

LAZY LEARNER ON DECISION TREE FOR RANKING

YUHONG YAN

*Institute for Information Technology,
National Research of Council of Canada,
Fredericton, NB E3B 5X9, Canada
yuhong.yan@nrc.gc.ca*

HAN LIANG*

*Faculty of Computer Science,
University of New Brunswick,
Fredericton, NB E3B 5A3, Canada
han.liang@unb.ca*

Received (9 March 2007)

Revised (6 April 2007)

This paper aims to improve probability-based ranking (e.g.AUC) under decision-tree paradigm. We observe the fact that probability-based ranking is to sort samples in terms of their class probabilities. Therefore, ranking is a relative evaluation metric among those samples. This motivates us to use a lazy learner to explicitly yield a set of unique class probabilities for a testing sample based on its similarities to the training samples within its neighborhood. We embed lazy learners at the leaves of a decision tree to give class probability assignments. This results in the first model, named *Lazy Distance-based Tree* (LDTree). Then we further improve this model by continuing to grow the tree for the second time, and call the resulting model *Eager Distance-based Tree* (EDTree). In addition to the benefits of lazy learning, EDTree also takes advantage of the finer resolution of a large tree structure. We compare our models with C4.5, C4.4 and their variants in AUC on a large suite of UCI sample sets. The improvement shows that our method follows a new path that leads to better ranking performance.

Keywords: Classification; Probability-based Ranking; Lazy Learner.

1. Introduction

Traditionally, classification accuracy has been used as the main evaluation criterion for the performances of learning models in the areas of machine learning and data mining. However, in some decision-making scenarios, probability-based ranking is more desirable than categorical classification. For instance, university scholarship granting is based on the rank of top $x\%$ students in terms of their different likelihoods of winning it, instead of simply classifying the applicants into *qualified* or

*The author is a visiting worker at NRC-IIT.

non-qualified categories.

Classic decision tree, such as C4.5 [16] and ID3 [15], has been observed to yield poor class probability estimation [14]. Thus, learning a decision tree with accurate class probability estimation, called *Probability Estimation Tree* (PET), has recently attracted a great deal of attention. The pertinent C4.4 improves class probability estimation by generating a larger tree than C4.5. However, large trees have the disadvantages of over-fitting training sample sets and high variance in error rate functions.

What we observe is that, although improving the accuracy of class probability estimation is a straightforward way to improve the ranking quality, we still can not expect it because the true class distributions are often unknown a priori for many real-world problems. However, since ranking is indeed about the relative position of a sample among others, we propose to focus on improving the capacity of decision tree to discriminate samples. In this paper, we propose to use a lazy learner to generate a set of unique class probabilities for a testing sample based on its extent of similarity to other training samples within its neighborhood. In detail, we deploy lazy learners at the leaves of a decision tree to produce distance-based probability assignments. Thus, the class probability estimation of a testing sample represents its relative position to the training samples at the leaf into which the testing sample falls. This results in our first model, called ***Lazy Distance-based Tree*** (LDTree). Then, we further make our progress by expanding a decision tree twice. Specially speaking, in the first phase, a typical decision tree, such as C4.5, is developed and lazy learners are used to calculate weights for leaf training samples. After that, we continue to grow each tree leaf until either one of its posterior nodes is pure (all leaf samples belong to one class) or there are no more attributes to split. The class probability estimation for a testing sample is a normalization of the weights of training samples at the leaf where the testing sample falls, in terms of their class values. The resulting model benefits from both a large tree structure that has a good resolution capacity and the learned weights that smooth the class probabilities to reduce variance. We name the new model ***Eager Distance-based Tree*** (EDTree). We systematically carry out a large-scale empirical comparison of our models with several popular PETs by examining their learning performances in terms of probability-based ranking measured by AUC on a large suite of UCI sample sets [2]. These PETs include C4.5, C4.4, CFTs and tree variants with the *state-of-the-art* smoothing methods (e.g. *Laplace* correction and *m-Branch*) applied. All of them are representative of single-tree PETs. We also compare LDTree with bagging used with other ensemble PETs. The improvement on results show that our method follows a new path that leads to the better ranking performance of decision tree.

The rest of the paper is structured as follows: Section 2 presents the current methods applied to decision tree for better probability-based ranking. Section 3 describes our new models. Section 4 introduces the experiment setup and analyzes the empirical results. We conclude and outline the future work in Section 5.

2. Related Work

2.1. Probability Estimation with Decision Trees

A trained decision tree can be easily adapted to be a probability estimator by using the absolute class frequencies at leaves. For instance, if a leaf has a set of absolute class frequencies n_1, n_2, \dots, n_k , the estimated probability for each class membership at this leaf is represented as

$$\hat{p}(c_i) = n_i / \sum_{j=1}^k n_j. \quad (1)$$

But this estimation is poor for two reasons [14]: uniform class probability assignment (assign one priori class probability $\hat{p}(c_i)$ to any testing sample that falls into the same leaf) and tree pruning (traditional tree inductive algorithms may remove some branches that contribute little to classification accuracy but much more to the ranking performance). With the goal of better ranking performance, Provost and Domingos [13] introduced two methods to improve the class probability estimation of decision tree. First, by using *Laplace* correction at leaves, class probabilities can be systematically smoothed towards the priori class distribution. Second, by turning off pruning and collapsing mechanisms, decision tree can keep some branches that are crucial for accurate class probability estimation. The final model is called C4.4. However, C4.4 still has two contradictions that are:

- Turning off pruning may result in a fairly large tree. Therefore, within some leaves that contain only few training samples, the class probability estimation is unreliable.
- A large tree usually over-fits training sample sets, and the yielded class probabilities could be still doubtful even using the *Laplace* correction.

Besides, values of class probabilities can easily repeat, which may decrease the ranking quality as well.

Some researchers have noticed that the information used to estimate the class probabilities for a testing sample should not be limited to the leaf where the testing sample falls. Ling and Yan [11] proposed the *Confusion Factor Tree* (CFT) by using a factor s ($s < 1$) to denote the amount of “confusion” among the children of the node. It can be regarded as the probability of errors that alter the attribute value. Thus when a testing example is classified by a CFT, it has a small probability s of going down to other branches. Therefore, CFT produces the class probabilities for a testing sample \mathbf{e} via the class probabilities from all other leaves as well as the leaf into which \mathbf{e} falls. The contribution of each leaf is determined by the number of unequal parent attribute values (parent attributes of a leaf are defined as the attributes on the path from this leaf to the root) that the leaf has, compared with \mathbf{e} . The final class probabilities for \mathbf{e} are the weighted averages of the contributions

from all of the leaves in a tree, as Eq. (2) depicts.

$$\hat{p}(c_i|\mathbf{e}) = \frac{\sum \hat{p}_{L_j}(c_i) \times s^q}{\sum s^q}, \quad (2)$$

where $\hat{p}_{L_j}(c_i)$ is the class probability at leaf j calculated as in Eq. (1), and q is the number of unequal attribute values. Although CFT produces distinct class probabilities for the testing samples at the same leaf, choosing the optimal confusion factor could still be a difficult task. Furthermore, CFT has to go through the whole tree to compute the contribution of each leaf. Consequently, the time complexity of CFT tends to be relatively high.

Ferri *et al.* [6] introduced a smoothing method, called m -Branch. M -Branch is a recursive, root-to-leaf extension of the m probability estimation, in which, for each leaf, the class probabilities are generated by propagating the class probabilities of each of its parent nodes from the root down to itself. Eq. (3) is the formal expression of m -Branch method:

$$\hat{p}_{child}(c_i|\mathbf{e}) = \frac{n_i + m * \hat{p}_{parent}(c_i|\mathbf{e})}{\sum_{j=1}^k n_j + m}, \quad (3)$$

where \hat{p}_{parent} is the class probability smoothed from the root to its direct parent node, k represents the number of class values and parameter m is adjusted by the depth and cardinality (the number of training samples associated with the node) of the current node. Although the authors have noticed that the information from other paths should be utilized, they still assign the same class probabilities to the testing samples at the same leaf. The ranks of these samples with equal class probabilities are generated randomly and thus the AUC value tends to decrease.

Bootstrap Aggregating (bagging), an ensemble approach to aggregate the class probabilities from a suite of base learners, is used on decision tree to overcome the instability of its class probability estimation [9]. This method creates an ensemble of trees from a given training sample set. Each tree is generated with a different bootstrap replicate of the original set. The final output is formed by use of a plurality vote among those trees. Recently, bagging has been applied to improve the ranking performance of decision tree [1][13]. One drawback of using bagging is that the generated results are not easily comprehended.

Boosting was introduced by Schapire [17] as an approach for scaling up the accuracy of a learning model. *Adaptive Boosting* (AdaBoost) is an improvement of boosting proposed by Freund and Schapire [7]. Like bagging, AdaBoost also generates a set of base learners and adopts the voting mechanism among them. However, AdaBoost also changes the weights of the training samples provided as input to every learner at each iteration. It focuses on those training samples that have been misclassified in the previous iteration and therefore minimizes the training errors. In addition, AdaBoost generates base learners sequentially but bagging generates them in parallel. AdaBoost has been proved to be able to improve classification of unstable learning models, such as decision tree.

Su and Zhang [19] proposed to understand decision tree from a probabilistic perspective. They use C4.5 without pruning and the product rule in probability theory to calculate class probabilities, as shown in Eq. (4).

$$\hat{p}(c_j|\mathbf{e}) \approx \hat{p}(c_j|\mathbf{A}_p) = \alpha \hat{p}(c_j) \prod_{A_i \in \mathbf{A}_p} \hat{p}(A_i|c_j, \mathbf{A}_p(A_i)), \quad (4)$$

where $\hat{p}(c_j)$ is the priori probability distribution of class c_j and $\hat{p}(A_i|c_j, \mathbf{A}_p(A_i))$ is the conditional probability of A_i given the path attributes \mathbf{A}_p (e.g. from the root to the current node A_i) and c_j . α is a normalization factor. The proposed model is called *Probabilistic Inference Tree* (PIT). As one can see that a decision tree can represent the probability distribution instead of its traditional role as a decision boundary-based model. However, PIT also could not generate different class probabilities for the testing samples at the same leaf.

The learning methods previously designed for probability-based ranking are either sophisticated or with high computational complexity. Is there a simple and efficient way to improve decision tree for ranking? Aiming at this goal, we focus on lazy learning and deploy a lazy model at each tree leaf to generate distinct class probabilities for different testing samples.

2.2. AUC as a Metric of Ranking

Probability-based ranking sorts a set of samples according to the probabilities that the samples belong to one class. If we aim to obtain a more precise ranking from a model, one might naturally consider that we must know the true ranking in the training samples [4]. In most situations, however, what we are given is only a set of samples with class labels. Thus, providing classification labels in training and testing sample sets, are there better approaches than accuracy to evaluate learning models that achieve good ranking quality? The answer is *Receiver Operating Characteristics* (ROC) curve.

ROC curve has been introduced to machine learning relatively recently, in response to classification tasks with varying class distributions or misclassification costs [18]. Its key feature is the distinction between hit rate (true positive rate or TP) and false alarm rate (false positive rate or FP) as two separate performance measures. For a binary-class problem, assuming that the threshold t is the probability of randomly chosen negative points belonging to the positive class. Therefore, a ROC curve is constructed by plotting different points (FP,TP) as we move the threshold t between the extreme points 0 and 1 [10]. For a specific threshold t , $TP(t)$ is the probability that a randomly chosen positive point will have a larger probability of belonging to the positive class than t . We suppose that the probability density function of the probability that a randomly chosen negative point will have a larger probability of belonging to the positive class than t is $fp(t)$. Thus, we can obtain a ROC curve for a learning model by moving the threshold t to cover the whole FP distribution. The Area Under the ROC Curve, or simply AUC, has

been used to provide a good “summary” for the performance of the ROC curves. A simple approach to calculate the AUC value for a binary-class sample set is shown as below [8]:

$$AUC = \frac{S_+ - n_+(n_+ + 1)/2}{n_+n_-}, \quad (5)$$

where n_+ and n_- are the numbers of positive and negative samples respectively, and $S_+ = \sum_{|+|} r_i$, where r_i is the i th positive sample in the ranking list. In an extreme situation, the AUC value of a model could reach one if all positive samples are ranked behind any negative sample. For multi-class situations, we separately calculate the AUC value for each pair of classes (regarding each pair of classes as a binary-class situation) and average all the values [8].

3. Using Lazy Learners to Improve Tree-Based Ranking

3.1. The Lazy Learner

We observe that probability-based ranking is exactly a relative evaluation metric where a decent ranking depends on the relative position of a sample among others. More specifically, the ranks of samples $\{\mathbf{e}_i\}$ for a class value c_i (where $\mathbf{e}_i \in E$ and E is a sample set) are determined by sorting the class probabilities $\{\hat{p}(c|\mathbf{e}_i)\}$. Thus, a correct rank for a binary-class problem means placing a negative sample \mathbf{e}_- preceding a positive sample \mathbf{e}_+ , i.e. $\hat{p}(+|\mathbf{e}_-) < \hat{p}(+|\mathbf{e}_+)$ (or $\hat{p}(-|\mathbf{e}_-) > \hat{p}(-|\mathbf{e}_+)$). In contrast, in classification, a learning model makes a decision for a testing sample with reference to its assigned class probabilities without comparing with other testing samples, i.e. for the positive sample \mathbf{e}_+ , if $\hat{p}(+|\mathbf{e}_+) > \hat{p}(-|\mathbf{e}_+)$, its classification is correct.

It is reasonable that if a learning model is accidentally an approximation of the class distribution function, we will have the most accurate class probabilities and could obtain a perfect ranking. For most practical applications, unfortunately, it is almost impossible to get the true class distributions. But if we review the definition of ranking again, we could know that it is the order of the relative positions among the samples, not their estimated class probabilities, that plays a vital role in the ranking quality.

Table 1. The relative sample positions determine the AUC value

r_i	1	2	3	4	5	6	7	8	9	10
Model1	-	-	-	-	-	+	+	+	+	+
Model2	-	-	-	-	+	-	+	+	+	+
Model3	-	-	-	-	+	+	+	+	+	-

Now let us re-examine AUC more carefully. The extreme value of AUC is one if all the testing samples that belong to one class are sorted after all the testing samples that belong to other classes. This is demonstrated by the case of

Model1 in Table 1. $AUC_1 = \frac{(6+7+8+9+10)-5 \times 6/2}{5 \times 5} = 1$. Both Model2 and Model3 misplace one testing sample, but their AUC values are different. For Model2, $AUC_2 = \frac{(5+7+8+9+10)-5 \times 6/2}{5 \times 5} = \frac{24}{25}$ versus $AUC_3 = \frac{(5+6+7+8+9)-5 \times 6/2}{5 \times 5} = \frac{20}{25}$ for Model 3. This indicates that:

- It is worse that placing a testing sample into a higher rank in a wrong class than placing it to a lower rank in the wrong class.
- Separating the cluster of samples in one class from the rest of the testing samples can scale up the AUC value.

From above, it is clear that AUC is a discriminating measure. Notice that high AUC and high classification accuracy do not imply each other. One difference between them is that classification accuracy is based on 1-0 loss function that any misclassified testing sample is weighted equivalently, while the position of any misplaced testing sample affects AUC differently. In summary, AUC is a more discriminating measure than classification accuracy.

Therefore, we focus on developing an approach that could determine the relative positions of testing samples directly. We propose to use a probability estimator to explicitly compute the class probability $\hat{p}(c_j | \mathbf{e}_t)$ based on the similarities of a testing sample s_t to the training samples at the leaf into which \mathbf{e}_t falls. A probability estimator is defined as:

Definition 3.1. Given a set of testing samples \mathbf{E} and a set of class values $\mathbf{C} = \{c_i\}$, $\forall \mathbf{e}_t \in \mathbf{E}$, a **Probability Estimator** is a set of functions $\hat{p}(c_i | \mathbf{e}_t) \mapsto (0, 1)$ such that $\sum_{c_j \in \mathbf{C}} \hat{p}(c_j | \mathbf{e}_t) = 1$.

Using the probability estimator can produce a set of unique class probabilities for a testing sample instead of assigning a set of identical ones as in C4.5. Secondly, and more essentially, we have observed that probability-based ranking is indeed a relative evaluation measurement where the rightness of ranking depends on the relative position of a sample among others. These issues inspire us to use a lazy approach on decision tree. Lazy learning simply store some of the training samples and postpone any generalization effort until a testing sample must be classified. It can thus create many sample-specific local approximations, which attempt to fit the training samples only within the neighborhood of the testing sample. The lazy learner calculates the class probabilities for a testing sample based on its neighbors. It finds n closest neighbors at a leaf (we define that the neighbors are all the training samples at a leaf) for a testing sample and generates a weight for each neighbor using a new weighting function.

Assume that sample \mathbf{e} can be represented by an attribute vector as $\langle a_1(\mathbf{e}), a_2(\mathbf{e}), \dots, a_m(\mathbf{e}) \rangle$, where $a_i(\mathbf{e})$ denotes the value of i th attribute. The dis-

tance between sample \mathbf{e}_1 and \mathbf{e}_2 is calculated as in Eq. (6).

$$d(\mathbf{e}_1, \mathbf{e}_2) = \sqrt{\sum_{i=1}^m \delta(a_i(\mathbf{e}_1), a_i(\mathbf{e}_2))}, \quad (6)$$

where $\delta(a_i(\mathbf{e}_1), a_i(\mathbf{e}_2))$ outputs zero if $a_i(\mathbf{e}_1)$ is equivalent to $a_i(\mathbf{e}_2)$, otherwise it outputs one. The distance evaluates the degree of similarity between \mathbf{e}_1 and \mathbf{e}_2 . For a testing sample \mathbf{e}_t and a set of training samples $\{\mathbf{e}_i | i = 1, \dots, n\}$ in its neighborhood, we assign a weight to each \mathbf{e}_i based on its distance to \mathbf{e}_t , as in Eq. (7).

$$w_i = \frac{d_{max} - d_i}{d_{max} - d_{min}}, \quad (7)$$

where $d_i = d(\mathbf{e}_t, \mathbf{e}_i)$ and d_{max} (d_{min}) is the maximum (minimum) distance to \mathbf{e}_t at a leaf. For any two samples \mathbf{e}_i and \mathbf{e}_j , the shorter the distance to \mathbf{e}_t (assuming $d_i \leq d_j$), the larger weight the sample ($w_i \geq w_j$) gets. This indicates that the sample nearest to \mathbf{e}_t will contribute most when calculating the class probabilities for \mathbf{e}_t .

The final class probability of \mathbf{e}_t is a normalization of all the weights of the training samples in terms of their class values, as described in Eq. (8).

$$\hat{p}(c_j | \mathbf{e}_t) = \frac{\sum_{r=1}^m w_r^j + \frac{1}{|\mathbf{C}|}}{\sum_{k=1}^n w_k + 1}, \quad (8)$$

where n represents the number of training samples in the leaf, m is the number of training samples in class c_j and w_r^j is the weight of a training sample that belongs to class c_j within the neighborhood. $|\mathbf{C}|$ is the number of class values. Eq. (8) implies that the more training samples belonging to c_j , the bigger the sum of their weights and the more possibly that \mathbf{e}_t is in c_j .

3.2. Induce Lazy Distance-based Tree

We deploy a lazy learner at each leaf of a decision tree and call this model **Lazy Distance-based Tree** (LDTree). Compared with classic tree models, LDTree assigns a distinct class probability $\hat{p}(c|\mathbf{e}_t)$ to a testing sample \mathbf{e}_t instead of giving the uniform one as C4.5 does. As a result, testing samples could be greatly distinguished, which is much important for a good quality of ranking. Although lazy learning usually needs much time in computation, we deploy lazy learners at the sample subspaces of the leaves where they spend less computing time.

To induce the tree, we use a typical C4.5 algorithm. However, the expansion of the tree terminates when it satisfies that each of its leaves contains about 30 samples. It is suggested by [5] that when the goal is to yield good ranking performance, AUC as the splitting criterion is better than other splitting criteria that are aimed at increasing classification accuracy, such as *Gain Ratio* and *GINI*. But in order to show the improvement caused by lazy learners, we still use *Gain Ratio* as the splitting criterion.

Assume there is a decision tree T and a set of its leaves \mathbf{L} . In each leaf $l \in \mathbf{L}$, there is a set of leaf training samples \mathbf{S}_l . Algorithm 1 returns a set of class probabilities $\{\hat{p}(c_j|\mathbf{e}_t)\}$ for each class value $\forall c_j \in \mathbf{C}$ using T . In detail, $\{\hat{p}(c_j|\mathbf{e}_t)\}$ are initialized as zeros at the beginning. A testing sample \mathbf{e}_t is dispatched into a leaf l according to the values of its attributes. All the training samples at that leaf are regarded as the neighbors of \mathbf{e}_t . At that leaf, a lazy learner will assign a set of class probabilities to \mathbf{e}_t according to Eq. (6), Eq. (7) and Eq. (8).

Algorithm 1 $LDTree(T, \mathbf{e}_t)$ returns $\{\hat{p}(c_j|\mathbf{e}_t)\}$

T : a tree with a set of leaves \mathbf{L}

l : a leaf

\mathbf{S}_l : the training samples at leaf l

\mathbf{e}_t : a testing sample

$\{\hat{p}(c_j|\mathbf{e}_t)|c_j \in \mathbf{C}\}$: a set of class probabilities of \mathbf{e}_t

Dispatch \mathbf{e}_t to a leaf l according to its attributes

for each training sample $e_l \in \mathbf{S}_l$ **do**

 Use Eq. (6) to compute the distance d_l between e_l and \mathbf{e}_t

 Use Eq. (7) to compute a weight w_l for e_l

for each class value $c_j \in \mathbf{C}$ **do**

 Use Eq. (8) to compute $\{\hat{p}(c_j|\mathbf{e}_t)\}$

Return $\{\hat{p}(c_j|\mathbf{e}_t)\}$ for the testing sample \mathbf{e}_t

LDTree is actually a mixture of lazy learning and eager learning. The model is half built before a testing sample feeds. Despite the improvement shown in Section 4, we see that the testing phase of LDTree is much longer than an eager algorithm. This drives us to explore a novel way to improve LDTree.

3.3. Double Raise Decision Tree for Ranking

Classic decision trees are designed for classification and have been observed to prune the branches, which do not contribute to classification accuracy but are crucial to class probability estimation. By extending each of its branches, the resulting tree could improve the probability-based ranking. But as reviewed in Section 2, large trees also have their own drawbacks. Therefore, we consider to take advantage of large tree structure and solve its deficiencies using distance-based smoothing. Instead of directly averaging the class probabilities generated from the leaves or internal nodes as m -Branch or CFT does, our method links the final class probability estimation for a testing sample to the weights of the training samples on the path of the leaf into which the testing sample falls. The training samples with higher weight to the testing sample will have more effect. The tree inductive algorithm experiences two growth phases and is finally transformed into an eager model. Thus, we call it *Eager Distance-based Tree* (EDTree).

The first growth is conducted to get the weights for the training samples on tree paths. This phase is the same with LDTree. A typical tree algorithm with *Gain Ratio* as the splitting criterion is used, and we stop developing the tree when its leaves have enough samples. Now we will calculate the weight for each leaf training sample in a *leave-one-out* manner. At a leaf, we pick up a training sample \mathbf{e}_0 and assume its class value is c_x . Ignore this class value first and use Eq. (6), Eq. (7) and Eq. (8) to predict a set of class probabilities that \mathbf{e}_0 belongs to any class in \mathbf{C} . We use the class probability $\hat{p}(c_x|\mathbf{e}_0)$ as the weight of \mathbf{e}_0 in terms of c_x , and discard others. The weight of \mathbf{e}_0 is defined as below:

$$w_x(\mathbf{e}_0) = \hat{p}(c_x|\mathbf{e}_0). \quad (9)$$

We enumerate all the leaf training samples to compute their weights. The weights directly reflect the relative positions of the samples. If there exists a cluster of samples that have identical attribute values to \mathbf{e}_0 , their $w_x(\mathbf{e}_0)$ will be close to one.

We then continue to grow the tree until either its new leaves are pure or there are no more attributes to split. This process ends with a new tree \hat{T} and a set of new leaves \mathbf{L}_{new} . At each leaf $l_{\text{new}} \in \mathbf{L}_{\text{new}}$, there is a set of training samples $\mathbf{S}_{l_{\text{new}}}$ associated with it. These samples in $\mathbf{S}_{l_{\text{new}}}$ bring the weights computed at the first growth with them. We generate the probability estimation for each class value by normalizing all the weights of the training samples at leaf l_{new} in terms of their class values. Eq. (10) is almost the same as Eq. (8), except for one point that the yielded class probabilities are prepared for leaf l_{new} instead of a testing sample:

$$\hat{p}(c_j) = \frac{\sum_{r=1}^m w_r^j + \frac{1}{|\mathbf{C}|}}{\sum_{k=1}^n w_k + 1}, \quad (10)$$

Algorithm 2 specifically describes how EDTree works. In the testing period, given a testing sample \mathbf{e}_t , EDTree dispatches it into a leaf and assigns the estimated class probabilities of that leaf to \mathbf{e}_t as its final class probabilities. Note that, although both EDTree and C4.4 take advantage of large tree structures and *Laplace* correction, EDTree treat each leaf training sample discriminatively by referring to its weight. As a result, the number of values of yielded class probabilities should be richer than C4.4 does (C4.4 regards all leaf training samples equivalently when estimating class probabilities), and its corresponding AUC value will be reasonably higher than C4.4.

4. Experimental Methodology and Results

We evaluate learning models by using 34 UCI sample sets [2] recommended by *Weka* [20], a machine learning platform. These sample sets are extracted from different application domains and represent a variety of data characteristics. Table 2 presents a brief summary of the sample sets. All experiments were performed on a Pentium 4 with 2.0 GHz CPU and 512 MB RAM.

The preprocessing stages of sample sets mainly include four steps as follows:

Algorithm 2 $EDTree(E)$ return $\hat{T}, \{\hat{p}(c_j)\}$

\mathbf{E} : a sample set

T : a tree with a set of leaves \mathbf{L}

w : a learned weight

l : a leaf

\mathbf{S}_l : the training samples at leaf l

\hat{T} : eager distance-based tree

$\{\hat{p}(c_j)\}$: a set of class probabilities at a leaf

grow a tree T using training sample set \mathbf{E} and employing *Gain Ratio* as the splitting criterion. Stop developing until there are around 30 training samples at each leaf.

for each leaf $l \in \mathbf{L}$ **do**

for each training sample $\mathbf{e}_l \in \mathbf{S}_l$ **do**

 compute $w(\mathbf{e}_l)$ using Eq. (9)

grow T further by extending each of its branches and generate a new tree \hat{T}

for each leaf $l_{new} \in \mathbf{L}_{new}$ in \hat{T} **do**

for each class value $c_j \in \mathbf{C}$ **do**

 compute $\hat{p}(c_j)$ using Eq. (10)

Return \hat{T} and $\{\hat{p}(c_j)\}$

- (1) Applying the filter of *ReplaceMissingValues* to replace the missing values of attributes.
- (2) Applying the filter of *Discretize* to make numeric attributes discrete. Therefore, all the attributes are treated as nominal.
- (3) It is well known that, if the number of values of an attribute is almost equal to the number of samples in a sample set, this attribute does not contribute any information to classification. So we used the filter of *Remove* to delete these attributes. Three occurred within the 34 sample sets, namely *Hospital Number* in sample set *Horse-colic.ORIG*, *Instance Name* in sample set *Splice* and *Animal* in sample set *Zoo*.
- (4) Due to the relatively high time complexity of LDTree during the testing period, we applied the filter of unsupervised *Resample* to re-select sample set *Letter* and generate a new one named *Letter-2000*. The selection rate is 10%.

We conducted three groups of experiments with respect of probability-based ranking, measured by AUC. In each group, the AUC value on each sample set was measured via a five-fold cross validation five times. Runs with various models were carried out on the same training sets and evaluated on the same testing sets. Finally, we performed two-tailed *t*-tests [12] with a significantly different probability of 0.95 to compare our models with others. That is, we speak of two results for a sample set as being “significantly different” only if the difference is statistically significant

Table 2. Description of sample sets used in the experiments.

Sample Set	Size	Classes	Missing	Numeric
anneal	898	6	Y	Y
anneal.ORIG	898	6	Y	Y
audiology	226	24	Y	N
autos	205	7	Y	Y
balance	625	3	N	Y
breast	286	2	Y	N
breast-w	699	2	Y	N
colic	368	2	Y	Y
colic.ORIG	368	2	Y	Y
credit-a	690	2	Y	Y
credit-g	1000	2	N	Y
diabetes	768	2	N	Y
glass	214	7	N	Y
heart-c	303	5	Y	Y
heart-h	294	5	Y	Y
heart-s	270	2	N	Y
hepatitis	155	2	Y	Y
ionosphere	351	2	N	Y
iris	150	3	N	Y
kr-vs-kp	3196	2	N	N
labor	57	2	Y	Y
letter-2000	2000	26	N	Y
lymph	148	4	N	Y
mushroom	8124	2	Y	N
p.-tumor	339	21	Y	N
sick	3772	2	Y	Y
sonar	208	2	N	Y
soybean	683	19	Y	N
splice	3190	3	N	N
vehicle	846	4	N	Y
vote	435	2	Y	N
vowel	990	11	N	Y
waveform-5000	5000	3	N	Y
zoo	101	7	N	Y

Note: We downloaded these sample sets in the format of *arff* from the main web page of *Weka*.

at the 0.05 level according to the corrected two-tailed t -test. Besides, in all t -test tables, each entry $w/t/l$ means that the model in the corresponding row wins w sample sets, ties in t sample sets, and loses l sample sets, in contrast with the model in the corresponding column.

As mentioned in previous sections, we used the *Laplace* correction on all tree models to avoid the zero-frequency problem. More specifically, assume there are n_c samples that have the class value as c , t total samples, and k class values in a sample set. The *Laplace* correction calculates the probability $\hat{p}(c)$ as $\hat{p}(c) = \frac{n_c+1}{t+k}$. And similarly, $\hat{p}(a_i|c)$ is calculated by $\hat{p}(a_i|c) = \frac{n_{ic}+1}{n_c+v_i}$, where v_i is the number of values of attribute A_i and n_{ic} is the number of samples in class c with $A_i = a_i$.

The abbreviations used in tables are described below.

- LDTree-B:** using bagging [9] with LDTree as the base learner.
C4.5: the traditional decision-tree model [16].
C4.5-L: C4.5 with *Laplace* correction applied at leaves.
C4.5-M: C4.5 with *m*-Branch [6] applied.
CFT4.5: C4.5 with the confusion factor algorithm [11] applied.
C4.5-B: using bagging with C4.5 as the base learner.
C4.5-Ada: using AdaBoost with C4.5 as the base learner.
C4.4: an improved decision tree for better probability estimation [13].
C4.4-NL: C4.4 without *Laplace* correction applied at leaves.
C4.4-M: C4.4 with *m*-Branch applied.
CFT4.4: C4.4 with the confusion factor algorithm applied.
C4.4-B: using bagging with C4.4 as the base learner.
C4.4-Ada: using AdaBoost with C4.4 as the base learner.

In the first group of our experiments (Table 3), EDTree and LDTree were compared with C4.5 and its single-tree variants (C4.5-L, C4.5-M and CFT4.5). In the second group (Table 5), we compared EDTree and LDTree with C4.4 and its single-tree variants (C4.4-NL, C4.4-M and CFT4.4). In the last group (Table 7), we made comparisons between LDTree-B and the ensemble PETs that contain bagged or boosted decision trees (C4.5-B, C4.4-B, C4.5-Ada and C4.4-Ada).

We implemented EDTree, LDTree, AUC metric, *m*-Branch and CFTs within *Weka* and used the current implementations of other models, bagging and AdaBoost methods in *Weka*. For multi-class AUC situations, *M*-measure [8] had been adopted. Furthermore, we observed that, although [11] indicated that CFTs are not sensitive to the confusion factor setting, we found that using the percentage of the subset as the confusion factor was slightly better than the proposed optimal parameter 0.3. Thus, we used a different confusion factor in our experiments. The highlights of our observations are listed as below:

- In Table 3 and Table 5, EDTree has better AUC values than LDTree. Compared with LDTree, EDTree wins 10 sample sets and loses no sample set. As we have discussed in Section 3, after constructing a LDTree, EDTree continues to develop the tree into a large size. This is because a large tree has the natural inherit of discerning samples. The average AUC values of EDTree is 89.35%, the highest among all the competitive models in our experiments. Furthermore, results show EDTree has less variance than LDTree reflected by their average standard deviations, which proves that it is efficient to use the training sample weights that are learned when the tree size is relatively small and there are sufficient training samples at its leaves, to relieve the high variance of a large tree. In the fairly large tree, the approach of weighting samples can also avoid the problem of over fitting the training sample sets.
- EDTree and LDTree achieve remarkably good performances in AUC among C4.5 and its single-tree variants showed in Table 3 and Table 4. Compared with C4.5, EDTree wins 26 sample sets and loses no sample set. On the 26 winning sample

sets, there are 11 sample sets in which the AUC values of EDTree are 10% higher than C4.5. LDTree wins C4.5 in 24 sample sets and also never loses. Although the variants of C4.5 could also improve its AUC values, our models still tremendously outperform them all measured by AUC.

- C4.4, the improvement version of C4.5 on ranking, has better AUC values than C4.5 presented in Table 5 and Table 6. Compared with C4.4, EDTree (LDTree) wins 17 (8) sample sets and loses 0 (1) sample set. Compared with C4.4 single-tree variants, EDTree tremendously outperforms others and LDTree is also better than most of them. Although C4.4-M slightly outperforms LDTree in AUC, C4.4-M sacrifices the tree size to improve class probability estimation, which could incur the “over-fitting” problem and would be sensitive to noise data. LDTree preserves a small tree and is better than C4.4-M in classification accuracy (we did not present the empirical results due to the space limitation). Besides, decision trees with m -Branch could not generate multiple class probabilities for the testing samples that fall into the same leaf. This is different from LDTree in which each testing sample will be assigned a set of unique class probabilities based on its leaf training samples. Thus, in the applications where both high classification accuracy and accurate ranking are required, LDTree is a better model than C4.4-M.
- In Table 7 and Table 8, LDTree-B outperforms C4.5-B in 14 sample sets and loses no sample set. It proves that deploying lazy learners at leaves could be a good way to weight each leaf training sample in terms of its differences with a testing sample. As a result, bagged trees with training samples weighted result in better AUC values. Having fewer branches, LDTree-B is also slightly better than C4.4-B in 2 sample sets with no sample set lost. Note that, from this comparison, we can learn that lazy learners could approximately discriminate testing samples just the same as turning off pruning does, except for the point that lazy learners do not change the tree structure with the risk of over-fitting the sample set. In aspect of boosted PETs, LDTree-B works as competitively as C4.5-Ada or C4.4-Ada does. LDTree-B wins in 4 sample sets and loses 3 sample sets, compared with them both. We also notice that boosting seems to be more effective than bagging when applied to C4.5 and C4.4, which supports the conclusions draw by [3]. For instance, C4.5-Ada is superior to C4.5-B in 9 sample sets and inferior on 3 ones.
- Other than having good AUC results, our models also have better robustness and stability than their counterpart comparators. EDTree’s average standard deviation in AUC is 3.21, which is one of the lowest in all models. The average standard deviation of LDTree (3.47) is also the best in its groups. Besides, the AUC values of LDTree-B are more stable than most of other ensemble PETs.

From the above comparisons, we conclude that EDTree works better than LDTree due to its extension of tree size. Actually, EDTree could be regarded as a trade-off between the quality of probability-based ranking and the comprehensibility of results in model selection. EDTree has theoretically fully expressed the representation of decision trees via its large tree structure. However, in order to

overcome some intrinsic obstacles to build an accurate PET under the paradigm of a large tree, we develop a tree in two stages and calculate a weight for each leaf training sample when the tree still has enough training samples on its leaves. One another interesting observation is that CFTs work well in a relatively small tree compared with a large one and m -Branch works consistently in any tree size.

5. Conclusions

In this paper, we reveal the fact that probability-based ranking is a relative evaluation metric that the relative positions of the samples determine its performance. We also observe that when using AUC as the ranking criterion, the best practice is to assign class probabilities that completely separate samples that belong to different classes. Therefore, we focus on improving the ranking performance of decision tree by assigning a class probability to a testing sample via evaluating its relative position among its neighbors. We design a lazy learner to explicitly compute the class probability for a testing sample based on its similarities to the training samples within its neighborhood. Two new models are generated with lazy learners applied. Empirical results proved our expectation. We believe that our try has opened a new path to improve the probability-based ranking performance of decision tree. In future work, we propose to combine lazy learners with other models, such as Bayesian network, to improve the ranking quality of these models.

References

1. E. Bauer and R. Kohavi. An empirical comparison of voting classification algorithms: Bagging, boosting and variants. *Artificial Intelligence*, 36, 1999.
2. C. Blake and C.J. Merz. Uci repository of machine learning database.
3. R. Caruana and A. Niculescu-Mizil. An empirical evaluation of supervised learning for roc area. In *The First Workshop of ROC Analysis in AI*, 2004.
4. W. W. Cohen, R.E. Schapire, and Y. Singer. Learning to order things. *Journal of Artificial Intelligence Research*, 10.
5. C. Ferri, P. A. Flach, and J. Hernandez-Orallo. Learning decision trees using the area under the roc curve. In *Proceedings of International Conference on Machine Learning (ICML2002)*. Springer, 2002.
6. C. Ferri, P.A. Flach, and J. Hernandez-Orallo. Improving the auc of probabilistic estimation trees. In *Proceedings of the Fourteenth European Conference on Machine Learning*. Springer, 2003.
7. Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. In *European Conference on Computational Learning Theory*, pages 23–37. Springer, 1995.
8. D. J. Hand and R. J. Till. A simple generalisation of the area under the roc curve for multiple class classification problems. *Machine Learning*, 45, 2001.
9. T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer, 2001.
10. J. Huang and C. Ling. Using auc and accuracy in evaluating learning algorithms. 17(3), 2005.
11. C.X. Ling and R.J. Yan. Decision tree with better ranking. In *Proceedings of the Twentieth International Conference on Machine Learning*. Morgan Kaufmann, 2003.

16 Yuhong Yan and Han Liang

12. C. Nadeau and Y. Bengio. Inference for the generalization error. *Machine Learning*, 52(40), 2003.
13. F. J. Provost and P. Domingos. Tree induction for probability-based ranking. *Machine Learning*, 52(30), 2003.
14. F. J. Provost, T. Fawcett, and R. Kohavi. The case against accuracy estimation for comparing induction algorithms. In *Proceedings of the Fifteenth International Conference on Machine Learning*. Morgan Kaufmann, 1998.
15. J. R. Quinlan. Induction of decision trees. *Machine Learning*, 2(1), 1986.
16. J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann: San Mateo, CA, 1993.
17. Robert E. Schapire. The strength of weak learnability. *Machine Learning*, 5:197–227, 1990.
18. Kent A. Spackman. Signal detection theory: valuable tools for evaluating inductive learning. In *Proceedings of the sixth international workshop on Machine learning*, pages 160–163. Morgan Kaufmann Publishers Inc., 1989.
19. J. Su and H. Zhang. Probabilistic inference trees for classification and ranking. In *Proceedings of the Nineteenth Canadian Conference on Artificial Intelligence*. Springer, 2006.
20. I. H. Witten and E. Frank. *Data Mining – Practical Machine Learning Tools and Techniques with Java Implementation*. Morgan Kaufmann, 2000.

Table 3. Experiment results in AUC and its standard deviation. Comparing EDTree, LDTree with C4.5, C4.5 with *Laplace* correction (C4.5-L), C4.5 with *m*-Branch (C4.5-M) and C4.5 with confusion factor algorithm (CFT4.5)

Sample Set	EDTree	LDTree	C4.5	C4.5-L	C4.5-M	CFT4.5
anneal	95.56±3.80	95.70±5.83	87.22±4.46 ●	91.31±5.11	91.79±6.17	92.20±4.03
anneal.ORIG	94.65±3.57	93.13±7.62	89.10±4.47 ●	91.00±7.33	91.88±8.49	90.94±7.06
audiology	72.46±2.35	74.98±2.21	68.12±1.65 ●	67.45±2.17 ●	69.73±2.44	73.98±2.41
autos	94.62±1.53	94.33±1.68	77.16±5.97 ●	92.55±2.28 ●	93.94±1.72	87.78±2.70 ●
balance-scale	72.36±6.01	58.16±5.05●	53.18±6.13 ●	56.60±5.30 ●	54.97±5.33 ●	61.06±5.09 ●
breast-cancer	69.50±8.15	66.56±8.90	61.36±6.20 ●	61.50±6.24 ●	62.35±5.38	66.59±8.74
breast-w	98.15±1.01	98.32±0.89	95.84±2.25 ●	97.51±1.25	97.46±1.33	95.75±2.53
colic	87.69±4.04	86.95±4.34	82.60±5.45 ●	85.09±4.75 ●	85.42±4.44 ●	84.17±4.86
colic.ORIG	86.92±3.70	85.18±4.16	82.40±6.58	83.89±3.22	84.47±2.95	76.69±9.16 ●
credit-a	92.51±1.76	90.09±2.18●	87.73±3.62 ●	88.89±3.49 ●	88.95±3.51 ●	88.58±3.61 ●
credit-g	76.99±3.64	71.08±4.18●	68.45±4.13 ●	71.31±3.82 ●	71.98±3.68 ●	70.13±3.80 ●
diabetes	81.42±4.16	79.90±4.65	77.03±4.97 ●	78.14±4.65 ●	78.22±4.66 ●	75.77±4.58 ●
glass	89.97±2.70	83.89±4.34●	77.47±3.20 ●	83.22±4.04 ●	83.52±4.08 ●	80.28±6.36 ●
heart-c	83.60±0.51	83.54±0.58	83.02±0.80 ●	83.21±0.64	83.25±0.58 ●	83.51±0.47
heart-h	83.68±0.62	83.74±0.52	82.11±2.98	82.22±3.02	82.24±3.02	82.30±3.07
heart-statlog	86.32±5.06	84.48±4.98	79.64±6.38 ●	82.84±6.57	82.69±6.84	83.23±5.62
hepatitis	80.43±9.49	83.49±5.57	67.62±10.54●	68.88±11.26●	69.15±11.45●	68.76±12.90
ionosphere	94.06±3.08	92.50±3.83	87.87±4.77 ●	90.06±3.56 ●	89.86±3.78 ●	80.56±6.48 ●
iris	98.74±1.16	98.61±1.69	98.83±0.98	98.31±1.41	98.29±1.41	98.54±1.31 ●
kr-vs-kp	99.93±0.06	99.90±0.08	99.74±0.21	99.82±0.14	99.82±0.14	99.62±0.23
labor	88.83±11.43	92.13±9.96	76.48±15.12	82.20±13.00	82.20±13.00	81.88±13.87
letter-2000	95.50±1.22	90.25±0.94●	85.96±1.19 ●	84.65±1.56 ●	91.60±0.76 ●	84.40±1.63 ●
lymph	86.68±6.88	84.11±7.12	73.37±10.58●	84.65±7.70	83.60±7.78	84.49±9.92
mushroom	100.00±0.00	100.00±0.00	100.00±0.00	100.00±0.00	100.00±0.00	99.47±0.19 ●
primary-tumor	83.78±3.03	73.89±3.55●	67.49±3.59 ●	72.03±2.80 ●	75.69±2.91 ●	75.99±3.31 ●
sick	99.00±0.40	98.07±1.08	93.39±3.10 ●	93.68±2.46 ●	94.36±2.44 ●	92.02±2.82 ●
sonar	81.44±6.28	77.55±7.47	71.70±7.55 ●	75.47±7.22 ●	75.55±7.31 ●	72.99±7.64 ●
soybean	99.62±0.31	98.76±0.84●	98.69±0.80 ●	98.36±0.73 ●	99.56±0.24	98.40±2.14
splice	98.43±0.48	98.30±0.44	96.52±0.98 ●	97.95±0.61	98.04±0.57	98.06±0.59
vehicle	89.46±1.64	86.02±1.70●	81.51±2.44 ●	85.73±1.72 ●	86.21±1.59 ●	79.18±2.76 ●
vote	97.62±1.50	98.29±1.36	97.05±2.07	97.52±1.62	97.50±1.63	98.33±1.44
vowel	96.25±1.19	93.77±1.47●	92.23±1.57 ●	89.65±1.77 ●	95.05±1.00 ●	84.11±2.79 ●
waveform-5000	90.25±0.76	86.97±1.00●	84.44±1.23 ●	86.70±1.00 ●	88.29±0.91 ●	84.19±1.65 ●
zoo	91.57±7.51	93.13±7.71	91.43±6.86	91.38±7.33	91.67±7.14	93.79±5.22
Average	89.35±3.21	87.52±3.47	82.85±4.20	85.11±3.82	85.86±3.78	84.35±4.44

Note: ○, ● statistically significant improvement or degradation over EDTree

Table 4. Summary of the experiment results in Table 3

Models	C4.5	C4.5-L	C4.5-M	CFT4.5	LDTree
C4.5-L	10/23/1				
C4.5-M	11/23/0	6/28/0			
CFT4.5	6/26/2	2/25/7	1/24/9		
LDTree	24/10/0	10/24/0	5/25/4	13/21/0	
EDTree	26/8/0	19/15/0	16/18/0	17/17/0	10/24/0

Table 5. Experiment results on AUC and its standard deviation. Comparing EDTree, LDTree with C4.4, C4.4 without Laplace correction (C4.4-NL), C4.4 with m -Branch (C4.4-M) and C4.4 with confusion factor algorithm (CFT4.4)

Sample Set	EDTree	LDTree	C4.4	C4.4-NL	C4.4-M	CFT4.4
anneal	95.56±3.80	95.70±5.83	95.04±6.91	85.33±4.81 ●	95.16±6.25	93.58±5.40
anneal.ORIG	94.65±3.57	93.13±7.62	91.56±8.18	87.59±3.79 ●	92.69±8.27	91.64±7.53
audiology	72.46±2.35	74.98±2.21	69.45±2.78 ●	68.76±1.85 ●	71.39±2.19	74.84±2.53 ○
autos	94.62±1.53	94.33±1.68	92.13±2.10 ●	77.31±5.99 ●	94.19±1.62	87.10±2.61 ●
balance-scale	72.36±6.01	58.16±5.05●	62.11±5.90 ●	56.68±4.72 ●	57.57±5.32 ●	62.88±4.54 ●
breast-cancer	69.50±8.15	66.56±8.90	62.89±6.67 ●	59.14±6.72 ●	63.42±7.19 ●	62.10±11.53
breast-w	98.15±1.01	98.32±0.89	98.06±0.99	95.13±2.32 ●	98.08±0.97	89.14±3.21 ●
colic	87.69±4.04	86.95±4.34	84.56±4.51	79.80±6.05 ●	87.02±4.51	83.67±4.74 ●
colic.ORIG	86.92±3.70	85.18±4.16	82.33±4.36 ●	78.70±5.42 ●	83.63±3.49 ●	74.90±6.04 ●
credit-a	92.51±1.76	90.09±2.18●	89.44±2.46 ●	84.71±3.22 ●	91.17±2.16 ●	88.49±2.88 ●
credit-g	76.99±3.64	71.08±4.18●	68.54±2.93 ●	63.54±3.51 ●	72.42±2.80 ●	70.37±3.59 ●
diabetes	81.42±4.16	79.90±4.65	76.37±3.00 ●	70.99±4.38 ●	78.86±3.31 ●	71.78±5.07 ●
glass	89.97±2.70	83.89±4.34●	82.75±5.33 ●	74.95±3.82 ●	84.24±4.83 ●	79.99±5.84 ●
heart-c	83.60±0.51	83.54±0.58	83.13±0.68 ●	82.63±0.79 ●	83.38±0.63	83.18±0.65
heart-h	83.68±0.62	83.74±0.52	83.38±0.61	82.68±0.65 ●	83.69±0.57	82.97±0.68
heart-statlog	86.32±5.06	84.48±4.98	81.94±4.35 ●	77.23±4.57 ●	84.34±5.27	80.63±6.85
hepatitis	80.43±9.49	83.49±5.57	76.62±10.78	69.24±12.13●	78.31±12.09	74.08±10.74
ionosphere	94.06±3.08	92.50±3.83	92.88±2.80	85.95±5.61 ●	92.47±3.21	82.14±4.55 ●
iris	98.74±1.16	98.61±1.69	98.43±1.41	97.53±2.22	98.42±1.41	97.85±1.56 ●
kr-vs-kp	99.93±0.06	99.90±0.08	99.90±0.07	99.77±0.20	99.90±0.07	99.77±0.17
labor	88.83±11.43	92.13±9.96	84.13±15.41	81.16±14.85	86.35±14.04	81.90±16.49
letter-2000	95.50±1.22	90.25±0.94●	84.31±1.66 ●	85.47±1.11 ●	92.13±0.84 ●	83.13±1.47 ●
lymph	86.68±6.88	84.11±7.12	85.45±7.27	73.73±9.70 ●	85.04±6.75	83.93±9.24
mushroom	100.00±0.00	100.00±0.00	100.00±0.00	100.00±0.00	100.00±0.00	99.47±0.19 ●
primary-tumor	83.78±3.03	73.89±3.55●	72.35±2.89 ●	65.79±2.86 ●	76.18±2.53 ●	76.15±3.23 ●
sick	99.00±0.40	98.07±1.08	99.08±0.33	96.02±2.38 ●	99.14±0.34	94.94±2.07 ●
sonar	81.44±6.28	77.55±7.47	77.14±7.48	72.33±7.31 ●	78.94±7.47	73.21±8.84
soybean	99.62±0.31	98.76±0.84●	98.35±0.78 ●	98.36±0.76 ●	99.55±0.26	98.24±2.11
splice	98.43±0.48	98.30±0.44	97.95±0.58	94.86±1.10 ●	98.40±0.55	97.39±0.74 ●
vehicle	89.46±1.64	86.02±1.70●	85.18±2.22 ●	77.85±3.07 ●	86.89±1.70 ●	78.61±2.57 ●
vote	97.62±1.50	98.29±1.36	97.50±1.53	96.84±2.05	97.59±1.54	98.52±1.14
vowel	96.25±1.19	93.77±1.47●	90.55±1.63 ●	91.64±1.80 ●	95.49±0.97	84.17±2.81 ●
waveform-5000	90.25±0.76	86.97±1.00●	81.25±1.06 ●	79.11±1.23 ●	86.92±0.83 ●	82.38±1.35 ●
zoo	91.57±7.51	93.13±7.71	91.57±7.44	91.57±6.93	91.67±7.36	93.78±5.24
Average	89.35±3.21	87.52±3.47	85.77±3.74	81.84±4.06	87.20±3.57	84.03±4.36

Note: ○, ● statistically significant improvement or degradation over EDTree

Table 6. Summary of the experiment results in Table 5

Models	C4.4	C4.4-NL	C4.4-M	CFT4.4	LDTree
C4.4-NL	0/13/21				
C4.4-M	15/18/1	24/10/0			
CFT4.4	3/23/8	8/22/4	2/20/12		
LDTree	8/25/1	23/11/0	1/29/4	12/22/0	
EDTree	17/17/0	28/6/0	11/23/0	19/14/1	10/24/0

Table 7. Experiment results on AUC and its standard deviation. Comparing bagged LDTree (LDTree-B) with C4.5-B, C4.4-B, C4.5-Ada and C4.4-Ada

Sample Set	LDTree-B	C4.5-B	C4.4-B	C4.5-Ada	C4.4-Ada
anneal	96.46±4.97	90.13±3.81 ●	95.48±6.60	96.57±3.61	93.21±6.91
anneal.ORIG	93.85±7.35	89.40±4.56 ●	93.80±8.08	94.42±7.53	94.16±7.87
audiology	76.20±2.29	71.75±1.75 ●	74.73±2.29	74.61±2.28	75.82±2.24
autos	95.44±1.32	81.22±3.09 ●	95.00±1.47	95.73±1.24	95.46±1.41
balance-scale	64.32±4.11	59.91±4.64 ●	66.04±3.18	75.88±4.40 ○	70.10±3.52 ○
breast-cancer	66.13±8.91	64.51±8.40	65.46±7.66	61.84±7.28	62.37±8.40
breast-w	98.62±0.82	98.24±1.29	98.69±0.86	98.82±0.75	98.55±0.92
colic	88.49±4.31	85.40±5.65 ●	88.14±3.81	84.21±3.88	85.53±4.52
colic.ORIG	86.84±3.51	86.83±3.56	85.38±3.60	83.23±3.79	83.13±4.44
credit-a	90.66±2.48	90.70±2.66	90.52±2.28	87.96±2.53 ●	89.17±2.02
credit-g	76.21±3.29	74.98±3.53 ●	74.38±3.02	72.43±3.46 ●	71.49±2.61 ●
diabetes	81.05±3.64	80.02±3.95 ●	78.73±3.17 ●	76.57±3.86 ●	76.14±3.70 ●
glass	86.98±3.75	82.09±1.84 ●	84.89±4.42	84.84±3.58	82.00±4.58 ●
heart-c	83.81±0.42	83.63±0.50	83.63±0.45	83.52±0.48	83.42±0.50
heart-h	83.83±0.53	83.68±0.52	83.64±0.60	83.44±0.59	83.49±0.49
heart-statlog	87.18±4.86	85.37±5.66 ●	86.18±5.00	83.94±6.75	84.12±4.77
hepatitis	83.99±7.34	79.95±9.62	83.55±7.42	81.76±8.79	82.94±7.54
ionosphere	94.58±3.29	94.06±3.92	95.04±2.71	95.17±3.08	95.02±2.53
iris	98.67±1.18	98.99±1.01	98.65±1.30	98.87±1.04	98.16±1.59
kr-vs-kp	99.96±0.03	99.92±0.07	99.96±0.04	99.94±0.12	99.94±0.08
labor	92.34±9.90	86.16±13.18	88.63±13.35	91.79±12.35	91.09±10.90
letter-2000	93.63±1.12	91.08±1.34 ●	93.26±1.03	95.87±0.58 ○	95.77±0.51 ○
lymph	88.47±4.52	84.37±8.64	87.26±4.09	88.61±2.46	88.16±4.44
mushroom	100.00±0.00	100.00±0.00	100.00±0.00	100.00±0.00	100.00±0.00
primary-tumor	76.12±2.83	72.28±2.79 ●	76.29±2.63	72.48±3.50	74.70±3.34
sick	98.35±1.04	93.99±2.18 ●	99.20±0.26	98.39±1.25	98.24±1.45
sonar	81.66±6.38	80.68±6.49	82.23±6.40	79.89±6.25	77.87±6.95
soybean	99.59±0.43	99.79±0.33	99.57±0.34	99.75±0.28	99.73±0.33
splice	98.78±0.35	98.52±0.46 ●	98.74±0.43	98.76±0.37	98.65±0.46
vehicle	89.32±1.71	88.97±1.86	89.03±1.78	86.66±2.24 ●	86.12±2.27 ●
vote	98.62±1.22	97.60±1.59	98.29±1.48	98.11±1.49	97.99±1.63
vowel	96.15±1.00	96.00±1.01	95.88±1.03	97.33±1.01 ○	97.64±0.92 ○
waveform-5000	90.87±0.79	90.86±0.73	89.73±0.91 ●	91.73±0.72	90.72±0.67
zoo	93.62±7.66	92.98±7.01	93.17±8.09	94.67±6.16	94.24±7.17
Average	89.14±3.16	86.88±3.46	88.62±3.23	88.46±3.17	88.09±3.28

Note: ○, ● statistically significant improvement or degradation over LDTree-B

Table 8. Summary of the experiment results in Table 7

Models	C4.5-B	C4.4-B	C4.5-Ada	C4.4-Ada
C4.4-B	8/25/1			
C4.5-Ada	9/22/3	4/28/2		
C4.4-Ada	7/25/2	3/30/1	0/33/1	
LDTree-B	14/20/0	2/32/0	4/27/3	4/27/3