Improving the Ranking Performance of Decision Trees

Bin Wang and Harry Zhang

Faculty of Computer Science, University of New Brunswick, Canada
{bin.wang, hzhang}@unb.ca

Abstract. An accurate ranking of instances based on their class probabilities, which is measured by AUC (area under the Receiver Operating Characteristics curve), is desired in many applications. In a traditional decision tree, two obstacles prevent it from yielding accurate rankings: one is that the sample size on a leaf is small, and the other is that the instances falling into the same leaf are assigned to the same class probability. In this paper, we propose two techniques to address these two issues. First, we use the statistical technique *shrinkage* which estimates the class probability of a test instance by using a linear interpolation of the local class probabilities on each node along the path from leaf to root. An efficient algorithm is also brought forward to learn the interpolating weights. Second, we introduce an instance-based method, the weighted probability estimation (*WPE*), to generate distinct local probability estimates for the test instances falling into the same leaf. The key idea is to assign different weights to training instances based on their similarities to the test instance in probability estimation. Furthermore, we combine *shrinkage* and *WPE* together to compensate for the defects of each. Our experiments show that both *shrinkage* and *WPE* improve the ranking performance of decision trees, and that their combination works even better. The experiments also indicate that various decision tree algorithms with the combination of *shrinkage* and *WPE* significantly outperform the original ones and other state-of-the-art techniques proposed to enhance the ranking performance of decision trees.

Keywords: Decision Tree, Class Probability, Ranking, AUC, *Shrinkage*, *WPE*

1 Introduction

Decision trees have been regarded as one of the most popular models in the fields of machine learning and data mining. Traditionally, accuracy is often used to evaluate the classification performance of decision trees. However, it is not sufficient to merely classify an instance into the most possible class in many applications. A ranking of instances based on the class probability $P(c|e)$, the probability of an instance e in the class c, is more desirable. For example, a credit card company can consider the top X% in a ranking of applicants, who are most likely to belong to the profitable class. In this paper, we use AUC (Area

Under the Receiver Operating Characteristics Curve) to evaluate the ranking performance of decision trees, which has received considerable attention as a measure of ranking [14,8,6].

Accurate probability estimation certainly leads to accurate ranking which is based on the class probabilities. Unfortunately, decision trees, such as C4.5 [15], have been observed to produce poor probability estimates which result in the poor ranking performance [11,3,13]. In a decision tree, the class probability $P(c|e)$ is estimated by the fraction of instances of class c on the leaf which e falls into. It causes two problems [17]. One is the high bias: decision tree growing methods try to make leaves pure, so the probability estimates on leaves are shifted towards zero or one; the other is the high variance: the training instances on a leaf are often not enough to provide reliable probability estimates. Besides, decision tree algorithms often assign the same class probability to the instances falling into the same leaf. The ranks of the instances with equal probability are generated randomly, and thus the AUC score tends to decrease.

In this paper, we introduce *shrinkage* and the weighted probability estimation (*WPE*) to solve the above problems. The probability estimate with *shrinkage* for a test instance is decided by the linear interpolation of the local probability estimates on each node along the path from leaf to root, instead of merely being decided by the leaf. An efficient algorithm is proposed to determine the interpolating weights. *WPE* is an instance-based method, which assigns the distinct class probabilities to the instances falling into the same node, and thus leads to better ranking performance. However, there are still some flaws in *shrinkage* and *WPE*. We combine these two techniques together to compensate for the defects of each. *Shrinkage*, *WPE* and their combination are applicable to any decision tree algorithms without changing the tree-building process and tree structure. We design empirical experiments to verify that both *shrinkage* and *WPE* can improve the ranking performance of traditional decision tree algorithms, such as C4.5 [15] and C4.4 [12], in terms of AUC. We also show that the combination of these two techniques is even stronger than either single one, and according to AUC outperforms other techniques such as m-Branch [5], bagging [12] and Ling's algorithm [9], which aim to improve the probability-based ranking in decision trees.

The rest of the paper is organized as follows. In Section 2, we discuss the related work on improving the probability-based ranking performance of decision trees. In Section 3, we describe *shrinkage* and the algorithm for training the interpolating weights. In Section 4, we illuminate the process of *WPE*. In Section 5, we show how to combine *shrinkage* and *WPE* together. In Section 6, the experiments and results are presented and analyzed. Finally, we summarize our work and bring forward the future research in Section 7.

2 Related Work

As the foregoing analysis has shown, the ranking performance of C4.5 is poor (i.e., low AUC score). Provost and Domingos [12] utilize two techniques to improve the AUC of C4.5. The first is to turn off pruning and the second is to use *Laplace* correction. They call the resulting algorithm C4.4. However, turning off pruning results in a large tree so that the number of training instances on each leaf tends to be small. Then the corresponding probability estimation could be unreliable even when using *Laplace* correction. Moreover, the same probability estimate is assigned on the same leaf in C4.4. Bagging is also used to improve the AUC of decision trees [2,13], but the results produced by bagging are not comprehensible.

Some researchers have noticed that the information used for estimating the probability of an instance should not be limited to the leaf which the instance falls into. Ling and Yan [9] present an algorithm to average probability estimates from all leaves, instead of a single leaf. The contribution of each leaf is determined by the deviation in attribute values from root to leaf. But the deviation they described has only been reflected by a "confusion factor" which can be regarded as the probability of errors that alters the attribute values. Although it produces distinct probability estimates for the instances on the same leaf, setting up good "confusion factors" could still be an issue. Moreover, the algorithm should go through the whole tree to calculate the contribution of each leaf. Consequently, the complexity of this algorithm tends to be fairly high.

Ferri et al. [5] introduce the m-Branch method to smooth the probability estimates on leaves with the history information along the paths. M-Branch is a recursive process in which the probability estimate on the parent node is put into the probability estimate on the child node. The parameters of m-Branch are adjusted by the height of a node and the cardinality (the number of instances associated with a node). Although they notice that the information from other nodes should be utilized, they still assign the same class probability to the instances on the same leaf.

Zadrozny and Elkan [17] suggest a method called *curtailment*, to improve the probability estimation of decision trees. In *curtailment*, if a leaf contains few training instances and can not induce reliable probability estimates, probability estimation can be raised to an ancestor node of this leaf, in which there are enough training instances. *Curtailment* blurs the distinction between internal nodes and leaves, because a node may serve as an internal node which owns child nodes, or serve as a leaf which assigns the same probability estimate to instances. *Curtailment* is reminiscent of the methods proposed by Bahl et al. [1] and Buntine [4] that calculate a weighted average of training frequencies at nodes along the path from root to leaf. However, they do not propose an effective algorithm to learn the weights. Hastie and Pregibon [7] provide a shrinking process called *recursive shrinking* to smooth the pruning in decision trees. The shrinking process is parameterized by a scalar θ which must be specified based on the data.

Shrinkage has been brought into text classification [10], in which the word probability estimates are improved by *shrinkage* in a hierarchical structure. In this method, the final class probabilities are estimated by naive Bayes. Besides, the weights of *shrinkage* are determined by an EM algorithm. The EM algorithm needs an iterative procedure that converges usually after many iterations. Thus, it is quite time-consuming.

3 Shrinkage

Figure 1 shows a sample decision tree, which has five internal nodes N_1,...,N_5 and six leaves N_6,...,N_{11}, associated with the subsets of training instances D_1, ..., D_{11}.

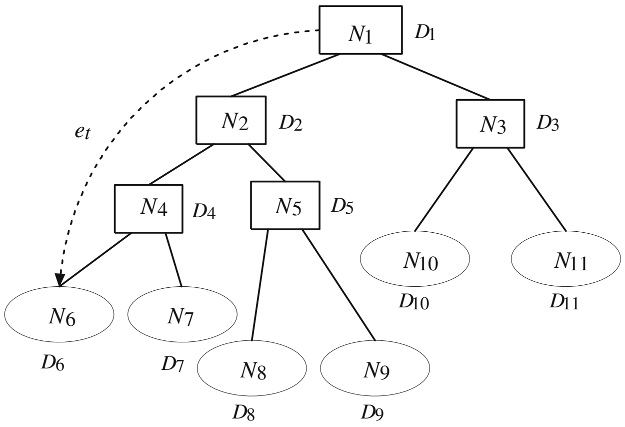

Fig. 1. A sample of decision tree

Assume that the test instance e_t falls into leaf N_6 passing internal nodes N_1, N_2 and N_4. In the traditional decision tree algorithm, the class probability $P(c_j|e_t)$ is estimated by the fraction of training instances in class c_j on leaf N_6. The *shrinkage* estimate of $P(c_j|e_t)$ is a linear combination of the local class probability estimates on the nodes N_1, N_2, N_4 and N_6. Given class c_j, the local class probability of e_t on the *ith* node is estimated as follows.

$$P^i(c_j|e_t) = \frac{n_j + 1/|C|}{|D_i| + 1}, \tag{1}$$

where n_j is the number of training instances belonging to class c_j, $|C|$ is the number of classes, and $|D_i|$ is the number of training instances on the *ith* node. In Equation 1, Laplace correction is used to smooth the probability estimate towards the uniform distribution of class labels.

Although the root N_1 contains all the training instances, it probably has few instances whose class labels are rare, which may result in unreliable probability estimation. Therefore, we extend the tree by adding a uniform node N_0 beyond the root [10], on which the uniform distribution of instances is adopted. In order to keep the consistency of expression, we define the class probability of e_t given class c_j on the uniform node N_0 as $P^0(c_j|e_t) = \frac{1}{|D_1|}$, where $|D_1|$ is the number of training instance on root N_1. The *shrinkage* estimate of $P(c_j|e_t)$ is shown as follows.

$$P(c_j|e_t) = w_j^0 P^0(c_j|e_t) + w_j^1 P^1(c_j|e_t) + w_j^2 P^2(c_j|e_t) + w_j^3 P^3(c_j|e_t) + w_j^4 P^4(c_j|e_t), \tag{2}$$

where $P^1(c_j|e_t)$, $P^2(c_j|e_t)$, $P^3(c_j|e_t)$, $P^4(c_j|e_t)$ are the local probability estimates on N_1, N_2, N_4, N_6 respectively, w_j^0, w_j^1, w_j^2, w_j^3, w_j^4 are the interpolating weights for class c_j assigned to the corresponding nodes, in which $\sum_{m=0}^{4} w_j^m = 1$. Figure 2 shows the path, probabilities and related weights. Equation 2 can be extended to deal with the general case which has k nodes on the path.

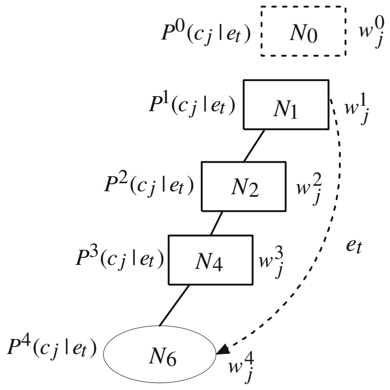

Fig. 2. The path of e_t in the decision tree. Uniform node is added above N_1. The local probabilities are estimated on each node, and the interpolating weights are assigned to each node.

Shrinkage represents a tradeoff between the specificity and generality. At a leaf, since the probability estimates are yielded based on the training instances that come through a series of partitions on the internal nodes, they are more specific but less general than the ones on the ancestors of the leaf. At the root, the estimates are more general because all the training instances are included, but they are less specific than the estimates on the descendants of the root.

A key problem to apply *shrinkage* is to determine the interpolating weights effectively and efficiently. Assume that N_0, N_1, ..., N_k is a path, where N_0 is the uniform node, N_1 is the root, and N_k is the leaf. Given class label c_j, let β_j^0, β_j^1, ..., β_j^k be the influence degree of node N_i to class c_j, and w_j^0, w_j^1, ..., w_j^k be the interpolating weights. We use the following algorithm to determine the interpolating weights.

Algorithm DIW (*path*, D, c_j)
Input: *path* is a sequence of nodes N_0, N_1, ..., N_k, D is the training set, and c_j is a class label.
Ouput: A set of interpolating weights for c_j for all nodes N_0, N_1, ..., N_k.

Step 1: Initialize each weight w_j^i as $\frac{1}{k+1}$ so that $\sum w_j^i = 1$, and initialize each degree β_j^i to zero. Set each training instance on leaf unmarked.

Step 2: Choose an unmarked training instance x on leaf N_k, remove x from all the training subsets $D_1, \ldots, D_k$ on each node along the path.

Step 3: Set $P^0(c_j|x) = 1/|D_1|$, where $|D_1|$ is the number of training instances on root N_1. From N_1 to N_k, estimate the local class probabilities $P^i(c_j|x)(i = 1, \ldots, k)$ using Equation 1.

Step 4: For each node, update its degree as follows:

$$\beta_j^i = \beta_j^i + \frac{w_j^i P^i(c_j|x)}{\sum_{m=0}^{k} w_j^m P^m(c_j|x)}, i = 0, \ldots, k. \tag{3}$$

Step 5: Mark instance x and put x back to each training subset. If there is an un-marked training instance on N_k, go back to **Step 2**.

Step 6: Compute w_j^i by normalizing the set of degrees $\{\beta_j^0, \beta_j^1, \ldots, \beta_j^k\}$ as follows:

$$w_j^i = \frac{\beta_j^i}{\sum_{m=0}^{k} \beta_j^m}, i = 0, \ldots, k. \tag{4}$$

Return: $\{w_j^0, w_j^1, \ldots, w_j^k\}$.

Note that on a node N_i, the local class probabilities $P^i(c_j|x)(x \in D_k)$ have the same estimate in **Step 3**, since the fraction of training instance in class c_j on N_i is used in Equation 1. Because of this, Algorithm **DIW** is not able to be adapted to a multiple-iteration algorithm. The interpolating weights are returned only after one iteration (go through each instance on the leaf once).

When we apply *shrinkage* to a decision tree algorithm, a decision tree is built by that algorithm first. Then the Algorithm **DIW** is applied to each path to set up the interpolating weights for each node and each class label. Given a test instance e_t, it is sorted down to a leaf, and then its class probability $P(c_j|e_t)$ is computed using Equation 2.

4 Weighted Probability Estimation

As mentioned before, a major issue for decision trees is that all the instances falling into the same leaf will have the same probability estimate, which is an obstacle to yielding accurate ranking. We notice that the instance-based method can generate the distinct and local estimates.

We introduce *WPE*, an instance-based method, to estimate the class probabilities on the leaves. Given a test instance e_t which consists of a set of attribute values and a class label, e_t falls into a leaf L. We define the similarity between e_t and a training instance e_r on L as follows.

$$sim(e_t, e_r) = \sum_{i=1}^{n} equ(A_i(e_t), A_i(e_r)), \tag{5}$$

where n is the number of attributes, $equ(a,b)$ is a boolean function whose value is either 1 ($a = b$) or 0 ($a \neq b$), and $A_i(e)$ is the *ith* attribute value of e. In *WPE*, we calculate $sim(e_t, e_r)$ for each training instance e_r on L as the weight of e_r and estimate $P(c_j|e_t)$ as follows.

$$P(c_j|e_t) = \frac{\sum_{e_r \in D_L, C(e_r)=c_j}(sim(e_t,e_r)+1) + 1/|C|}{\sum_{e_r \in D_L}(sim(e_t,e_r)+1) + 1}, \tag{6}$$

where D_L is the training instance subset on L, $C(e_r)$ is the class label of e_r. At the numerator of Equation 6, the total weights of the instances which belong to class c_j in D_L is computed, and the denominator is roughly the total weights of all the instances in D_L. Equation 6 uses the same *Laplace* correction as Equation 1.

Without changing the process of building a decision tree, we return the class probabilities from the leaves, which are estimated by *WPE*. Note that, for different test instances e_{t1} and e_{t2} falling into the same leaf, $P(c_j|e_{t1})$ and $P(c_j|e_{t2})$ could be different based on Equation 6. This means that we could make distinct probability estimates for the instances on the same node, which is the key to obtaining accurate ranking of instances.

5 Combination of *Shrinkage* and Weighted Probability Estimation

Both *shrinkage* and *WPE* are able to make up for the deficiencies of decision tree algorithms. *Shrinkage* breaks the restriction that the probabilities only can be estimated on the leaves, and *WPE* solves the problem that the different test instances falling into the same leaf are assigned to the same class probability estimate. However, they still suffer from some problems. In Algorithm **DIW**, the same probability estimate is assigned to different instances on the same node in **Step 3**. The generated interpolating weights for *shrinkage* are not effective enough. Moreover, when the sizes of training subsets on leaves are small, *WPE* could not generate reliable class probability estimates without the support of other nodes on the paths. The combination of *shrinkage* and *WPE* could compensate for the weakness of each. We illuminate the whole process of decision tree algorithm with the combination as follows.

Training:
First, a decision tree is built by a traditional decision tree algorithm. Then, Algorithm **DIW** is carried out to train the interpolating weights for *shrinkage*. We use *WPE* to estimate the class probabilities $P^i(c_j|x)$ for a training instance x on the leaf in **Step 3** of Algorithm **DIW**. Here x is treated as a test instance, and $P^i(c_j|x)$ is estimated by Equation 6 (instead of Equation 1). When the similarity is calculated using Equation 5, we compare all of the attribute values and the class labels, since the class label is known for x.

Testing:
Given a test instance e_t, it is sorted along a path from the root to a leaf. The local class probabilities $P^i(c_j|e_t)$ are estimated by *WPE* (Equation 6). We do not use

the class label to calculate the similarity in Equation 5, since the class label is unknown for e_t. Finally, the returned class probability $P(c_j|e_t)$ is estimated by *shrinkage* (Equation 2) with the interpolating weights determined by Algorithm **DIW** and the local probabilities estimated by *WPE*.

Using *WPE* in **Step 3** of Algorithm **DIW**, the distinct probability estimates are assigned to the instances on the same node so that the returned interpolating weights are more effective. This also makes it possible that Algorithm **DIW** can be adapted to a multiple-iteration algorithm, which terminates when the interpolating weights are converged. The multiple-iteration algorithm appears like an EM algorithm. However, a typical EM algorithm takes thousands of iterations to converge, which is fairly time-consuming. In the experiments, we still use the single-iteration Algorithm **DIW**. The experimental results show that the outcome of weights after many iterations is not significantly better than the one from just one iteration.

6 Experiments and Results

Our experiments are conducted on 35 data sets from Weka [16], which come from the UCI repository. We download these data sets in the format of $arff$ from the website of Weka. There are some preprocessing stages adopted on each data set. First, we use the filter $ReplaceMissingValues$ in Weka to replace the missing values of attributes in each data set. Second, we use the filter *Discretize*, the unsupervised ten-bin discretization in Weka, to discretize numeric attributes. Thus, all the attributes are treated as nominal. Third, we notice that, if the number of values of an attribute is almost equal to the number of instances in a data set, this attribute does not contribute any information to the purpose of prediction. So we use the filter *Remove* in Weka to delete this type of attribute.

In the first experiment, we compare the algorithms, such as C4.5 with combination of *shrinkage* and *WPE* (C45-C), C4.5 with *shrinkage* (C45-S), C4.5 with *WPE* (C45-W), C4.5 (C45), C4.5 with m-Branch (C45-M), C4.5 with LingYan's algorithm (C45-L), C4.5 with bagging (C45-B). In the second experiment, we compare the algorithms, such as C4.4 with the combination of *shrinkage* and *WPE* (C44-C), C4.4 with *shrinkage* (C44-S), C4.4 with *WPE* (C44-W), C4.4 (C44), C4.4 with m-Branch (C44-M), C4.4 with LingYan's algorithm (C44-L), C4.4 with bagging (C44-B). We implement *shrinkage*, *WPE*, the combination[1], C4.4, m-Branch, LingYan's algorithm and AUC evaluation within the Weka framework, and use the implementation of C4.5 and bagging in Weka. In all experiments, the AUC score of an algorithm on a data set is obtained via 5 runs of ten-fold cross validation. Runs with the various algorithms are carried out on the same training data sets and evaluated on the same test data sets. Finally, we conduct two-tailed t-test with a 95% confidence level to compare each pair of algorithms.

[1] Codes for *shrinkag*, *WPE* and their combination are available at `http://www.cs.unb.ca/~hzhang/ShrinkageCode.rar`

Table 3 and Table 4 show the AUC scores of the algorithms on each data set. The two-tailed t-test results are shown in Table 1 and Table 2. Each entry of Table 1 and Table 2 has the format of $w/t/l$. It means that compared with the algorithm in the corresponding column, the algorithm in the corresponding row wins in w data sets, ties in t data sets and loses in l data sets. From our experiments, we have the following observations:

– C45-S and C44-S outperform C45 and C44 respectively. C45-W and C44-W outperform C45 and C44 respectively.
– Either *shrinkage* or *WPE* is not perfect compared to some techniques. C45-S is worse than C4.5 with other techniques. C44-S is not as good as C44-M and C44-B. C44-W is worse than C4.4 with other techniques.
– C45-C outperforms C45 and C4.5 with any other techniques. C44-C outperforms C44 and C4.4 with any other techniques.

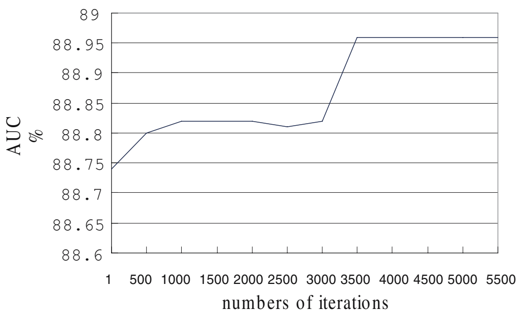

Fig. 3. The AUC scores corresponding to the different numbers of iterations

The experimental results are not surprising. Due to assigning the same local probability estimates on each node, Algorithm **DIW** could not return more effective interpolating weights, so *shrinkage* is worse than other techniques. Although *WPE* is able to estimate the probability distinctly, it suffers from the case when the sizes of training subsets on leaves are small, and that's why C44-W is worse than C4.4 with other techniques. The combination of *shrinkage* and *WPE* solves the above problems: First, applying *WPE* in Algorithm **DIW**

Table 1. Summary of comparisons for the algorithms related with C4.5

	C45-C	C45-S	C45-W	C45	C45-M	C45-L
C45-S	0/5/30					
C45-W	3/18/14	19/16/0				
C45	0/5/30	3/18/14	0/4/31			
C45-M	2/9/24	8/27/0	1/17/17	16/19/0		
C45-L	0/10/25	14/15/6	2/17/16	19/13/3	10/18/7	
C45-B	2/7/26	14/18/3	4/17/14	24/11/0	10/22/3	7/20/8

Table 2. Summary of comparisons for the algorithms related with C4.4

	C44-C	C44-S	C44-W	C44	C44-M	C44-L
C44-S	2/14/19					
C44-W	0/12/23	3/24/8				
C44	1/11/23	0/11/24	4/20/11			
C44-M	2/14/19	7/24/4	7/27/1	16/19/0		
C44-L	0/14/21	4/24/7	10/16/9	14/16/5	7/20/8	
C44-B	2/17/16	9/24/2	7/27/1	19/16/0	6/29/0	9/25/1

Table 3. Experimental results on AUC for the algorithms related with C4.5

Data set	C45-C	C45-S	C45-W	C45	C45-M	C45-L	C45-B
anneal	95.91	90.45	95.24	83.45	89.73	93.70	86.49
anneal.ORIG	95.61	91.42	94.18	86.00	90.04	91.33	86.53
audiology	70.66	62.81	70.08	61.54	62.68	69.98	64.86
autos	95.07	93.11	95.22	73.84	93.95	89.93	77.91
balance-scale	88.00	56.01	63.34	52.72	54.94	67.76	59.20
breast-cancer	92.28	62.89	92.21	60.86	62.73	67.54	64.97
breast-w	99.45	94.62	98.64	96.43	97.36	98.24	98.12
colic	95.01	85.42	91.35	81.17	85.41	85.63	85.32
colic.ORIG	92.42	84.50	90.25	83.56	85.38	80.96	87.68
credit-a	96.46	89.71	94.54	88.17	89.82	91.26	91.31
credit-g	82.49	72.90	75.35	68.48	72.32	75.08	74.16
diabetes	93.68	77.69	85.47	76.26	78.05	79.33	79.74
glass	89.61	81.93	84.60	77.11	79.88	82.46	81.26
heart-c	84.25	83.27	83.89	83.16	83.31	83.85	83.75
heart-h	84.58	81.01	84.40	80.91	81.02	83.76	83.77
heart-statlog	92.98	84.65	88.01	81.65	85.48	88.17	86.39
hepatitis	91.42	70.61	90.78	70.43	70.50	72.90	81.54
hypothyroid	95.12	66.50	98.33	68.56	68.74	82.05	83.31
ionosphere	96.32	84.29	94.25	88.98	90.57	88.32	94.72
iris	99.68	99.12	99.41	98.99	98.67	98.12	99.00
kr-vs-kp	99.73	99.56	99.93	99.81	99.88	98.90	99.91
labor	97.50	82.75	93.50	78.25	82.75	85.42	84.46
lymph	88.70	86.47	85.90	71.16	86.29	87.33	83.02
mushroom	100.00	100.00	100.00	100.00	100.00	99.46	100.00
primary-tumor	75.93	69.73	72.67	63.98	70.67	74.29	69.01
segment	98.60	97.81	99.24	98.47	98.97	96.19	99.33
sick	99.12	93.69	96.67	93.42	94.48	94.90	94.01
sonar	86.84	75.05	76.26	69.49	74.14	75.33	81.92
soybean	99.73	98.95	99.15	98.12	98.96	99.50	98.99
splice	98.70	97.88	98.12	96.84	98.21	98.90	98.52
vehicle	88.74	84.59	86.66	83.13	87.57	81.49	89.03
vote	98.69	96.20	98.60	96.57	97.53	98.23	97.64
vowel	97.20	94.65	95.32	92.83	95.61	92.96	96.40
waveform-5000	91.71	85.79	87.06	84.63	88.54	90.92	91.03
zoo	88.48	80.38	88.00	79.57	80.29	85.76	80.62

can generate the different probability estimates on each node for each training instances on the leaf; Second, *shrinkage* balances the probability estimation towards the nodes with large training subsets; Third, the class probability for a test instance is estimated by the local probabilities from *WPE* and *shrinkage* weights from Algorithm **DIW**.

In another interesting experiment, we adapt Algorithm **DIW** in C45-C to a multiple-iteration algorithm. For data set “Vehicle”, the AUC scores from the different numbers of iterations are plotted in Figure 3. We observe that the AUC score is not changed at 88.96% after 3500 iterations. Compared with the AUC score 88.74% from a single iteration, we can see that the result after many iterations is not significantly better than the one from a single iteration.

Table 4. Experimental results on AUC for the algorithms related with C4.4

Data set	C44-C	C44-S	C44-W	C44	C44-M	C44-L	C44-B
anneal	96.02	94.71	94.77	93.95	94.13	93.67	94.83
anneal.ORIG	95.61	93.54	93.43	92.11	93.56	92.25	93.25
audiology	70.28	67.42	70.08	65.91	67.00	70.45	69.69
autos	95.12	93.68	94.92	91.28	94.31	90.43	94.91
balance-scale	79.50	57.76	66.79	59.42	57.90	66.45	65.15
breast-cancer	76.51	61.01	62.60	59.44	62.43	67.78	64.09
breast-w	99.30	96.58	98.59	98.08	98.18	98.27	98.79
colic	90.90	87.15	85.29	84.13	87.22	85.69	88.06
colic.ORIG	89.36	83.80	83.45	82.43	83.60	80.66	86.00
credit-a	94.40	90.99	89.43	88.30	91.31	91.26	90.42
credit-g	79.21	73.06	69.67	68.06	72.70	75.04	73.44
diabetes	89.04	78.91	77.74	74.66	78.65	79.98	78.78
glass	88.77	83.11	80.29	80.57	81.23	82.06	79.52
heart-c	84.03	83.29	83.35	83.19	83.47	83.84	83.65
heart-h	84.13	83.52	83.45	83.21	83.67	83.82	83.65
heart-statlog	90.14	85.24	84.33	82.82	85.59	88.66	86.73
hepatitis	86.62	81.29	81.62	78.64	80.76	77.92	83.03
hypothyroid	85.60	86.31	83.13	82.23	83.74	81.40	82.28
ionosphere	94.28	88.86	93.18	92.17	92.46	91.28	95.17
iris	99.65	98.00	98.93	98.52	98.85	97.33	98.68
kr-vs-kp	99.74	99.65	99.96	99.95	99.91	98.75	99.97
labor	95.42	82.96	86.58	84.63	86.29	84.04	89.42
lymph	88.78	88.12	87.04	86.44	87.22	87.61	88.08
mushroom	100.00	100.00	100.00	100.00	100.00	99.46	100.00
primary-tumor	76.05	72.48	72.10	69.23	72.96	74.44	73.07
segment	99.09	98.79	99.35	99.20	99.29	95.88	99.53
sick	98.92	97.87	99.20	99.11	99.19	94.41	99.24
sonar	83.30	78.44	79.32	77.35	78.65	75.57	82.58
soybean	99.76	99.08	98.93	98.12	99.01	99.54	98.90
splice	98.74	98.10	98.19	97.90	98.50	98.93	98.70
vehicle	88.56	85.32	86.76	86.19	87.84	82.06	88.91
vote	98.72	96.53	98.50	97.62	97.90	98.83	98.45
vowel	97.43	95.48	94.81	91.37	96.14	93.03	96.33
waveform-5000	90.55	87.46	83.50	80.85	86.98	90.77	89.87
zoo	88.48	81.00	88.00	80.57	80.81	87.05	81.19

7 Conclusions and Future Work

In this paper, we present a statistical technique, *shrinkage*, and an instance-based method, *WPE*, to improve the ranking performance of decision trees, which is measured by AUC. The class probability estimate with *shrinkage* is a linear interpolation for the local probability estimates on each node along the path from leaf to root. Algorithm **DIW** is proposed to determine the interpolating weights used in *shrinkage*. *WPE* produces the distinct probability estimates for the instances on the same node. In order to compensate for the deficiencies of *shrinkage* and *WPE*, we combine them together. In this process, we use *WPE* to generate distinct probability estimates on each node in training and testing processes, and also use *shrinkage* to return the final class probability estimate. The experiments show that decision tree algorithms with *shrinkage* and *WPE* outperform the original ones, and that the decision tree algorithms with this combination significantly outperform the original ones and other state-of-the-art techniques proposed to enhance the ranking performance of decision trees.

In our future research, we will study other local probability estimation methods in decision trees. We notice that naive Bayes model also generates distinct probability estimates for different instances falling into the same node. In *shrinkage*, deploying naive Bayes model along a path to estimate class probabilities may produce good ranking performance.

References

1. Bahl, L., Brown, P., deSouza, P., Mercer, R.: A tree-based statistical language model for natural language speech recognition. IEEE Transactions on Acoustics, Speech and Signal Processing. (1989) bf 37, 1001-1008
2. Bauer, E., Kohavi, R.: An empirical comparison of voting classification algorithms: Bagging, boosting and variants. Artificial Intelligence **36** (1989) 105-142
3. Bradley, A. P.: The use of the area under the ROC curve in the evaluation of machine learning algorithms. Pattern Recognition **30** (1997) 1145-1159
4. Buntine, W.: Learning classification trees. Artificial Intelligence frontiers in statistics. Chapman & Hall, London (1993) 182–201
5. Ferri, C., Flach, P. A., Hernandez-Orallo, J.: Improving the AUC of Probabilistic Estimation Trees. Proceedings of the 14th European Conference on Machine Learning (ECML2003). Springer (2003)
6. Hand, D. J., Till, R. J.: A Simple Generalisation of the Area Under the ROC Curve for Multiple Class Classification Problems. Machine Learning **45** (2001) 171-186
7. Hastie, T., Pregibon, L.: Shrinking Trees. AT & T Bell Laboratories (1990)
8. Ling, C. X., Huang, J., Zhang, H.: AUC: a statistically consistent and more discriminating measure than accuracy. Proceedings of 18th International Conference on Artificial Intelligence (IJCAI-2003). Morgan Kaufmann (2003) 329-341
9. Ling, C. X., Yan, R. J.: Decision tree with Better Ranking. Proceedings of the Twentieth International Conference on Machine Learning (ICML2003). AAAI Press (2003)
10. McCallum, A., Rosenfeld, R., Mitchell, T., Ng, A. Y.: Improving text classification by shrinkage in a hierarchy of classes. Proceedings of the 15th International Conference on Machine Learning. Morgan Kaufmann (1998) 359-367
11. Pazzani, M., Merz, C., Murphy, P., Ali, K., Hume, T., Bunk, C.: Reducing misclassification costs. Proceedings of the Eleventh International Conference on Machine Learning. Morgan Kaufmann (1994) 217-225
12. Provost, F., Domingos, P.: Tree Induction for Probability-based Ranking. Machine Learning. Kluwer Academic Publishers (2002)
13. Provost, F., Fawcett, T., Kohavi, R.: The case against accuracy estimation for comparing induction algorithms. Proceedings of the Fifteenth International Conference on Machine Learning. Morgan Kaufmann (1999) 445-453
14. Provost, F., Fawcett, T.: Analysis and visualization of classifier performance: Comparison under imprecise class and cost distributions. Proceedings of the Third International Conference on Knowledge Discovery and Data Mining (KDD-97). AAAI Press (1997) 43-48
15. Quinlan, J. R.: C4.5 Programs for Machine Learning. Morgan Kaufmann, San Mateo, CA (1993)
16. Witten, I. H., Frank, E.: Data mining-practical machine learning tools and techniques with java implementation. Morgan Kaufmann, San Mateo, CA (2000)
17. Zadrozny, B., Elkan, C.: Obtaining calibrated probability estimates from decision trees and Naive Bayesian classifiers. Proceedings of the 18th International Conference on Machine Learning. Morgan Kaufmann (2001) 609-616