

Improve Decision Trees for Probability-Based Ranking by Lazy Learners

Han Liang *

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

Yuhong Yan

National Research Council of Canada
Fredericton, NB, Canada E3B 5X9
yuhong.yan@nrc.gc.ca

Abstract

Existing work shows that classic decision trees have inherent deficiencies in obtaining a good probability-based ranking (e.g. AUC). This paper aims to improve the ranking performance under decision-tree paradigms by presenting two new models. The intuition behind our work is that probability-based ranking is a relative metric among samples, therefore, distinct probabilities are crucial for accurate ranking. The first model, **Lazy Distance-based Tree (LDTree)**, uses a lazy learner at each leaf to explicitly distinguish the different contributions of leaf samples when estimating the probabilities for an unlabeled sample. The second model, **Eager Distance-based Tree (EDTree)**, improves LDTree by changing it into an eager algorithm. In both models, each unlabeled sample is assigned a set of unique probabilities of class membership instead of a set of uniformed ones, which gives finer resolution to differentiate samples and leads to the improvement of ranking. On 34 UCI sample sets, experiments verify that our models greatly outperform C4.5, C4.4 and other standard smoothing methods designed for better ranking.

1 Introduction

Classification accuracy has been regarded as a major evaluation metric for the performances of learning models in many supervised-learning applications. However, in some real-world problems, a precise ranking of samples based on the probabilities of their class memberships is more desirable. For instance, in a banking system, we often need an accurate ranking of applicants in terms of their likelihoods of profitability, which is more important than merely classifying them into *qualified* or *non-qualified* categories.

Traditional decision trees, such as C4.5 [10] and ID3 [9],

have been observed to yield poor probability-based ranking [8]. Learning decision trees with accurate probability estimation, called *Probability Estimation Trees (PETs)*, has recently attracted substantial attention. Provost and Domingos [7] have proposed to improve C4.5 for better ranking by applying two techniques on it: turning off pruning and collapsing to keep some branches that contribute much to the quality of ranking, and using *Laplace* correction at leaves to smooth the generated probabilities towards the prior distribution. The new model is called C4.4. Although experiments have shown that C4.4 greatly scales up the ranking quality of decision trees, there still exist two contradictions.

1. Without pruning, C4.4 is relatively much larger than traditional decision trees, which may over-fit sample sets in the training time and the resulting probabilities are definitely inaccurate.
2. Likewise, with the expansion of tree size, the number of leaves will increase so that the number of samples at each leaf will become relatively small. The probabilities produced from such a small sample set are unreliable. Besides, probability values could easily repeat, which also incurs the problem that more unlabeled samples share the same probability and will be ranked randomly. This results in a negative effect on the ranking performance.

The improvement presented in this paper comes from the intuition that probability-based ranking is a relative metric. A precise ranking is based on the correct order of the conditional probabilities that the unlabeled samples $\{s_t\}$ belong to a specific class c_j , i.e. ranking is to sort the items in set $\{\hat{p}(c_j|s_t)|s_t \in S\}$, where S is the testing set. Therefore, we propose to use a lazy learner to explicitly compute $\hat{p}(c_j|s_t)$ based on the similarities of s_t to the samples at the leaf into which s_t falls. We also want to give a set of unique probabilities to each unlabeled sample instead of assigning a set of uniformed ones to all the unlabeled samples that fall into the same leaf, as in C4.4. This can give finer reso-

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

lution to distinguish the samples and result in better ranking. In this paper, we introduce two new models called *Lazy Distance-based Tree* (LDTree) and *Eager Distance-based Tree* (EDTree). For LDTree, first grow a tree until its leaves still have a relatively large number of samples (around 30 samples), then for any unlabeled sample, a lazy learner that takes advantage of maximum and minimum distances between the unlabeled sample and the samples at the leaf into which the unlabeled sample falls, is used to generate a weight for each leaf sample. LDTree yields the probability estimation by normalizing all the weights of leaf samples in terms of their class values. LDTree preserves a traditional decision tree without modification of its structure. The second model, EDTree, is an improvement of LDTree. First, we use the same spirit to consider the similarities between samples, but change the LDTree algorithm into an eager one. Second, we know that the improvement of C4.4 comes from its larger tree size contrast to C4.5, since a large tree has the priority of discerning samples. Therefore, we want to build a larger tree as well. The proposed EDTree applies the leave-one-out technique on each leaf of LDTree to select an unlabeled case within the leaf, deploy a lazy learner to learn a set of probabilities for the case, and after all the samples at the leaf are exhaustively enumerated, continue growing this branch until either its posterior leaves are pure or there are no more attributes to split. The probability estimation for each leaf of the new tree is a normalization of the corresponding probabilities of leaf samples in terms of their class values. In the testing time, EDTree assigns the set of normalized values of a leaf into which an unlabeled sample falls directly to the unlabeled sample. Based on 34 UCI sample sets, our experiments indicate that LDTree greatly outperforms C4.5, C4.4 and most of other tree variants designed for better ranking. EDTree is best among all the compared models in the decision tree families.

The rest of this paper is organized as follows. Section 2 introduces the related work in improving decision trees for better ranking. Section 3 presents our new models. Section 4 describes the experiment configuration and results. We conclude in Section 5.

2 Related Work

Under traditional decision-tree paradigms, such as C4.5 [10], an explicit decision boundary is induced from labeled leaf samples and an unlabeled sample s_t is classified into class c_j if s_t falls into the decision area corresponding to c_j . However, It has been learned that traditional decision trees have two inherent defects that result in their poor probability-based ranking: **(1)** uniform probability estimation (decision trees assign a uniform probability to all the unlabeled samples falling into the same leaf) and

(2) adopting pruning (based on accuracy maximization, tree inductive algorithms remove some branches that contribute substantially to the ranking performance but less in improving classification accuracy). Aiming at these facts, Provost and Domingos [7] proposed several techniques to improve the ranking performance of C4.5. First, using *Laplace* correction at leaves to smooth the yielded probabilities toward the prior probability distribution. Second, turning off pruning and collapsing mechanisms in C4.5. The final version is C4.4. They also pointed out that *bagging*, an ensemble method, could greatly improve decision trees in terms of probability-based ranking.

Ferri et al. [5] introduced another method, called *m-Branch*, to generate probability estimates at leaves. *m-Branch* is a recursive root-to-leaf extension of the *m* probability estimation, in which, on each path, the probability estimates on a parent node are propagated downwards to all of its children. Equation 1 describes the process.

$$\hat{p}_{child}(c_j|s_t) = \frac{n_{c_j} + m * \hat{p}_{parent}(c_j|s_t)}{\sum_{c_j \in C} n_{c_j} + m}, \quad (1)$$

where parameter m is adjusted by the depth and cardinality of the node, and n_{c_j} is the number of samples that belong to class c_j within the node.

Ling and Yan [3] introduced a novel algorithm to improve the ranking performance of decision trees, that is, for a given unlabeled sample s_t , the probability estimates for s_t are the averages of probability estimates from all the leaves of a tree. 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 the leaf to the root) that the leaf has, compared with s_t .

In their most recent work, Su and Zhang [11] proposed to understand decision trees from a probabilistic perspective. They use C4.5 without pruning and the product rule in probability theory to calculate probability estimates, as shown in Equation 2.

$$\hat{p}(c_j|\mathbf{A}_p) = \alpha \hat{p}(c_j) \prod_{A_i \in \mathbf{A}_p} \hat{p}(A_i|c_j, \mathbf{A}_p(A_i)), \quad (2)$$

where $\hat{p}(c_j)$ is the prior 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 (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* (PITree).

Another approach to tackle the deficiencies of traditional decision trees is to put probability density estimators at leaves. Kohavi [2] proposed a Naïve Bayes Tree (NBTree) that is a hybrid of decision tree and naïve Bayes, where a naïve Bayes is deployed at each leaf to produce classification and probability estimation. Based on the samples at a leaf, NBTree denotes $\hat{p}(c_j|s_t)$ as below:

$$\hat{p}(c_j|\mathbf{A}(L)) = \alpha \hat{p}(\mathbf{A}_1(L)|c_j, \mathbf{A}_p(L)) \hat{p}(c_j|\mathbf{A}_p(L)), \quad (3)$$

where $\mathbf{A}(L)$ is the combined set of leaf attributes $\mathbf{A}_1(L)$ and path attributes $\mathbf{A}_p(L)$. $\hat{p}(c_j|\mathbf{A}_p(L))$ is the conditional probability on path attributes. $\hat{p}(\mathbf{A}_1(L)|c_j, \mathbf{A}_p(L))$ is calculated by the naïve Bayes deployed at this leaf. From the conditional independence assumption of naïve Bayes, Equation 4 stands:

$$\hat{p}(\mathbf{A}_1(L)|c_j, \mathbf{A}_p(L)) = \prod_{i=1}^n \hat{p}(A_{li}(L)|c_j, \mathbf{A}_p(L)), \quad (4)$$

where $A_{li}(L)$ is an individual leaf attribute and n represents the number of leaf attributes.

3 Applying Lazy Learners to Decision Trees

3.1 Motivations

The main goal of our work is to optimize the ranking performance of decision trees. Traditional decision trees partition the sample space into regions and use the class frequencies to estimate probabilities. This suffers from high *bias* and high *variance*. A traditional tree inductive algorithm preferably makes leaves pure so that the resulting probabilities will be systematically shifted towards zero or one (*bias*). In addition, the probabilities generated from a leaf with a small number of samples will not be reliable (*variance*). Improvements from PETs focus on balancing between *bias* and *variance* of decision trees. For example, smoothing can reduce variance, whereas generating large trees can reduce bias. Taking this trade-off into account in this paper, we propose to use lazy learners to estimate the probabilities.

We notice that probability-based ranking is a relative metric that a precise ranking concerns not only one sample but its relative position among a set of other samples based on the class probability assigned to it. In other words, if we want to rank a set of unlabeled samples $\{s_t\}$ in class c_j , we actually sort the probabilities $\{\hat{p}(c_j|s_t)\}$. We evaluate the ranking performance of a model in terms of AUC (*the Area Under the Receiver Operating Characteristics Curve*) [6]. AUC compares learning models within the entire range of class distributions as well as error costs. Hand and Till [1] introduced a simple approach to calculate the AUC value for a binary-class sample set \mathbf{S}_{bin} given a learning model Γ as shown below:

$$AUC(\Gamma|\mathbf{S}_{\text{bin}}) = \frac{S_+ - n_+(n_+ + 1)/2}{n_+n_-}, \quad (5)$$

where n_+ and n_- are the numbers of positive and negative samples respectively, and $S_+ = \sum r_i$, where r_i is the rank of i th positive sample in the ranking list. Multi-class AUC can be computed by M -measure [1]. As one can see that AUC only demands that all the samples that belong to c_j

are ranked posterior the samples that do not belong to c_j . It is just as to group samples into two clusters: one cluster is for the samples $\{s_t|C(s_t) = c_j\}$, and the other is for the samples $\{s_t|C(s_t) \neq c_j\}$. This inspires us to estimate the probability $\hat{p}(c_j|s_t)$ according to the similarities of s_t to other samples. A lazy learner is used to give a set of unique probabilities to an unlabeled sample according to its similarities to other samples in its neighborhood. We deploy lazy learners at tree leaves. The probability estimation of such a tree is more accurate and multiple than the trees on which either m -Branch or *Ling&Yan*'s algorithm is applied, since these methods also assign a set of uniformed probabilities to any unlabeled sample that falls into the same leaf.

3.2 LDTree Model

Assume that each sample s could be represented by a feature vector $\langle a_0(s), \dots, a_n(s) \rangle$ where $a_i(s)$ is the value of i th attribute of s . Meanwhile, we suppose that all samples correspond to points in the n -dimensional space \mathcal{R}^n . Thus, given an unlabeled sample s_t and a sample s_i at a leaf L , the distance between s_t and s_i is defined as:

$$d_i(s_i, s_t) = \sqrt{\sum_{i=1}^n \delta(a_i(s_i), a_i(s_t))}, \quad (6)$$

where $\delta(a_i(s_i), a_i(s_t))$ outputs one if $a_i(s_i) \neq a_i(s_t)$, otherwise it outputs zero. The degree of similarity between s_i and s_t can be measured by Equation 7:

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

where d_{max} (d_{min}) is the maximum (minimum) distance to s_t at L . We define weight w_{s_i} of sample s_i as its similarity to s_t , i.e. $w_{s_i} = similarity(s_i, s_t)$.

For the unlabeled sample s_t , a lazy learner first finds m closest neighbors (here m means all the samples at L) for s_t by computing their distances to s_t using Equation 6, and then calculates a weight for each neighbor using Equation 7. Finally, the probability estimation of s_t is a set of normalized values of all the weights of the samples at L via Equation 8 as shown below.

$$\hat{p}(c_j|s_t) = \frac{\sum_{r=1}^m w_{s_r}^j}{\sum_{k=1}^n w_{s_k}}, \quad (8)$$

here, n represents the number of samples at L , r is the number of samples in class c_j at L , and $w_{s_r}^j$ is the weight of sample s_r that belongs to c_j .

In LDTree, we adopt a heuristic search process, in which we exhaustively build all possible trees in each step and keep only the best one for the next level expansion. Suppose that finite k attributes are available. When expanding

the tree at level q , there are $k-q+1$ attributes to be chosen. On each iteration, each candidate attribute is chosen as the root of the (sub) tree, the generated tree is evaluated, and we select the attribute that achieves the highest *gain ratio* as the next level node to grow the tree. We consider two criteria for halting the search process. We could stop splitting when none of the alternative attributes can statistically significantly upgrade the classification accuracy. Or, to avoid the “fragmentation” problem, there are at least 30 samples at the current node. Furthermore, we still permit splitting if the relative increment in accuracy is not a negative value, which makes LDTree greedier than C4.5. When an unlabeled sample is ready, we deploy a lazy learner at the leaf where the sample falls and it will output the final probability estimation of the sample as previously described. The LDTree model is depicted as in Algorithm 1.

Algorithm 1 $LDTree(S_L, s_t, \hat{p}(c|s_t))$

S_L : a set of samples at leaf L

s_t : an unlabeled sample

$\hat{p}(C|s_t)$: a set of probability estimates of s_t

Deploy a lazy learner Ψ that executes:

for each sample $s_i \in S_L$ **do**

 Calculate the distance d_i between s_i and s_t by utilizing Equation 6.

 Calculate the sample weight w_i by utilizing Equation 7 in terms of its similarity to s_t .

Within leaf L , after all samples are exhaustively iterated, for each class value c_j , summarize the weights of samples that have the same class value c_j , and use Equation 8 to do the normalization.

Return the set of normalized values as $\hat{p}(C|s_t)$ for s_t .

3.3 EDTree Model

EDTree is an improvement of LDTree. LDTree is actually a semi-lazy algorithm where the major computation is done when an unlabeled sample comes. EDTree adopts the same spirit to use the similarities between samples to calculate the probability, however it is an eager algorithm. In addition, we also want to take advantage of **large trees**, so we grow the tree in a different way. Although large trees risk more variance due to fewer samples at leaves, they could have a good capacity of discriminating samples. To solve this dilemma, we first expand a tree until, for each leaf, there are relatively enough samples. Then, for each leaf sample, a lazy learner is placed to calculate a set of probabilities for it in advance. To keep some branches that are beneficial to the ranking quality, we go on splitting the tree completely, but the final probability estimation of each new leaf is based on the sets of probabilities of leaf samples calculated from

a relatively large sample set (when the tree is still small), which relieves the variance of probability estimation.

In detail, we build up a tree using the same inductive process as LDTree, and denote it as \hat{T} . For each leaf, we apply the leave-one-out technique to choose a leaf sample as the unlabeled case and regard others as labeled cases, employ a lazy learner to calculate the distance for each labeled case and get its weight. We normalize all the weights of labeled cases with respect to their class values via Equation 8, and generate a set of probabilities for this unlabeled case. After all the samples within this leaf are iterated, we continue to grow this branch until it meets one of the limitations: **(1)** the sample set of current splitting node is pure; **(2)** there is no attribute to be used for further splitting. After constructing the tree structure, for each sample at a new tree leaf, we sum up the corresponding probability in terms of its class value, and for all class values we normalize these sums via Equation 8. The set of normalized values is regarded as the final probability estimation for this leaf. Given an unlabeled sample s_t , EDTree dispatches it into a leaf and assigns the probability estimation of that leaf to s_t as its probability estimation. The EDTree model is described as in Algorithm 2.

Algorithm 2 $EDTree(S, T)$

S : a set of samples

T : EDTree

Learn a traditional decision tree \hat{T} .

for each leaf L in \hat{T} **do**

 Apply *leave-one-out* technique to choose an unlabeled case c_t and regard any of others as labeled case c_{train} .

 Deploy a lazy learner Ψ that executes:

for each $c_{train} \in L$ **do**

 Calculate the distance d_{train} between c_{train} and c_t by utilizing Equation 6.

 Calculate the case weight w_{train} by utilizing Equation 7.

 Normalize all the case weights according to the class values of labeled cases by using Equation 8, and generate a set of probabilities $\langle p_0^{temp}, \dots, p_n^{temp} \rangle$ for c_t .

 After all labeled cases are exhaustively iterated, fully grow \hat{T} . The generated tree is denoted as T .

for each new leaf L_{new} in T **do**

 for each class value $c_j \in C$, add together the c_j probability of any c_{train} whose class value is equal to c_j . For all the sums, use Equation 8 to do the normalization.

 Assign the set of normalized values $\langle \hat{p}_0, \dots, \hat{p}_n \rangle$ to L_{new} as its final probability prediction.

 Return T

4 Experiment Setting and Results

We conducted the experiments on the basis of 34 UCI sample sets recommended by *Weka* [12], a machine learning platform. The sample sets present a wide range of domains and cover a comprehensive suite of data characteristics. Table 1 gives a brief summary. The preprocessing

Table 1. Description of sample sets.

Data Set	Size	Attr.	Classes	Missing	Numeric
anneal	898	39	6	Y	Y
anneal.ORIG	898	39	6	Y	Y
audiology	226	70	24	Y	N
autos	205	26	7	Y	Y
balance	625	5	3	N	Y
breast	286	10	2	Y	N
breast-w	699	10	2	Y	N
colic	368	23	2	Y	Y
colic.ORIG	368	28	2	Y	Y
credit-a	690	16	2	Y	Y
credit-g	1000	21	2	N	Y
diabetes	768	9	2	N	Y
glass	214	10	7	N	Y
heart-c	303	14	5	Y	Y
heart-h	294	14	5	Y	Y
heart-s	270	14	2	N	Y
hepatitis	155	20	2	Y	Y
ionosphere	351	35	2	N	Y
iris	150	5	3	N	Y
kr-vs-kp	3196	37	2	N	N
labor	57	17	2	Y	Y
letter-2000	2000	17	26	N	Y
lymph	148	19	4	N	Y
mushroom	8124	23	2	Y	N
p.-tumor	339	18	21	Y	N
sick	3772	30	2	Y	Y
sonar	208	61	2	N	Y
soybean	683	36	19	Y	N
splice	3190	62	3	N	N
vehicle	846	19	4	N	Y
vote	435	17	2	Y	N
vowel	990	14	11	N	Y
waveform-5000	5000	41	3	N	Y
zoo	101	18	7	N	Y

stages of these sample sets mainly include four steps:

1. Applying the filter of *ReplaceMissingValues* in *Weka* to replace the missing values of attributes.
2. Applying the filter of *Discretize* in *Weka* 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* in *Weka* 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*, we applied the filter of unsupervised *Resample* in *Weka*

to reselect sample set *Letter* and generate a new sample set named *Letter-2000*. The selection rate is 10%.

The AUC result on each sample set was measured via a 5-fold cross validation 5 times. Runs with various models were carried out on the same training sets and evaluated on the same testing sets. We also performed two-tailed *t*-tests [4] with a significantly different probability of 0.95 to compare our models with others, which means we speak of two results for a sample set as being “significantly different” only if the difference is statistically significant at the 0.05 level according to the corrected two-tailed *t*-test. Besides, each entry *w/t/l* in all *t*-test tables indicates 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.

To avoid the zero-frequency problem, we adopted the *Laplace* correction. More specifically, assume that there are n_{c_j} samples that have the class value as c_j , t total samples, and k class labels in a sample set. The *Laplace* correction calculates the probability $\hat{p}(c_j)$ as $\hat{p}(c_j) = \frac{n_{c_j} + 1}{t + k}$, and similarly, $\hat{p}(a_i|c_j)$ is calculated by $\hat{p}(a_i|c_j) = \frac{n_{i,c_j} + 1}{n_{c_j} + v_i}$ where v_i is the number of values of attribute A_i and n_{i,c_j} is the number of samples in class c_j with $A_i = a_i$.

We conducted three groups of experiments on AUC. In the first group (Table 2), *EDTree* and *LDTree* were compared to *C4.5* and its *PET* variants. In the second group (Table 4), we compared our models with *C4.4* and its *PET* variants. In the last group, *LDTree* with *bagging* (*LDTree-B*) was compared with bagged decision trees. We implemented *EDTree*, *LDTree*, AUC metric, *m-Branch* and *Ling&Yan*’s algorithm under the framework of *Weka*, and used the current implementations of other models and *bagging* method within *Weka*. From the experiments, we have the following observations and result analyses.

1. As shown in Table 2 and Table 3, our models achieve remarkably good performances on AUC among *C4.5* and its variants. Compared with *C4.5*, *EDTree* wins 26 sample sets and loses no sample set. In the 26 winning sample sets, there are 11 sample sets in which the AUC values of *EDTree* are 10% higher than *C4.5*. For *LDTree*, it wins *C4.5* in 24 sample sets and also loses no sample set. The variants of *C4.5* can improve the ranking performance of *C4.5*, however, our models perform significantly better than those measured by AUC (see Table 3).
2. *C4.4*, the improvement version of *C4.5* on ranking, has better AUC values than *C4.5* presented in Table 4 and Table 5. Compared to *C4.4*, our models still have better ranking performances. *EDTree* (*LDTree*) wins 17(8) sample sets and loses 0(1) sample sets against *C4.4*. Compared with *C4.4* variants, *EDTree* tremen-

dously outperforms other models on AUC and LDTree is also better than most of them (see Table 5). It is worthy of noticing that the AUC values of C4.4 variants are greatly better than C4.5 variants according to the empirical results. This means large trees could result in relatively better ranking, but to alleviate the effect of higher variance, smoothing methods, such as *m*-Branch, could be adopted to tune the probabilities of a leaf using more samples outside the leaf.

3. In Table 6 and Table 7, 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 sample in terms of its differences with an unlabeled sample. As a result, bagged trees with 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 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.
4. Other than having better results on ranking, EDTree, LDTree and LDTree-B also have the best robustness and stability among their counterpart comparators. EDTree's average standard deviation on AUC is 3.21, which is the lowest in all models; LDTree's average standard deviation (3.47) is also among the lowest ones. Compared to C4.5-B and C4.4-B, LDTree-B achieves the best score (3.16).

5 Conclusion

In this paper, we explain the deficiencies that cause classic decision trees to give inaccurate ranking and propose to resolve those issues from two aspects. (1) Placing lazy learners at leaves, which are used to weight each leaf sample for an unlabeled sample regarding the extent of similarity between them. One key observation is that deploying a lazy learner is an optimal option to trade off between *bias* and *variance* of traditional decision trees. (2) Reusing a large tree structure for obtaining finer resolution to distinguish samples but avoiding its higher variance by weighting each leaf sample when the tree leaves still have enough samples. The presented models are named LDTree and EDTree respectively. Experiments show that LDTree outperforms C4.5, C4.4 and most of their variants in terms of accurate ranking measured by AUC. EDTree is the best model among all models. In future research, we learn that naïve Bayes could also generate distinct probability estimates for different unlabeled samples. Thus, we propose to calcu-

late the class probabilities based on lazy naïve Bayes that is placed within the neighborhood of an unlabeled sample.

References

- [1] 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.
- [2] Ron Kohavi. Scaling up the accuracy of naive-bayes classifiers: a decision-tree hybrid. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, 1996.
- [3] 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.
- [4] C. Nadeau and Y. Bengio. Inference for the generalization error. *Machine Learning*, 52(40), 2003.
- [5] 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.
- [6] F. Provost and T. Fawcett. Analysis and visualization of classifier performance: Comparison under imprecise class and cost distribution. In *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining*. AAAI Press, 1997.
- [7] F. J. Provost and P. Domingos. Tree induction for probability-based ranking. *Machine Learning*, 52(30), 2003.
- [8] 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.
- [9] J. R. Quinlan. Induction of decision trees. *Machine Learning*, 2(1), 1986.
- [10] J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann: San Mateo, CA, 1993.
- [11] J. Su and H. Zhang. Probabilistic inference trees for classification and ranking. In *Proceedings of the Nineteenth Canadian Conference on Artificial Intelligence*. Springer, 2006.
- [12] I. H. Witten and E. Frank. *Data Mining –Practical Machine Learning Tools and Techniques with Java Implementation*. Morgan Kaufmann, 2000.

Table 2. Experiment results on 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), C4.5 with Ling&Yan’s algorithm (C4.5-LY) and C4.5 with bagging (C4.5-B)

Sample Set	EDTree	LDTree	C4.5	C4.5-L	C4.5-M	C4.5-LY	C4.5-B
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	90.13±3.81 ●
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	89.40±4.56 ●
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	71.75±1.75
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 ●	81.22±3.09 ●
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 ●	59.91±4.64 ●
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	64.51±8.40
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	98.24±1.29
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	85.40±5.65
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 ●	86.83±3.56
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 ●	90.70±2.66
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 ●	74.98±3.53
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 ●	80.02±3.95
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 ●	82.09±1.84 ●
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	83.63±0.50
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	83.68±0.52
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	85.37±5.66
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	79.95±9.62
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 ●	94.06±3.92
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 ●	98.99±1.01
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	99.92±0.07
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	86.16±13.18
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 ●	91.08±1.34 ●
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	84.37±8.64
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 ●	100.00±0.00
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 ●	72.28±2.79 ●
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 ●	93.99±2.18 ●
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 ●	80.68±6.49
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	99.79±0.33
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	98.52±0.46
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 ●	88.97±1.86
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	97.60±1.59
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 ●	96.00±1.01
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 ●	90.86±0.73
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	92.98±7.01
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	86.88±3.46

○, ● statistically significant improvement or degradation over EDTree

Table 3. Summary of the AUC experiment results on EDTree, LDTree and C4.5 variants.

Models	C4.5	C4.5-L	C4.5-M	C4.5-LY	C4.5-B	LDTree
C4.5-L	10/23/1					
C4.5-M	11/23/0	6/28/0				
C4.5-LY	6/26/2	2/25/7	1/24/9			
C4.5-B	18/16/0	8/25/1	5/27/2	10/23/1		
LDTree	24/10/0	10/24/0	5/25/4	13/21/0	5/25/4	
EDTree	26/8/0	19/15/0	16/18/0	17/17/0	8/26/0	10/24/0

Table 4. 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), C4.4 with Ling&Yan’s algorithm (C4.4-LY) and C4.4 with bagging (C4.4-B)

Sample Set	EDTree	LDTree	C4.4	C4.4-NL	C4.4-M	C4.4-LY	C4.4-B
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	95.48±6.60
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	93.80±8.08
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 ○	74.73±2.29
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 ●	95.00±1.47
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 ●	66.04±3.18
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	65.46±7.66
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 ●	98.69±0.86
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 ●	88.14±3.81
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 ●	85.38±3.60
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 ●	90.52±2.28 ●
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 ●	74.38±3.02
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 ●	78.73±3.17
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 ●	84.89±4.42
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	83.63±0.45
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	83.64±0.60
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	86.18±5.00
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	83.55±7.42
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 ●	95.04±2.71
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 ●	98.65±1.30
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	99.96±0.04
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	88.63±13.65
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 ●	93.26±1.03 ●
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	87.26±4.09
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 ●	100.00±0.00
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 ●	76.29±2.63 ●
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 ●	99.20±0.26
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	82.23±6.40
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	99.57±0.34
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 ●	98.74±0.43
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 ●	89.03±1.78
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	98.29±1.48
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 ●	95.88±1.03
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 ●	89.73±0.91
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	93.17±8.09
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	88.62±3.24

○, ● statistically significant improvement or degradation over EDTree

Table 5. Summary of the AUC experiment results on EDTree, LDTree and C4.4 variants.

Models	C4.4	C4.4-NL	C4.4-M	C4.4-LY	C4.4-B	LDTree
C4.4-NL	0/13/21					
C4.4-M	15/18/1	24/10/0				
C4.4-LY	3/23/8	8/22/4	2/20/12			
C4.4-B	17/17/0	27/7/0	8/26/0	15/19/0		
LDTree	8/25/1	23/11/0	1/29/4	12/22/0	0/27/7	
EDTree	17/17/0	28/6/0	11/23/0	19/14/1	3/31/0	10/24/0

Table 6. Experiment results on AUC and its standard deviation. Comparing LDTree with *bagging* (LDTree-B) with C4.5 with *bagging* (C4.5-B) and C4.4 with *bagging* (C4.4-B)

Sample Set	LDTree-B	C4.5-B	C4.4-B
anneal	96.46±4.97	90.13±3.81 ●	95.48±6.60
anneal.ORIG	93.85±7.35	89.40±4.56 ●	93.80±8.08
audiology	76.20±2.29	71.75±1.75 ●	74.73±2.29
autos	95.44±1.32	81.22±3.09 ●	95.00±1.47
balance-scale	64.32±4.11	59.91±4.64 ●	66.04±3.18
breast-cancer	66.13±8.91	64.51±8.40	65.46±7.66
breast-w	98.62±0.82	98.24±1.29	98.69±0.86
colic	88.49±4.31	85.40±5.65 ●	88.14±3.81
colic.ORIG	86.84±3.51	86.83±3.56	85.38±3.60
credit-a	90.66±2.48	90.70±2.66	90.52±2.28
credit-g	76.21±3.29	74.98±3.53 ●	74.38±3.02
diabetes	81.05±3.64	80.02±3.95 ●	78.73±3.17 ●
glass	86.98±3.75	82.09±1.84 ●	84.89±4.42
heart-c	83.81±0.42	83.63±0.50	83.63±0.45
heart-h	83.83±0.53	83.68±0.52	83.64±0.60
heart-statlog	87.18±4.86	85.37±5.66 ●	86.18±5.00
hepatitis	83.99±7.34	79.95±9.62	83.55±7.42
ionosphere	94.58±3.29	94.06±3.92	95.04±2.71
iris	98.67±1.18	98.99±1.01	98.65±1.30
kr-vs-kp	99.96±0.03	99.92±0.07	99.96±0.04
labor	92.34±9.90	86.16±13.18	88.63±13.35
letter-2000	93.63±1.12	91.08±1.34 ●	93.26±1.03
lymph	88.47±4.52	84.37±8.64	87.26±4.09
mushroom	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
sick	98.35±1.04	93.99±2.18 ●	99.20±0.26
sonar	81.66±6.38	80.68±6.49	82.23±6.40
soybean	99.59±0.43	99.79±0.33	99.57±0.34
splice	98.78±0.35	98.52±0.46 ●	98.74±0.43
vehicle	89.32±1.71	88.97±1.86	89.03±1.78
vote	98.62±1.22	97.60±1.59	98.29±1.48
vowel	96.15±1.00	96.00±1.01	95.88±1.03
waveform-5000	90.87±0.79	90.86±0.73	89.73±0.91 ●
zoo	93.62±7.66	92.98±7.01	93.17±8.09
Average	89.14±3.16	86.88±3.46	88.62±3.23

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

Table 7. Summary of the AUC experiment results on LDTree-B, C4.5-B and C4.4-B.

Models	C4.5-B	C4.4-B
C4.4-B	8/25/1	
LDTree-B	14/20/0	2/32/0