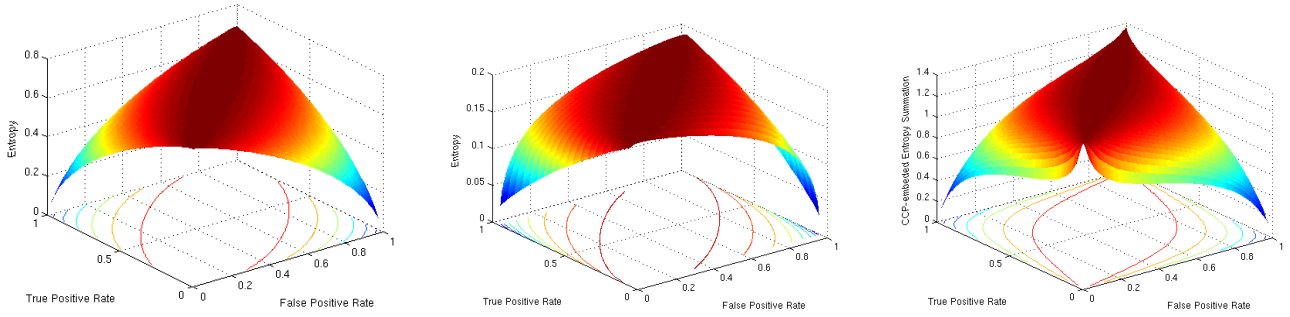
The new isometric plots, with the CCP replacement, are presented in Figure 3 (a,b). A comparison of the two figures tells that contour lines remain unchanged, demonstrating that CCP is unaffected by the changes in the class ratio.

**3.3 Properties of CCP** If all instances contained in a node belong to the same class, its entropy is minimized (zero). The entropy is maximized when a node contains equal number of elements from both classes.

By taking all possible combinations of elements in the confusion matrix (Table 2), we can plot the entropy surface as a function of *tpr* and *fpr* as shown in Figure 4. Entropy (Figure 4a) is the highest when *tpr* and *fpr* are equal, since "tpr = fpr" in subnodes is equivalent to elements in the subnodes being equally split between the two classes. On the other hand, the larger the difference between tpr and fpr, the purer the subnodes and the smaller their entropy. However, as stated in Section 3.2, when data sets are imbalanced, the pattern of traditional entropy will become *distorted* (Figure 4b).

Since CCP-embedded "entropy" is insensitive to class skewness, its will always exhibit a fixed pattern, and this pattern is the same as traditional entropy's balanced data situation. This can be formalized as follows:

By using the notations in the confusion matrix, the frequency term in traditional entropy is $p_{traditional} = \frac{tp}{tp+fp}$, while in CCP-based entropy it is $p_{CCP} = \frac{tpr}{tpr+fpr}$. When classes in a data set are evenly distributed, we have *tp+fn* =




(a) Traditional entropy on balanced data sets

(b) Traditional entropy on imbalanced data sets. (Positive:Negative = 1:10)

(c) CCP-embedded entropy on any data sets

Figure 4: The sum of subnodes' entropy after splitting. When a data set is imbalanced, the entropy surf (b) is "distored" from (a); but for CCP-embedded "entropy" (c), the surface is always the same independent of the imbalance in the data.

*fp+tn*, and by applying it in the definition of CCP we obtain:

$$p_{CCP} = \frac{tpr}{tpr + fpr} = \frac{\frac{tp}{tp+fn}}{\frac{tp}{tp+fn} + \frac{fp}{fp+tn}} = \frac{tp}{tp + fp}$$
$$= p_{traditional}$$

Thus when there are same number of instances in each class, the patterns of CCP-embedded entropy and traditional entropy will be the same. More importantly, this pattern is preserved for CCP-embedded entropy independent of the imbalance the data sets. This is confirmed in Figure 4c which is always similar to the pattern of Figure 4a regardless of the class distributions.

### 3.4 Hellinger Distance and its relationship with CCP

The divergence of two absolutely continuous distributions can be measured by *Hellinger distance* with respect to the parameter $\lambda$ [17, 11], in the form of:

$$d_H(P, Q) = \sqrt{\int_\Omega (\sqrt{P} - \sqrt{Q})^2 d\lambda}$$

In the Hellinger distance based decision tree (HDDT) technique [8], the distribution $P$ and $Q$ are assumed to be the normalized frequencies of feature values ("X" in our notation) across classes. The Hellinger distance is used to capture the propensity of a feature to separate the classes. In the tree-construction algorithm in HDDT, a feature is selected as a splitting attribute when it produces the largest Hellinger distance between the two classes. This distance is essentially captured in the differences in the relative frequencies of the attribute values for the two classes, respectively.

The following formula, derived in [8], relates HDDT with the true positive rate (tpr) and false positive rate (fpr).

(3.17)
$$Impurity_{HD} = \sqrt{(\sqrt{tpr} - \sqrt{fpr})^2 + (\sqrt{1 - tpr} - \sqrt{1 - fpr})^2}$$

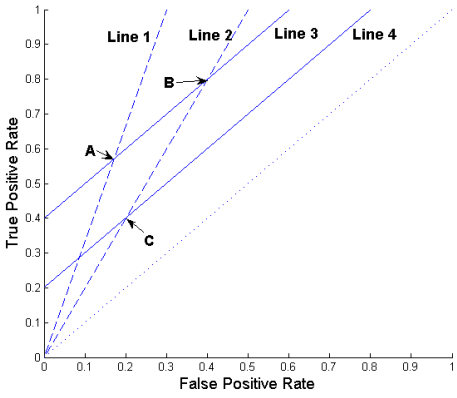



Figure 5: The attribute selection mechanisms of CCP and Hellinger distances. This example illustrates a complementary situation where, while Hellinger distance can only prioritizes B and C, CCP distinguishes only A and B.

This was also shown to be insensitive to class distributions in [8], since the only two variables in this formula are *tpr* and *fpr*, without the dominating class priors.

Like the Hellinger distance, CCP is also just based on *tpr* and *fpr* as shown in Equation 3.13. However, there is a significant difference between CCP and Hellinger distance. While Hellinger distance take the square root **difference** of *tpr* and *fpr* ($|\sqrt{tpr} - \sqrt{fpr}|$) as the divergence of one class distribution from the other, CCP takes the **proportion** of *tpr* and *fpr* as a measurement of interest. A graphical difference between the two measures is shown in Figure 5.

If we draw a straight line (Line 3) parallel to the diagonal in Figure 5, the segment length from origin to cross-point between Line 3 and the y-axis is $|\ tpr_o - fpr_o\ |$ ($tpr_o$ and $fpr_o$ can be the coordinates of any point in Line 3), is proportional to the Hellinger distance ($|\sqrt{tpr} - \sqrt{fpr}|$).

From this point of view, HDDT selects the point on those parallel lines with the longest segment. Therefore, in Figure 5, all the points in Line 3 have a larger Hellinger distance than those in Line 4; thus points in Line 3 will have higher priority in the selection of attributes. As $CCP = \frac{tpr}{tpr+fpr}$ can be rewritten as $tpr = \frac{CCP}{1-CCP} fpr$, CCP is proportional to the the slope of the line formed by the data point and the origin, and consequently favors the line with the highest slope. In Figure 5, the points in Line 1 are considered by CCP as better splitting attributes than those in Line 2.

By analyzing CCP and Hellinger distances in terms of lines in a *tpr versus fpr* reference frame, we note that CCP and Hellinger distance share a common problem. We give an example as follows. Suppose we have three points, A ($fpr_A$, $tpr_A$), B ($fpr_B$, $tpr_B$) and C ($fpr_C$, $tpr_C$), where A is one Line 1 and 3, B on Line 2 and 3, and C on Line 2 and 4 (shown in Figure 5). Then A and B are on the same line (Line 3) that is parallel to the diagonal (i.e. $\mid fpr_A - tpr_A \mid = \mid fpr_B - tpr_B \mid$), while B and C are on the same line (Line 2) passing through the origin (i.e. $\frac{tpr_B}{fpr_B} = \frac{tpr_C}{fpr_C}$). Hellinger distances will treat A and B as better splitting attributes than C, because as explained above all points in Line 3 has longer Hellinger distances than Line 4. By contrast, CCP will consider A has higher splitting priorities than both B and C, since all points in Line 1 obtains greater CCP than Line 2. However, on points in Line 3 such as A and B, Hellinger distance fails to distinguish them , since they will generate the same *tpr vs. fpr* difference. In this circumstance, HDDT may make an noneffective decision in attribute selection. This problem will become significant when the number of attributes is large, and many attributes have similar $\mid tpr - fpr \mid$ (or more precisely $\mid \sqrt{tpr} - \sqrt{fpr} \mid$) difference. The same problem occurs in the CCP measurement on testing points in Line 2 such as B against C.

Our solution to this problem is straightforward: when choosing the splitting attribute in decision tree construction, we select the one with the highest CCP by default, and if there are attributes that possess similar CCP values, we prioritize them on the basis of their Hellinger distances. Thus, in Figure 5, the priority of the three points will be *A>B>C*, since Point A has a greater CCP value than Points B and C, and Point B has higher Hellinger distance than Point C. Details of these attribute-selecting algorithms are in Section 4.

**3.5 Fisher's Exact Test** While CCP helps to select which branch of a tree are "good" to discriminate between classes, we also want to evaluate the statistical significance of each branch. This is done by the Fisher's exact test (FET). For a rule $X \rightarrow y$, the FET will find the probability of obtaining the contingency table where $X$ and $y$ are more positively associated, under the null hypothesis that $\{X, \neg X\}$ and $\{y, \neg y\}$ are independent [19]. The $p$ value of this rule is given by:

(3.18)
$$p([a,b;c,d]) = \sum_{i=0}^{min(b,c)} \frac{(a+b)!(c+d)!(a+c)!(b+d)!}{n!(a+i)!(b-i)!(c-i)!(d+i)!}$$

During implementation, the factorials in the *p*-value definition can be handled by expressing their values logarithmically.

A low *p* value means that the variable independence null hypothesis is rejected (no relationship between $X$ and $y$); in other words, there is a positive association between the upper-left cell in the contingency table (true positives) and the lower-right (true negatives). Therefore, given a threshold for the *p* value, we can find and keep the tree branches that are statistically significant (with lower *p* values), and discard those tree nodes that are not.

## 4 CCP-based decision trees (CCPDT)

In this section we provide details of the CCPDT algorithm. We modify the C4.5 splitting criterion based on entropy and replace the frequency term by CCP. Due to space limits, we omit the algorithms for CCP-embedded CART, but the approach is identical to C4.5 (in that the same factor is replaced with CCP).

**Algorithm 1** (*CCP-C4.5*) Creation of CCP-based C4.5

**Input:** Training Data: $TD$
**Output:** Decision Tree

**if** All instances are in the same class **then**
  Return decision tree with one node (root), labeled as the instances' class,
**else**
  // Find the best splitting attribute ($Attri$),
  $Attri$ = MaxCCPGain($TD$),
  Assign $Attri$ to the tree root (root = $Attri$),
  **for** each value $v_i$ of $Attri$ **do**
    Add a branch for $v_i$,
    **if** No instance is $v_i$ at attribute $Attri$ **then**
      Add a leaf to this branch.
    **else**
      Add a subtree $CCP - C4.5(TD_{v_i})$ to this branch,
    **end if**
  **end for**
**end if**

**4.1 Build tree** The original definition of entropy in decision trees is presented in Equation 2.4. As explained in Section 2.2, the factor $p(j|t)$ in Equation 2.4 is not a good criterion for learning from imbalanced data sets, so we replace it with CCP and define the CCP-embedded entropy as:

**Algorithm 2** (*MaxCCPGain*) Subroutine for discovering the attribute with the greatest information gain

**Input:** Training Data: $TD$
**Output:** The attribute to be split: $Attri$

1: Let *MaxHellinger*, *MaxInfoGain* and *Attri* to 0;
2: **for** Each attribute $A_j$ in $TD$ **do**
3: Calculate Hellinger distance: $A_j.Hellinger$,
4: Obtain the entropy before splitting: $A_j.oldEnt$,
5: Set the sum of sub-nodes' entropy $A_j.newEnt$ to 0,
6: **for** Each value $V_j^i$ of attribute $A_j$ **do**
7: // $|T_{x,y}|$ means the number of instance that have value $x$ and class $y$,
8: tpr = $\frac{T_{x=V_j^i,y=+}}{T_{x=V_j^i,y=+}+T_{x\neq V_j^i,y=+}}$,
9: fpr = $\frac{T_{x=V_j^i,y\neq+}}{T_{x=V_j^i,y\neq+}+T_{x\neq V_j^i,y\neq+}}$,
10: $A_j.newEnt$ += $T_{x=V_j^i} * (-\frac{tpr}{tpr+fpr}log\frac{fpr}{tpr+fpr} - \frac{fpr}{tpr+fpr}log\frac{fpr}{tpr+fpr})$,
11: **end for**
12: *CurrentInfoGain* = $A_j.oldEnt - A_j.newEnt$,
13: **if** *MaxInfoGain* $<$ *CurrentInfoGain* **then**
14: *Attri* = j,
15: *MaxInfoGain* = *CurrentInfoGain*,
16: *MaxHellinger* = $A_j$.*Hellinger*,
17: **else**
18: **if** *MaxHellinger* $<$ $A_j$.*Hellinger* AND *MaxInfoGain* $==$ *CurrentInfoGain* **then**
19: *Attri* = j,
20: *MaxHellinger* = $A_j$.*Hellinger*,
21: **end if**
22: **end if**
23: **end for**
24: Return *Attri*.

**Algorithm 3** (*Prune*) Pruning based on FET

**Input:** Unpruned decision tree $DT$, *p*-value threshold ($pVT$)
**Output:** Pruned $DT$

1: **for** Each leaf $Leaf_i$ **do**
2: **if** $Leaf_i.parent$ is not the Root of DT **then**
3: $Leaf_i.parent.pruneable = true$, // 'true' is default
4: $SetPruneable(Leaf_i.parent, pVT)$,
5: **end if**
6: **end for**
7: Obtain the root of $DT$,
8: **for** Each $child(i)$ of the root **do**
9: **if** child(i) is not a leaf **then**
10: **if** $child(i).pruneable == true$ **then**
11: Set $child(i)$ to be a leaf,
12: **else**
13: $PrunebyStatus(child(i))$.
14: **end if**
15: **end if**
16: **end for**

**Algorithm 4** (*SetPruneable*) Subroutine for setting pruneable status to each branch node by a bottom–up search

**Input:** A branch node $Node$, p-Value threshold ($pVT$)
**Output:** Pruneable status of this branch node

1: **for** each $child(i)$ if $Node$ **do**
2: **if** $child(i).pruneable == false$ **then**
3: Node.pruneable = false,
4: **end if**
5: **end for**
6: **if** Node.pruneable == true **then**
7: Calculate the *p* value of this node: $Node.pValue$,
8: **if** $Node.pValue < pVT$ **then**
9: Node.pruneable = false,
10: **end if**
11: **end if**
12: **if** $Node.parent$ is not the Root of the full tree **then**
13: Node.parent.pruneable = true // 'true' is default
14: SetPruneable(Node.parent, pVT),
15: **end if**

**Algorithm 5** (*PrunebyStatus*) Subroutine for pruning nodes by their pruneable status

**Input:** A branch represented by its top node $Node$
**Output:** Pruned branch

1: **if** Node.pruneable == true **then**
2: Set $Node$ as a leaf,
3: **else**
4: **for** Each $child(i)$ of Node **do**
5: **if** child(i) is not a leaf **then**
6: PrunebyStatus(child(i)),
7: **end if**
8: **end for**
9: **end if**

(4.19)
$$Entropy^{CCP}(t) = -\sum_j CCP(X \to y_j) log CCP(X \to y_j)$$

Then we can restate the conclusion made in Section 2.2 as: in CCP-based decision trees, $IG_{CCPDT}$ achieves the highest value when either $X \to y$ or $X \to \neg y$ has high CCP, and either $\neg X \to y$ or $\neg X \to \neg y$ has high CCP.

The process of creating CCP-based C4.5 (CCP-C4.5) is described in Algorithm 1. The major difference between CCP-C4.5 and C4.5 is the the way of selecting the candidate-splitting attribute (Line 5). The process of discovering the attribute with the highest information gain is presented in the subroutine Algorithm 2. In Algorithm 2, Line 4 obtains the entropy of an attribute before its splitting, Lines 6 – 11 obtain the new CCP-based entropy after the splitting of that attribute, and Lines 13 – 22 record the attribute with the highest information gain. In information gain comparisons of different attributes, Hellinger distance is used to select the attribute whenever InfoGain value of the two attributes are

equal (Lines 18–21), thus overcoming the inherent drawback of both Hellinger distances and CCP (Section 3.4).

In our decision tree model, we treat each branch node as the last antecedent of a rule. For example, if there are three branch nodes (BranchA, BranchB, and BranchC) from the root to a leaf LeafY, we assume that the following rule exists: $BranchA \wedge BranchB \wedge BranchC \rightarrow LeafY$. In Algorithm 2, we calculate the CCP for each branch node; using the preceding example, the CCP of BranchC is that of the previous rule, and the CCP of $BrachB$ is that of rule $BranchA \wedge BranchB \rightarrow LeafY$, etc. In this way, the attribute we select is guarenteed to be the one whose split can generate rules (paths in the tree) with the highest CCP.

**4.2 Prune tree** After the creation of decision tree, Fisher's exact test is applied on each branch node. A branch node will not be replaced by a leaf node if there is at least one significant descendant (a node with a lower *p* value than the threshold) under that branch.

Checking the significance of all descendants of an entire branch is an expensive operation. To perform a more efficient pruning, we designed a two-staged strategy as shown in Algorithm 3. The first stage is a bottom–up checking process from each leaf to the root. A node is marked "pruneable" if it and all of its descendants are non-significant. This process of checking the pruning status is done via Lines 1–6 in Algorithm 3, with subroutine Algorithm 4. In the beginning, all branch nodes are set to the default status of "pruneable" (Line 3 in Algorithm 3 and Line 13 in Algorithm 4). We check the significance of each node from leaves to the root. If any child of a node is "unpruneable" or the node itself represents a significant rule, this node will be reset from "pruneable" to "unpruneable". If the original unpruned tree is $n$ levels deep, and has $m$ leaves, the time complexity of this bottom–up checking process is $O(nm)$.

After checking for significance, we conduct the second pruning stage – a top-down pruning process performed according to the "pruneable" status of each node from root to leaves. A branch node is replaced by a leaf, if its "pruneable" status is "true" (Line 10 in Algorithm 3 and Line 1 in Algorithm 5).

Again, if the original unpruned tree is $n$ levels deep and has $m$ leaves, the time complexity of the top–down pruning process is $O(nm)$. Thus, the total time complexity of our pruning algorithm is $O(n^2)$.

This two-stage pruning strategy guarantees both the completeness and the correctness of the pruned tree. The first stage checks the significance of each possible rule (path through the tree), and ensures that each significant rule is "unpruneable", and thus complete; the second stage prunes all insignificant rules, so that the paths in the pruned tree are all correct.

## 5 Sampling Methods

Another mechanism of overcoming the imbalance class distribution is to synthetically delete or add training instances and thus balance the class distribution. To achieve this goal, various sampling techniques have been proposed to either remove instances from the majority class (aka under-sampling) or introduce new instances to the minority class (aka over-sampling).

We consider a wrapper framework that uses a combination of random under-sampling and SMOTE [5, 6]. The wrapper first determines the percentage of under-sampling that will result in an improvement in AUC over the decision tree trained on the original data. Then the number of instances in majority class is under-sampled to the stage where the AUC does not improve any more, the wrapper explores the appropriate level of SMOTE. Then taking the level of under-sampling into account, SMOTE introduces new synthetic examples to the minority class continuously until the AUC is optimized again. We point the reader to [6] for more details on the wrapper framework. The performance of decision trees trained on the data sets optimized by this wrapper framework is evaluated against CCP-based decision trees in experiments.

## 6 Experiments

In our experiments, we compared CCPDT with C4.5 [15], CART[2], HDDT [8] and SPARCCC [19] on binary class data sets. These comparisons demonstrate not only the efficiency of their splitting criteria, but the performance of their pruning strategies.

Weka's C4.5 and CART implementations [20] were employed in our experiments, based on which we implemented CCP-C4.5, CCP-CART, HDDT and SPARCCC [2], so that we can normalize the effects of different versions of the implementations.

All experiments were carried out using $5 \times 2$ folds cross-validation, and the final results were averaged over the five runs. We first compare purely on splitting criteria without applying any pruning techniques, and then comparisons between pruning methods on various decision trees are presented. Finally, we compare CCP-based decision trees with state-of-the-art sampling methods.

**6.1 Comparisons on splitting criteria** The binary-class data sets were mostly obtained from [8] (Table 3) which were pre-discretized. They include a number of real-world data sets from the UCI repository and other sources. "Estate" contains electrotopological state descriptors for a series of compounds from the US National Cancer Institute's Yeast Anti-Cancer drug screen. "Ism" ([5]) is highly unbalanced

[2] Implementation source code and data sets used in the experiments can be obtained from `http://www.cs.usyd.edu.au/~weiliu`

Table 3: Information about imbalanced binary-class data sets. The percentages listed in the last column is the proportion of the minor class in each data set.

| Data Sets | Instances | Attributes | MinClass % |
|---|---|---|---|
| Boundary | 3505 | 175 | 3.5% |
| Breast | 569 | 30 | 37.3% |
| Cam | 28374 | 132 | 5.0% |
| Covtype | 38500 | 10 | 7.1% |
| Estate | 5322 | 12 | 12.0% |
| Fourclass | 862 | 2 | 35.6% |
| German | 1000 | 24 | 30.0% |
| Ism | 11180 | 6 | 2.3% |
| Letter | 20000 | 16 | 3.9% |
| Oil | 937 | 50 | 4.4% |
| Page | 5473 | 10 | 10.2% |
| Pendigits | 10992 | 16 | 10.4% |
| Phoneme | 2700 | 5 | 29.3% |
| PhosS | 11411 | 481 | 5.4% |
| Pima | 768 | 8 | 34.9% |
| Satimage | 6430 | 37 | 9.7% |
| Segment | 2310 | 20 | 14.3% |
| Splice | 1000 | 60 | 48.3% |
| SVMguide | 3089 | 4 | 35.3% |

and records information on calcification in a mammogram. "Oil" contains information about oil spills; it is relatively small and very noisy [12]. "Phoneme" originates from the ELENA project and is used to distinguish between nasal and oral sounds. "Boundary", "Cam", and "PhosS" are biological data sets from [16]. "FourClass", "German", "Splice", and "SVMGuide" are available from LIBSVM [3]. The remaining data sets originate from the UCI repository [1]. Some were originally multi-class data sets, and we converted them into two-class problems by keeping the smallest class as the minority and the rest as the majority. The exception was "Letter", for which each vowel became a member of the minority class, against the consonants as the majority class.

As accuracy is considered a poor performance measure for imbalanced data sets, we used the area under ROC curve (AUC) [18] to estimate the performance of each classifier.

In our imbalanced data sets learning experiments, we only wanted to compare the effects of different splitting criteria; thus, the decision trees (C4.5, CCP-C4.5, CART, CCP-CART, and HDDT) were **unpruned**, and we used Laplace smoothing on leaves. Since SPARCCC has been proved more efficient in imbalanced data learning than CBA [19], we excluded CBA and included only SPARCCC. Table 4 lists the "Area Under ROC (*AUC*)" value on each data set for each classifier, followed by the ranking of these classifiers (presented in parentheses) on each data set.

We used the Friedman test on AUCs at 95% confidence level to compare among different classifiers [9]. In all experiments, we chose the best performance classifier as the "Base" classifier. If the "Base" classifier is statistically

Table 4: Splitting criteria comparisons on imbalanced data sets where all trees are unpruned. "Fr.T" is short for Friedman test. The classifier with a ✓ sign in the Friedman test is statistically outperformed by the "Base" classifier. The first two Friedman tests illustrate that CCP-C4.5 and CCP-CART are significantly better than C4.5 and CART, respectively. The third Friedman test confirms that CCP-based decision trees is significantly better than SPARCCC.

| Data Sets | Area Under ROC | | | | | |
|---|---|---|---|---|---|---|
| | C4.5 | CCP-C4.5 | CART | CCP-CART | HDDT | SPARCCC |
| Boundary | 0.533(4) | 0.595(2) | 0.529(5) | 0.628(1) | 0.594(3) | 0.510(6) |
| Breast | 0.919(5) | 0.955(2) | 0.927(4) | 0.958(1) | 0.952(3) | 0.863(6) |
| Cam | 0.707(3) | 0.791(1) | 0.702(4) | 0.772(2) | 0.680(5) | 0.636(6) |
| Covtype | 0.928(4) | 0.982(1) | 0.909(5) | 0.979(3) | 0.982(1) | 0.750(6) |
| Estate | 0.601(1) | 0.594(2) | 0.582(4) | 0.592(3) | 0.580(5) | 0.507(6) |
| Fourclass | 0.955(5) | 0.971(3) | 0.979(1) | 0.969(4) | 0.975(2) | 0.711(6) |
| German | 0.631(4) | 0.699(1) | 0.629(5) | 0.691(3) | 0.692(2) | 0.553(6) |
| Ism | 0.805(4) | 0.901(3) | 0.802(5) | 0.905(2) | 0.990(1) | 0.777(6) |
| Letter | 0.972(3) | 0.991(1) | 0.968(4) | 0.990(2) | 0.912(5) | 0.872(6) |
| Oil | 0.641(6) | 0.825(1) | 0.649(5) | 0.802(2) | 0.799(3) | 0.680(4) |
| Page | 0.906(5) | 0.979(1) | 0.918(4) | 0.978(2) | 0.974(3) | 0.781(6) |
| Pendigits | 0.966(4) | 0.990(2) | 0.966(4) | 0.990(2) | 0.992(1) | 0.804(6) |
| Phoneme | 0.824(5) | 0.872(3) | 0.835(4) | 0.876(2) | 0.906(1) | 0.517(6) |
| PhosS | 0.543(4) | 0.691(1) | 0.543(4) | 0.673(3) | 0.677(2) | 0.502(6) |
| Pima | 0.702(4) | 0.757(3) | 0.696(5) | 0.758(2) | 0.760(1) | 0.519(6) |
| Satimage | 0.774(4) | 0.916(1) | 0.730(5) | 0.915(2) | 0.911(3) | 0.706(6) |
| Segment | 0.981(5) | 0.987(1) | 0.982(4) | 0.987(1) | 0.984(3) | 0.887(6) |
| Splice | 0.913(4) | 0.952(1) | 0.894(5) | 0.926(3) | 0.950(2) | 0.781(6) |
| SVMguide | 0.976(4) | 0.989(1) | 0.974(5) | 0.989(1) | 0.989(1) | 0.924(6) |
| Avg. Rank | 3.95 | 1.6 | 4.15 | 2.09 | 2.4 | 5.65 |
| ***Fr.T*** (*C4.5*) | ✓9.6E-5 | Base | | | | |
| ***Fr.T*** (*CART*) | | | ✓9.6E-5 | Base | | |
| ***Fr.T*** (*Other*) | | Base | | | 0.0896 | ✓1.3E-5 |

significantly better than another classifier in comparison (i.e. the value of Friedman test is less than 0.05), we put a "✓" sign on the respective classifier (shown in Table 4).

The comparisons revealed that even though SPARCCC performs better than CBA [19], its overall results are far less robust than those from decision trees. It might be possible to obtain better SPARCCC results by repeatedly modifying their parameters and attempting to identify the optimized parameters, but the manual parameter configuration itself is a shortcoming for SPARCCC.

Because we are interested in the replacement of original factor $p(j|t)$ in Equation 2.4 by CCP, three separate Friedman tests were carried out: the first two between conventional decision trees (C4.5/CART) and our proposed decision trees (CCP-C4.5/CCP-CART), and the third on all the other classifiers. A considerable AUC increase from traditional to CCP-based decision trees was observed, and statistically confirmed by the first two Friedman tests: these small $p$ values of 9.6E-5 meant that we could confidently reject the hypothesis that the "Base" classifier showed no significant differences from current classifier. Even though CCP-C4.5

Table 5: Pruning strategy comparisons on AUC. "Err.ESt" is short for error estimation. The AUC of C4.5 and CCP-C4.5 pruned by FET are significantly better than those pruned by error estimation.

| Data set | C4.5 | | CCP-C4.5 | |
|---|---|---|---|---|
| | Err.Est | FET | Err.Est | FET |
| Boundary | 0.501(3) | 0.560(2) | 0.500(4) | 0.613(1) |
| Breast | 0.954(1) | 0.951(3) | 0.953(2) | 0.951(3) |
| Cam | 0.545(3) | 0.747(2) | 0.513(4) | 0.755(1) |
| Covtype | 0.977(4) | 0.979(2) | 0.979(2) | 0.980(1) |
| Estate | 0.505(3) | 0.539(2) | 0.505(3) | 0.595(1) |
| Fourclass | 0.964(3) | 0.961(4) | 0.965(2) | 0.969(1) |
| German | 0.708(3) | 0.715(2) | 0.706(4) | 0.719(1) |
| Ism | 0.870(3) | 0.891(1) | 0.848(4) | 0.887(2) |
| Letter | 0.985(3) | 0.993(1) | 0.982(4) | 0.989(2) |
| Oil | 0.776(4) | 0.791(3) | 0.812(2) | 0.824(1) |
| Page | 0.967(4) | 0.975(1) | 0.969(3) | 0.973(2) |
| Pendigits | 0.984(4) | 0.986(3) | 0.988(2) | 0.989(1) |
| Phoneme | 0.856(4) | 0.860(2) | 0.858(3) | 0.868(1) |
| PhosS | 0.694(1) | 0.649(3) | 0.595(4) | 0.688(2) |
| Pima | 0.751(4) | 0.760(1) | 0.758(2) | 0.755(3) |
| Satimage | 0.897(4) | 0.912(2) | 0.907(3) | 0.917(1) |
| Segment | 0.987(3) | 0.987(3) | 0.988(1) | 0.988(1) |
| Splice | 0.954(1) | 0.951(4) | 0.954(1) | 0.953(3) |
| SVMguide | 0.982(4) | 0.985(1) | 0.984(2) | 0.984(2) |
| Avg.Rank | 3.0 | 2.15 | 2.65 | 1.55 |
| ***Fr.T*** (*C4.5*) | ✓ 0.0184 | Base | | |
| ***Fr.T*** (*CCP*) | | | ✓ 0.0076 | Base |

Table 6: Pruning strategy comparisons on number of leaves. The leaves on trees of C4.5 and CCP-C4.5 pruned by FET are not significantly more than those pruned by error estimation.

| Data set | C4.5 | | CCP-C4.5 | |
|---|---|---|---|---|
| | Err.Est | FET | Err.Est | FET |
| Boundary | 2.7(2) | 33.9(3) | 1.2(1) | 87.6(4) |
| Breast | 7.7(4) | 6.4(2) | 7.0(3) | 6.2(1) |
| Cam | 54.5(2) | 445.9(3) | 7.9(1) | 664.6(4) |
| Covtype | 156.0(1) | 175.3(3) | 166.9(2) | 189.0(4) |
| Estate | 2.2(2) | 5.2(4) | 2.0(1) | 4.6(3) |
| Fourclass | 13.9(3) | 13.0(1) | 14.1(4) | 13.3(2) |
| German | 41.3(3) | 40.0(1) | 47.1(4) | 40.1(2) |
| Ism | 19.8(1) | 26.3(3) | 21.9(2) | 31.0(4) |
| Letter | 30.2(1) | 45.1(3) | 34.3(2) | 47.4(4) |
| Oil | 8.0(1) | 10.7(3) | 8.5(2) | 12.0(4) |
| Page | 29.4(2) | 28.8(1) | 32.6(3) | 34.0(4) |
| Pendigits | 35.1(3) | 37.8(4) | 31.5(1) | 32.8(2) |
| Phoneme | 32.6(4) | 29.7(2) | 30.1(3) | 27.0(1) |
| PhosS | 68.2(2) | 221.5(3) | 37.7(1) | 311.4(4) |
| Pima | 15.5(4) | 12.7(2) | 13.3(3) | 11.8(1) |
| Satimage | 83.7(1) | 107.8(3) | 94.6(2) | 119.2(4) |
| Segment | 8.3(3) | 8.4(4) | 6.4(1) | 7.2(2) |
| Splice | 21.3(3) | 18.0(1) | 22.6(4) | 18.4(2) |
| SVMguide | 14.5(3) | 12.3(2) | 15.3(4) | 11.9(1) |
| Avg.Rank | 2.3 | 2.45 | 2.25 | 2.7 |
| ***Fr.T*** (*C4.5*) | Base | 0.4913 | | |
| ***Fr.T*** (*CCP*) | | | Base | 0.2513 |

was not statistically better than HDDT, the strategy to combine CCP and HDDT (analyzed in Section 3.4) provided an improvement on AUC with higher than 91% confidence.

**6.2 Comparison of Pruning Strategies** In this section, we compared FET-based pruning with the pruning based on error estimation as originally proposed in C4.5 [15]. We reuse the data sets from previous subsection, and apply error-based pruning and FET-based pruning separately on the trees built by C4.5 and CCP-C4.5, where the confidence level of FET was set to 99% (i.e. the p-Value threshold is set to 0.01). Note that the tree constructions differences between C4.5 and CCP-C4.5 are out of scope in this subsection; we carried out separate Friedman test on C4.5 and CCP-C4.5 respectively and only compare the different pruning strategies. HDDT was excluded from this comparison since it provides no separate pruning strategies.

Table 5 and 6 present the performance of the two pairs of pruned trees. The numbers of leaves in Table 6 are not integers because they are the average values of $5 \times 2$ – fold cross validations. Statistically, the tree of C4.5 pruned by FET significantly outperformed the same tree pruned by error estimation (Table 5) without retaining significantly more leaves (Table 6). The same pattern is found in CCP-C4.5 trees, where FET-based pruning sacrificed *insignificant* more numbers of leaves to obtain a *significantly* larger AUC. This phenomenon proves the completeness of the FET pruned trees, since error estimation inappropriately cuts significant number of trees paths, and hence always has smaller number of leaves and lower AUC values.

**6.3 Comparisons with Sampling Techniques** We now compare CCP-based decision trees against sampling based methods discussed in Section 5. Note that the wrapper is optimized on training sets using $5 \times 2$ cross validation to determine the sampling levels. The algorithm was then evaluated on the corresponding $5 \times 2$ cross validated testing set.

The performances of three pairs of decision trees are shown in Table 7. The first pair "Original" has no modification on either data or decision tree algorithms. The second pair "Sampling based" uses the wrapper to re-sample the training data which is then used by original decision tree algorithms to build the classifier; and "CCP based" shows the performance of CCP-based decision trees learned from original data. Friedman test on the AUC values shows that, although the "wrapped" data can help to improve the performance of original decision trees, using CCP-based algorithms can obtain statistically better classifiers directly trained on the original data.

## 7 Conclusion and future work

We address the problem of designing a decision tree algorithm for classification which is robust against class imbalance in the data. We first explain why traditional decision tree measures, like information gain, are sensitive to class imbalance. We do that by expressing information gain in terms of the confidence of a rule. Information gain, like confidence, is biased towards the majority class. Having identified the cause of the problem, we propose a new measure, Class Confidence Proportion (CCP). Using both theoretical

Table 7: Performances comparisons on AUC generated by original, sampling and CCP-based techniques. "W+CART/W+C4.5" means applying wrappers to sample training data before it's learned by CART/C4.5. In this table, CCP-based CART decision tree is significantly better than all trees in "Original" and "Sampling based" categories.

| Data set | Original | | Sampling based | | CCP based | |
|---|---|---|---|---|---|---|
| | CART | C4.5 | W+CART | W+C4.5 | CCP-CART | CCP-C4.5 |
| Boundary | 0.514(5) | 0.501(6) | 0.582(4) | 0.616(2) | 0.631(1) | 0.613(3) |
| Breast | 0.928(6) | 0.954(2) | 0.955(1) | 0.953(4) | 0.954(2) | 0.951(5) |
| Cam | 0.701(3) | 0.545(6) | 0.660(5) | 0.676(4) | 0.777(1) | 0.755(2) |
| Covtype | 0.910(6) | 0.977(4) | 0.974(5) | 0.980(2) | 0.981(1) | 0.980(2) |
| Estate | 0.583(3) | 0.505(6) | 0.560(5) | 0.580(4) | 0.598(1) | 0.595(2) |
| Fourclass | 0.978(1) | 0.964(5) | 0.943(6) | 0.965(4) | 0.971(2) | 0.969(3 |
| German | 0.630(6) | 0.708(2) | 0.668(5) | 0.690(4) | 0.691(3) | 0.719(1) |
| Ism | 0.799(6) | 0.870(5) | 0.905(3) | 0.909(1) | 0.906(2) | 0.887(4) |
| Letter | 0.964(6) | 0.985(4) | 0.977(5) | 0.989(1) | 0.989(1) | 0.989(1) |
| Oil | 0.666(6) | 0.776(5) | 0.806(3) | 0.789(4) | 0.822(2) | 0.824(1) |
| Page | 0.920(6) | 0.967(5) | 0.970(4) | 0.978(1) | 0.978(1) | 0.973(3) |
| Pendigits | 0.963(6) | 0.984(4) | 0.982(5) | 0.987(3) | 0.990(1) | 0.989(2) |
| Phoneme | 0.838(6) | 0.856(5) | 0.890(2) | 0.894(1) | 0.871(3) | 0.868(4) |
| PhosS | 0.542(6) | 0.694(1) | 0.665(5) | 0.670(4) | 0.676(3) | 0.688(2) |
| Pima | 0.695(6) | 0.751(4) | 0.742(5) | 0.755(2) | 0.760(1) | 0.755(2) |
| Satimage | 0.736(6) | 0.897(4) | 0.887(5) | 0.904(3) | 0.914(2) | 0.917(1) |
| Segment | 0.980(5) | 0.987(2) | 0.980(5) | 0.982(4) | 0.987(2) | 0.988(1) |
| Splice | 0.885(5) | 0.954(1) | 0.829(6) | 0.942(3) | 0.940(4) | 0.953(2) |
| SVMguide | 0.973(6) | 0.982(5) | 0.985(3) | 0.987(2) | 0.989(1) | 0.984(4) |
| Avg.Rank | 5.05 | 3.85 | 4.15 | 2.7 | 1.75 | 2.3 |
| ***Fr.T*** | ✓ 9.6E-5 | ✓ 0.0076 | ✓ 5.8E-4 | ✓ 0.0076 | Base | 0.3173 |

and geometric arguments we show that CCP is insensitive to class distribution. We then embed CCP in information gain and use the improvised measure to construct the decision tree. Using a wide array of experiments we demonstrate the effectiveness of CCP when the data sets are imbalanced. We also propose the use of Fisher exact test as a method for pruning the decision tree. Besides improving the accuracy, an added benefit of the Fisher exact test is that all the rules found are statistically significant.

## Acknowledgements

The first author of this paper acknowledges the financial support of the Capital Markets CRC.

## References

[1] A. Asuncion and D.J. Newman. UCI Machine Learning Repository, 2007.

[2] L. Breiman. *Classification and regression trees*. Chapman & Hall/CRC, 1984.

[3] C.C. Chang and C.J. Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[4] N.V. Chawla. C4. 5 and imbalanced data sets: investigating the effect of sampling method, probabilistic estimate, and decision tree structure. 2003.

[5] N.V. Chawla, K.W. Bowyer, L.O. Hall, and W.P. Kegelmeyer. SMOTE: synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, 16(1):321–357, 2002.

[6] N.V. Chawla, D.A. Cieslak, L.O. Hall, and A. Joshi. Automatically countering imbalance and its empirical relationship to cost. *Data Mining and Knowledge Discovery*, 17(2):225–252, 2008.

[7] N.V. Chawla, A. Lazarevic, L.O. Hall, and K.W. Bowyer. SMOTEBoost: Improving prediction of the minority class in boosting. *Lecture notes in computer science*, pages 107–119, 2003.

[8] D.A. Cieslak and N.V. Chawla. Learning Decision Trees for Unbalanced Data. In *Proceedings of the 2008 European Conference on Machine Learning and Knowledge Discovery in Databases-Part I*, pages 241–256. Springer-Verlag Berlin, Heidelberg, 2008.

[9] J. Demšar. Statistical comparisons of classifiers over multiple data sets. *The Journal of Machine Learning Research*, 7:30, 2006.

[10] P.A. Flach. The geometry of ROC space: understanding machine learning metrics through ROC isometrics. In *Proceedings of the Twentieth International Conference on Machine Learning*, pages 194–201, 2003.

[11] T. Kailath. The divergence and Bhattacharyya distance measures in signal selection. *IEEE Transactions on Communication Technology*, 15(1):52–60, 1967.

[12] M. Kubat, R.C. Holte, and S. Matwin. Machine Learning for the Detection of Oil Spills in Satellite Radar Images. *Machine Learning*, 30(2-3):195–215, 1998.

[13] W. Li, J. Han, and J. Pei. CMAR: Accurate and Efficient Classification Based on Multiple Class-Association Rules. In *Proceedings of the 2001 IEEE International Conference on Data Mining*, pages 369–376. IEEE Computer Society Washington, DC, USA, 2001.

[14] B. Liu, W. Hsu, Y. Ma, A.A. Freitas, and J. Li. Integrating Classification and Association Rule Mining. *IEEE Transactions on Knowledge and Data Engineering*, 18:460–471.

[15] J.R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers Inc. San Francisco, CA, USA, 1993.

[16] P. Radivojac, N.V. Chawla, A.K. Dunker, and Z. Obradovic. Classification and knowledge discovery in protein databases. *Journal of Biomedical Informatics*, 37(4):224–239, 2004.

[17] C.R. Rao. *A review of canonical coordinates and an alternative to correspondence analysis using Hellinger distance*. Institut d'Estadística de Catalunya, 1995.

[18] J.A. Swets. *Signal detection theory and ROC analysis in psychology and diagnostics*. Lawrence Erlbaum Associates, 1996.

[19] F. Verhein and S. Chawla. Using Significant, Positively Associated and Relatively Class Correlated Rules for Associative Classification of Imbalanced Datasets. In *Seventh IEEE International Conference on Data Mining, 2007.*, pages 679–684, 2007.

[20] I.H. Witten and E. Frank. Data mining: practical machine learning tools and techniques with Java implementations. *ACM SIGMOD Record*, 31(1):76–77, 2002.