Decision Tree with Better Ranking

Charles X. Ling CLING@CSD.UWO.CA
Robert J. Yan JYAN@CSD.UWO.CA
Department of Computer Science, The University of Western Ontario, London, Ontario, Canada N6A 5B7

Abstract

AUC (Area Under the Curve) of ROC (Receiver Operating Characteristics) has been recently used as a measure for ranking performance of learning algorithms. In this paper, we present a novel probability estimation algorithm that improves the AUC value of decision trees. Instead of estimating the probability at the single leaf where the example falls into, our method averages probability estimates from all leaves of the tree. The contribution of each leaf is determined by the deviation in attribute values from the root to the leaf. We design empirical experiments to verify that our new algorithm outperforms C4.5 and its recent improvement C4.4 in AUC. Even though C4.4 with bagging outperforms our method in AUC, our method produces a single tree with interpretable results.

1. Introduction

Traditionally, machine learning researchers have been using accuracy (percentage of examples that are correctly classified) as the main evaluation criterion for the performance of classifiers (Shavlik et al., 1991). However, accuracy completely ignores probability estimations produced by classifiers. As many real-world applications require probability estimations or ranking, accuracy is not sufficient in measuring and comparing classifiers. As the true ranking of training examples (Cohen et al., 1999) is often unknown, given training and testing examples with only class labels, we need a better measure for classifiers that produce scores for ranking.

The area under the ROC (Receiver Operating Characteristics) curve, or simply AUC, has been shown as one of the measures for the quality of ranking (Bradley, 1997; Ling et al., 2003; Hand & Till, 2001). Hand and Till (2001) suggest a straightforward method to calculate AUC based on examples ranked by their probability estimation:

$$AUC = \frac{S_0 - n_0(n_0+1)/2}{n_0 n_1}$$

where S_0 is the sum of ranks of the positive examples and n_0 and n_1 are the numbers of positive and negative examples, respectively. (In this paper, we only consider the binary classification.) From the equation above, we can see that AUC essentially measures the quality of ranking: the more the positive examples are ranked to the right of the list, the larger the S_0 and the AUC score.

Decision trees built by C4.5 (Quinlan, 1993) have been observed to produce poor estimates of class probabilities and ranking (Smyth et al., 1995; Provost et al., 1998). In this paper, we describe a new tree evaluation algorithm to improve the probability estimation (and ranking) from a single tree. Instead of estimating the probability at the single leaf where the example falls into, our new evaluation algorithm averages probability estimates from all leaves of the tree. The contribution of each leaf in the average is determined by the deviation in attribute values from the root to the leaf. We design empirical experiments to verify that our new algorithm outperforms C4.5 and its recent improvement C4.4 (Provost & Domingos, 2003) in AUC. Even though C4.4 with bagging still outperforms our method in AUC, our method produces a single tree with interpretable rules. We also show that our new algorithm is robust against the parameters of the algorithm, making it more versatile and useful in real-world applications.

The paper is organized as follows. In Section 2 we discuss related work on improving ranking based on probability estimations in decision trees. We then describe our new tree evaluation algorithm which improves the probability estimation measured (indirectly) by

Proceedings of the Twentieth International Conference on Machine Learning (ICML-2003), Washington DC, 2003.

AUC in Section 3. We report results of comprehensive experiments using both simulated and real-world datasets in Section 4. Finally, we summarize our work and discuss the directions for future research.

2. Related Work

Some traditional decision tree algorithms, such as C4.5 and ID3, have been observed to produce poor estimates of probabilities, which result in low AUC (Provost et al., 1998). One of the reasons is that the algorithms aim at building a small and accurate tree which biases against good probability estimates (Provost & Domingos, 2003).

Researchers have studied how to improve probability estimations of decision trees (Hastie & Pregibon, 1990; Pazzani et al., 1994; Smyth et al., 1995; Margineantu & Dietterich, 2001; Provost & Domingos, 2003). In particular, Provost and Domingos (2003) apply the following techniques to improve the AUC of C4.5.

1. **Turn off pruning.** Error-based pruning has been proved successful at building a small and accurate (in terms of accuracy) decision tree. However, Provost and Domingos (2003) show that pruning also reduces the quality of the probability estimation. This is because the branches removed by pruning may still be useful for probability estimations. For this reason, they choose to build the tree without pruning, resulting in substantially large trees.

2. **Smooth probability estimates by the Laplace correction.** Without pruning, the decision tree becomes large and has more leaves. In this case, there are fewer testing examples falling into one leaf. Thus, a single leaf may comprise only a small number of examples (for example, 2). This leaf may produce probabilities of extreme values (e.g., 100%). In addition, it cannot provide reliable probability estimations. For this reason, Provost and Domingos (2003) use the Laplace correction to smooth the estimation and make it less extreme.

They call the resulting algorithm C4.4, show that it produces decision trees with higher AUC than C4.5 (Provost & Domingos, 2003).

One of the anticipated consequences of these changes is that the tree becomes huge. Although they use the Laplace correction to smooth the estimation, there are still two problems for the probability estimation in the large tree:

1. The tree may overfit the training examples. The probabilities estimated by such an overfitting tree may not be accurate.

2. The number of training examples in the leaves of a large tree is often very small. This makes the probability estimation less reliable. In addition, values of probabilities can easily repeat, which may decrease the quality of ranking based on them. For example, if many leaves have 3 examples, many repeated probabilities will be produced. Examples with the same probability estimates will be ranked randomly.

Another approach for improving the class probability estimations from decision trees is bagging (Bauer & Kohavi, 1999; Provost et al., 1998). Provost and Domingos (2003) show that bagging is very effective for improving AUC in C4.4. However, a disadvantage of bagging is that it does not produce comprehensible results.

3. New Algorithm For Improving Ranking

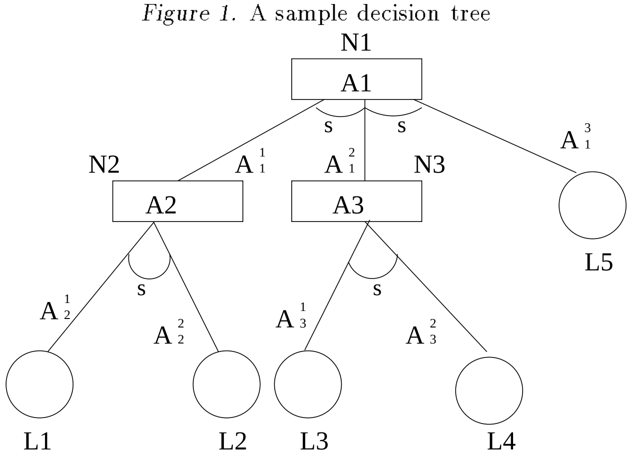
Let us first analyze the reason that C4.5 produces poor probability estimations. C4.5 (with pruning) is biased towards building small trees with fewer leaves. Lower branches of the tree are pruned if the accuracy is improved. This often results in incorrect probability estimates in leaves. Another consequence of a small tree is that it can only have a small number of distinctive probability values. Building a large tree by turning off pruning (as in C4.4) produces a large tree with many leaves. However, that is not ideal either, as the tree will likely be overfitted. A large tree may still have few different probabilities because the probability values of leaves can easily repeat themselves (see Section 2). In addition, probability estimation in leaves with small numbers of examples is not reliable.

What we hope to get is a moderate-sized decision tree with more accurate probability estimations. We notice that in decision trees, the class label and probability estimate of an example is solely determined by the one leaf where the example falls into. It is natural to expect that there may be some small chance that attribute values might have been altered due to noise or error in data collection, and thus the example would have fallen into other leaves. Therefore, the probability estimate should be aggregated from other leaves. This would improve the probability estimations, and increase the number of distinctive probability values in the tree.

3.1. New Algorithm for Probability Estimation

Assume that in a decision tree (see an example in Figure 1), there are some different internal nodes N_i and leaves L_i, each internal node is associated with an attribute A_i, and each A_i has several values A_i^j. In our method, we introduce another parameter – confusion factor s ($s < 1$) to each internal node.[1] The confusion factor denotes the amount of "confusion" among the children of the node. It also can be regarded as the probability of errors that alter the attribute values. Thus when a testing example is classified by a tree, it has a small probability (s) of going down to other branches of the tree. We assume that attribute errors are independent, thus the probability of the example falling into another leaf is the multiplication of the confusion factor, by the number of times that the attribute values have deviated.

Figure 1. A sample decision tree



Thus, in the new algorithm, the probability of every leaf can contribute to the final probability of an example. The contribution is decided by the number of unequal attribute values that the leaf has, compared to the example. Each unequal attribute value will cause a discount on the probability contribution in terms of s. Here the attribute values of a leaf are the attribute values in the path from this leaf to the root. For example, in Figure 1, leaf L_1 (with probability $p(L_1)$) has two attribute values $A_1 = A_1^1$ and $A_2 = A_2^1$. If the testing example e falls into leaf L_1, the contribution of L_1 to e is $1 \times p(L_1)$. On the other hand, the contribution of L_2 to e is $s \times p(L_2)$ because L_2 has one different attribute value. The contribution of leaf L_3 to e depends on the value of attribute A_3 of e. If $e_{A_3} = A_3^1$, L_3 has one different value and the contribution of L_3 to e is $s \times p(L_3)$. If $e_{A_3} = A_3^2$, the contribution of L_3 to e is $s \times s \times p(L_3)$. The final probability P_e for the example e is a weighted average of all the contributions of leaves:

$$P_e = \frac{\sum P_i \times s^j}{\sum s^j}$$

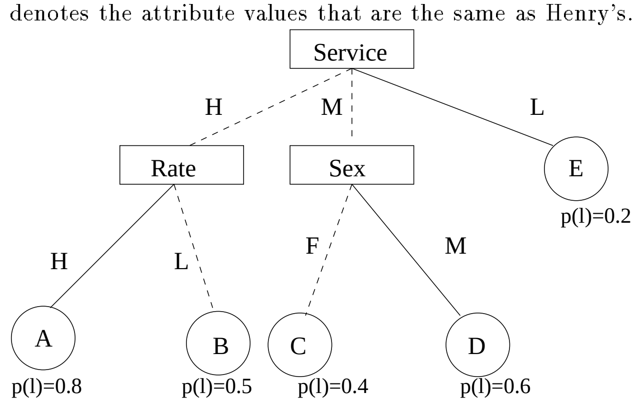
where P_i denotes the probability of leaf L_i, and j denotes the number of attribute values in the path from leaf L_i to the root that are different from the attribute values of the example e.

Let's see a hypothetical example (Figure 2). This decision tree has five leaf nodes (A, B, C, D, and E), each with a probability of customer loyalty (positive). We hope to predict the loyalty of the following two customers by this tree:

- Customer Henry, with Service (service level) being L (low), Rate (mortgage rate) being H (high) and Sex being M (male)
- Customer Lily, with Service being L, Rate being L and Sex being F (female).

According to this tree, Henry and Lily both fall into the same leaf E. C4.5 predicts that both of them have a 20% chance of being loyal. Let us calculate Henry's probability by our new method. As shown in Figure 2 (we assume that s equals 0.2), we first calculate the contribution of leaf A. It is easy to find that there is one different attribute value (Service) between leaf A and Henry. Thus we multiply one confusion factor $s = 0.2$ as a discount to its probability 0.8. The contribution of leaf A is $0.2 \times 0.8 = 0.16$. Then we calculate the contributions of leaf B, C, D, and E in a similar manner. Finally, we calculate the final probability of Henry's loyalty:

Figure 2. Henry's probability. A dotted line denotes the attribute values that are different from Henry's. A solid line denotes the attribute values that are the same as Henry's.



P_{Henry}:

$$\frac{0.2 \times 0.8 + (0.2 \times 0.2) \times 0.5 + (0.2 \times 0.2) \times 0.4 + 0.2 \times 0.6 + 1 \times 0.2}{0.2 + 0.04 + 0.04 + 0.2 + 1}$$

[1] In order to simplify our new algorithm, all attributes are assigned the same confusion factor value.

$$= 0.3486$$

Lily's probability can be similarly calculated as:

P_{Lily}:

$$\frac{(0.2 \times 0.2) \times 0.8 + 0.2 \times 0.5 + 0.2 \times 0.4 + (0.2 \times 0.2) \times 0.6 + 1 \times 0.2}{0.04 + 0.2 + 0.2 + 0.04 + 1}$$

$$= 0.2865$$

Our new method can thus assign more accurate and different probabilities to examples belonging to the same leaf.

Our algorithm can be easily extended to multiple-class datasets. In this case, an important property that the sum of probabilities of an example e belonging to different classes is equal to 1 remains unchanged.

3.2. Interpretability

Because our new evaluation algorithm is based on a single decision tree, the outcome is still interpretable. From the datasets used in extension experiments (see Section 4), we observe that the rate of change in classification (from positive to negative or vice versa) is about 10-15%. That is, for most cases, the more accurate probability estimation obtained by our new algorithm does not "flip" the classification prediction. Thus, if the class prediction is not altered, one can use the single decision tree just as in C4.5 (or C4.5 rules). In the example of Henry (see Figure 2), the explanation would simply be: as you satisfy the rule "IF Service=L THEN not Loyal", we predict that you would not be loyal.

If, however, the class label has been changed, or the probability of the prediction must also be explained (faithfully), it becomes slightly more complicated. Using the example of Henry (see Figure 2) again, the explanation to Henry would be:

- As you satisfy completely the rule "IF Service=L THEN Loyal with 20% probability", we take a full consideration (with a weight of 1) of this rule.
- As you satisfy the following two rules with all but one condition: "IF Service=M AND Sex=M THEN Loyal with 60% probability", and "IF Service=H AND Rate=H THEN Loyal with 80% probability", we take a partial consideration (with a weight of 20%) of these rules.
- As you satisfy the following two rules with all but two conditions: "IF Service=M AND Sex=F THEN Loyal with 40% probability", and "IF Service=H AND Rate=L THEN Loyal with 50% probability", we take a partial consideration (with a weight of 4%) of these rules.
- Combining all of these rules with their corresponding probabilities and weights, the probability of your being loyal is 34.86%.

Table 1. Simulated Datasets

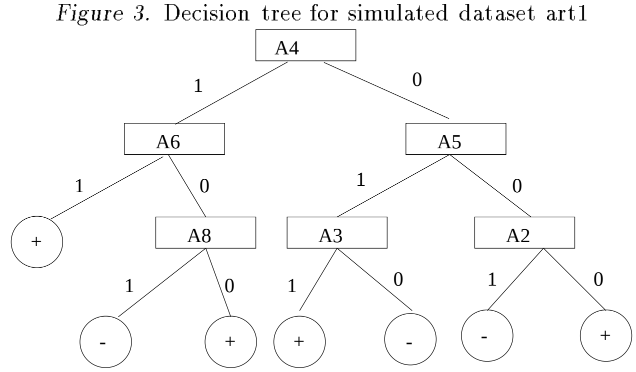
Name	Size	# of Attr.	% Majority examples
art1	500	9	56.5%
art2	500	9	57.1%
art3	500	9	67.5%
art4	500	9	90.2%

We think that if Henry cares about the probability of his verdict, there should be a reasonable chance that he can understand this explanation!

4. Experiments and Results

We use 4 simulated datasets (art1, art2, art3, and art4) and 7 real-world datasets in our experiments. The 4 simulated datasets (see Table 1) are generated from 4 different trees. As an example, Figure 3 shows one of the 4 trees. Each dataset has 500 examples and 9 attributes (not including the class attribute). All the attributes are binary, and attribute values are generated randomly. We intentionally inject some random noise in attribute values and class labels to mimic probability distributions in real-world domains.

Figure 3. Decision tree for simulated dataset art1



We also use 7 real-world datasets from the UCI repository (Blake & Merz, 1998). These datasets (see Table 2) are chosen because their class attributes are binary.

4.1. Comparing Single-Tree Algorithms

As we mentioned before, both C4.5 and C4.4 produce a single decision tree, and examples falling into the same leaf would be assigned the same probability estimation. Our new algorithm, on the other hand, has

Table 2. Descriptions of real-world datasets

Name	# of Attr.	# of classes	# of examples
australia	12	2	690
cars	10	2	683
eco	7	2	700
hepatitis	7	2	332
import	24	2	205
pima	7	2	392
vote	17	2	232

the ability to assign different probabilities to examples in the same leaf. This would produce more different probability values in a tree. We are interested in verifying the following two hypotheses:

1. Does our new algorithm produce more different probabilities compared to C4.5?
2. The goal of our new algorithm is to improve the probability-based ranking measured by AUC in decision trees. We want to know whether our new algorithm yields better AUC compared to C4.5 and C4.4.

We run C4.5, C4.4, and our new algorithm with a 5-fold cross validation on each dataset and repeat 20 times. Thus the results presented in Tables 3 and 4 are averages of these 20 runs. The basic tree building process is the same. In order to investigate the relationship between the number of different probabilities and the tree size, we set up a parameter to control the tree depth during the tree-building process of C4.5 and the new algorithm (C4.4 always builds a full tree without pruning). The confusion factor s of our new algorithm is set at the optimal value 0.3 (see Section 4.3) and the Laplace correction is turned on (see Section 4.4). We then compare the number of probabilities and AUC scores of trees that those two algorithms build at each depth. Only a part of the experimental results are shown in Tables 3 and 4 (we choose 5 different depths for artificial datasets and 3 different depths for real-world datasets). We can draw the following conclusions.

First of all, we are interested in verifying the two hypotheses.

1. From Table 3 and Table 4 we can see that for all 11 datasets, the new algorithm does produce many more different probabilities than C4.5 at different depths. Therefore our strategy does increase the number of different probabilities as a consequence of more accurate probability estimations.

Table 3. Experimental results in simulated datasets (Probs is the number of different probabilities)

Datasets set	Depth	C4.5		New		C4.4
		Probs	AUC	Probs	AUC	AUC
art1	3	15.41	74.08	80.54	76.31	68.55
	4	26.38	74.36	90.49	76.52	
	5	29.25	71.27	91.29	75.90	
	6	23.32	68.81	91.34	74.49	
	7	21.27	68.32	90.95	74.29	
art2	3	15.57	76.17	79.64	77.30	73.59
	4	26.34	76.08	90.37	77.91	
	5	30.03	74.87	90.8	78.58	
	6	25.2	73.95	91.24	78.22	
	7	23.18	72.66	91.17	77.99	
art3	3	15.54	76.22	82.45	76.84	74.59
	4	25.35	76.33	91.23	77.52	
	5	28.85	75.77	92.01	77.46	
	6	24.59	74.72	91.17	77.27	
	7	22.31	74.37	91.49	76.94	
art4	3	9.36	59.06	65.97	67.80	68.6
	4	10.61	63.44	85.28	71.07	
	5	9.89	61.56	87.50	70.06	
	6	8.04	56.7	87.21	69.30	
	7	6.77	55.58	87.42	69.32	

2. Comparing AUC scores, our new algorithm produces a higher AUC than C4.5 and C4.4 in almost all datasets.

We conduct a Wilcoxon test using 95% as the confidence interval. The result shows that the difference of AUC scores between our algorithm and C4.4 (and C4.5) is statistically significant. This presents strong evidence that our new method significantly improves the probability-based ranking of decision trees.

Secondly, from Table 3 and Table 4 we can also find that for both algorithms, the number of different probabilities does not always increase as the tree depth increases. As an example, Figure 4 shows the changes of number of different probabilities and AUC scores in both algorithms when the tree size increases. This figure is based on the data of art4. We can see that for both algorithms, the changes of AUC scores are correlated with changes of the number of different probabilities, rather than the change of the number of leaves (tree size). Furthermore, we can also find that the number of different probabilities starts to decrease when the tree depth further increases. This verifies that turning off pruning is not ideal, because it may produce a large tree with overfitting and repeated probability values (see Section 2).

Thirdly, because of averaging of probability estimations in the decision tree, our new algorithm is less sensitive to the tree size. We compare the AUC score of our new algorithm on the full and pruned trees ($Laplace = on$, $s = 0.3$). Results are shown in Table 5. We can see that although our algorithm produces slightly more different probabilities without pruning,

Table 4. Experimental results in real-world datasets (Probs is the number of different probabilities)

Datasets	Depth	C4.5		New		C4.4
		Probs	AUC	Probs	AUC	AUC
Australia	2	9.87	75.65	52.67	76.58	73.09
	4	17.8	72.84	98.47	77.06	
	7	16.42	66.56	99.68	76.89	
cars	2	15.12	90.49	97.90	91.11	94.29
	4	5.39	87.79	140.00	96.02	
	7	5.30	87.98	140.00	95.91	
eco	2	4.38	98.17	22.97	98.85	99.21
	4	4.62	97.76	29.53	98.86	
	7	4.84	97.97	29.62	98.87	
hepatitis	2	6.77	61.54	7.80	62.35	59.46
	4	6.88	60.98	7.86	63.18	
	7	6.58	60.69	7.86	62.30	
import	2	2.00	100	2.00	100	100
	4	2.00	100	2.00	100	
	7	2.00	100	2.00	100	
pima	2	4.25	74.18	10.83	75.21	75.56
	4	10.97	73.91	25.25	76.53	
	7	11.86	74.68	25.51	76.16	
vote	2	4.19	84.76	10.77	85.60	83.09
	4	4.72	80.60	28.99	86.03	
	7	4.8	73.59	35.33	85.49	

Table 5. Comparison of new algorithm (New) with and without pruning

Datasets	New (with pruning)		New (without pruning)	
	AUC	Probs	AUC	Probs
art1	76.2	90.59	75.4	91.06
art2	78.8	90.28	77.9	90.7
art3	77.2	91.12	76.9	91.55
art4	71.5	81.19	69.35	87.44
australia	77.2	97.37	77.4	100.05
cars	95.8	140	95.7	140
eco	98.1	25.58	98.4	29.92
hepatitis	62.2	7.78	61.4	7.83
import	100	2	100	2
pima	76.6	25.14	76.3	25.29
vote	85.7	30.55	85.1	37.14
Average	81.75	61.96	81.26	63.91

the AUC is slightly lowered than the pruned trees. This is clearly due to the overfitting in the large tree. Therefore, for the rest of the experiments, we apply our algorithm on the pruned tree. Comparing AUC scores by our algorithm to those of C4.4 and C4.5 (Tables 3 and 4), we can see that our algorithm on the pruned trees outperforms C4.5 and C4.4 in almost all datasets.

4.2. Comparing New Algorithm with Bagging

Our new algorithm relies on a single tree to produce more different probability estimations, and it is shown to outperform other single-tree algorithms in C4.4 and C4.5. The outcome of our algorithm is still quite comprehensible (see Section 3.2). As we mentioned in Section 2, bagging has been shown to improve AUC dramatically (Provost & Domingos, 2003), as it averages probabilities from many trees. In this section, we conduct experiments to compare C4.4 with bagging (C4.4-B) with our new algorithm. We use pruned trees, turn on the Laplace correction, and set the confusion factor s to 0.3 in our new algorithm. The results of the comparison are shown in Table 6. We find that bagging is better than our new algorithm in terms of AUC (7 wins, 2 ties, 2 losses). However, bagging has two disadvantages. One is the greater computational cost, and the other is the loss of comprehensibility. Our new algorithm only needs a single tree, and the results are still comprehensible (Section 3.2). There is a tradeoff among the quality of probability estimation, the comprehensibility of the results, and the computational cost when selecting the best algorithm.

Figure 4. Number of different probabilities in different tree depths

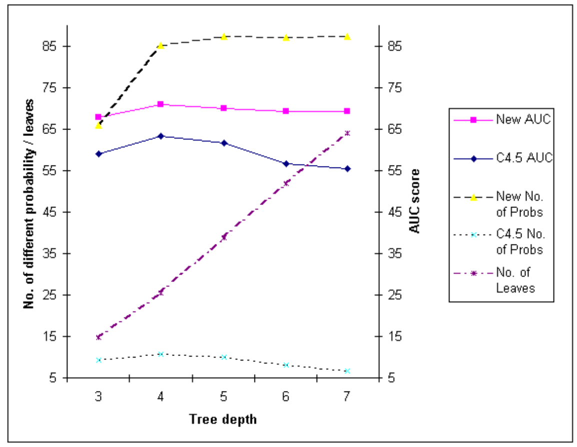


Table 6. AUC scores comparison (new algorithm vs C4.4 with bagging)

	New	C4.4-B
art1	75.33	79.3
art2	79.08	83.2
art3	77.06	82.6
art4	71.2	78.4
australia	76.89	81.9
cars	95.92	97.1
eco	98.51	99.3
hepatitis	62.55	60.3
import	100	100
pima	76.58	74.66
vote	85.48	88.7

4.3. Optimal Confusion Factor

Different values of the confusion factor s may have different influences on probability estimations in our new algorithm. Generally speaking, if a testing example e falls into leaf l, the smaller the s, the less contribution

Table 7. AUC scores on different values of the confusion factor; the bold one is the maximum of each dataset

Dataset	s=0.01	s=0.1	s=0.2	s=0.3	s=0.5	s=0.9
art1	69.43	72.11	74.32	**75.33**	75.40	69.41
art2	73.73	76.64	77.67	**79.08**	78.59	72.64
art3	76.34	76.67	77.02	**77.06**	74.33	67.44
art4	66.7	69.5	70.7	**71.2**	69.5	67.7
australia	73.65	76.19	76.86	**76.89**	76.31	75.54
cars	95.29	95.65	95.88	**95.92**	92.80	79.76
eco	98.81	**98.96**	98.90	98.51	97.54	96.39
hepatitis	61.77	62.27	62.35	**62.55**	61.97	62.04
import	100	100	100	100	100	100
pima	75.47	76.18	**76.87**	76.58	76.52	74.81
vote	84.35	84.76	84.97	85.48	**85.78**	85.11

other leaves will have in the resulting probability of e. If $s = 0$, the probability of e is only decided by the l, as probability estimation in C4.5. With the increase of s, more and more influence of other leaves are applying to the probability of e. If $s = 1$, each leaf has the same weight. In this case, no matter which leaf an example belongs to, the probability of this example is the same: the average of probabilities in all leaves.

We conduct an experiment to compare AUC scores as the confusion factor s changes in the 11 datasets. The goal is to find out the "best" value of s. We use pruned trees, and turn on the Laplace correction. For each dataset, we set s =0.01, 0.1, 0.2, 0.3, 0.5, and 0.9, respectively. For each value of s, we run the new algorithm with 5-fold cross-validation, and obtain the AUC scores.

The results are shown in Table 7. We can see that there is a maximum AUC in each row, which means that there is a "best value" of s for each dataset. The maximum AUC scores are all concentrated around a small area where the value of s is approximately 0.3. This result indicates that the best confusion factor s is insensitive to different datasets. In other words, for most datasets, if the confusion factor is set around 0.3, the AUC score is often optimal. This feature is very useful in applying our algorithm to new datasets.

4.4. Laplace Correction

Provost and Domingos (2003) performed experiments by using 25 datasets from UCI to verify the effect of the Laplace correction. Their results are duplicated in Table 8. We can see that by turning off pruning and using the Laplace correction, C4.4 achieves a total of 2.0% improvement in AUC over C4.5. By using the Laplace correction alone in C4.5, C4.5-L already achieves most of the improvement (1.7%). (The other 0.3% is presumably due to turning off pruning). The rate of improvement of AUC due to the Laplace correction is thus $1.7/2.0 = 85\%$. Therefore, most of the improvement that C4.4 made over C4.5 is due to the use of the Laplace correction (Provost & Domingos, 2003). The question we are interested in is as follows: Is the Laplace correction still that important in our new algorithm?

Table 8. The effect of the Laplace correction in C4.4 (C4.5-L means C4.5 with the Laplace correction). From (Provost & Domingos, 2003).

System	Wins-Ties-Losses	Avg. diff (%)
C4.4 vs. C4.5	18-1-6	2.0
C4.4 vs C4.5-L	13-3-9	0.2
C4.5-L vs C4.5	21-2-2	1.7

Table 9. AUC scores with the Laplace correction on and off.

Datasets	On	Off
art1	76	76.5
art2	78.5	78.4
art3	76.7	77.1
art4	71.3	69.8
australia	76.9	75.9
cars	95.6	94.6
eco	98.2	97.6
hepatitis	62.2	62
import	100	100
pima	76.4	75.8
vote	85.5	85.4
Average	81.57	81.19

We hypothesize that the advantage of the Laplace correction in our new algorithm is less remarkable than that in C4.5, as the improvement of AUC is mainly due to the use of the confusion factor. To verify this, we use these 11 datasets to conduct the experiment. For each dataset, we set the confusion factor $s = 0.3$ and use the pruned trees. Then we calculate the AUC scores by turning on and off the Laplace correction. The results are shown in Table 9. We can see that the Laplace correction improve the AUC score only slightly. The average improvement in AUC with the Laplace correction is $81.57 - 81.19 = 0.38$. We observe and calculate (not shown here) the total improvement in AUC of our algorithm over C4.5 is 2.49. Thus, the rate of AUC improvement due to the Laplace correction is only $0.38/2.49 = 15.3\%$. That is, most of the improvement (about 85%) in AUC of our new algorithm over C4.5 is due to aggregation of probabilities in leaves in decision trees.

5. Conclusions and Future Work

In this paper we present a new tree evaluation algorithm that improves probability-based ranking from a single tree. It has the advantage of easy comprehension of results. Experimental results have shown that our new algorithm does improve AUC over single-tree algorithms (such as C4.4 and C4.5), and that most of the improvement is due to aggregation of probabilities in leaves in decision trees. We also found that our algorithm is robust against the parameters of the algorithm, but it works best on pruned trees, with the Laplace correction, and the confusion factor set to 0.3. Although C4.4 with bagging outperformed our new algorithm in AUC, results from multiple trees with bagging are much harder to understand.

In our future research, we will study how to determine different values of confusion factors on different attributes. In this paper, we set the confusion factor for each attribute to be the same. In real-world applications, the values of confusion factors could be different. Attributes with a higher error rate in data could be assigned a higher confusion factor. This will give users more flexibility in incorporating domain specific knowledge in decision trees to obtain optimal results in prediction.

Acknowledgements

We gratefully thank Foster Provost for kindly providing us with the source codes of C4.4, which is a great help to us in the comparison of C4.5 and C4.4 to our algorithm. We also thank Dong Han and Jingjing Lu in helping us at various stages of experiments. The Area Chair and reviewers of this conference provided excellent suggestions in improving the paper.

References

Bauer, E., & Kohavi, R. (1999). An empirical comparison of voting classification algorithms: Bagging, boosting and variants. *Artificial Intelligence*, *36*, 105–142.

Blake, C., & Merz, C. (1998). UCI repository of machine learning databases. http://www.ics.uci.edu/~mlearn/MLRepository.html. University of California, Irvine, Dept. of Information and Computer Sciences.

Bradley, A. P. (1997). The use of the area under the ROC curve in the evaluation of machine learning algorithms. *Pattern Recognition*, *30*, 1145–1159.

Cohen, W. W., Schapire, R. E., & Singer, Y. (1999). Learning to order things. *Journal of Artificial Intelligence Research*, *10*, 243–270.

Hand, D. J., & Till, R. J. (2001). A simple generalisation of the area under the ROC curve for multiple class classification problems. *Machine Learning*, *45*, 171–186.

Hastie, T. J., & Pregibon, D. (1990). *Shrinking trees* (Technical Report). At&T laboratories.

Ling, C. X., Huang, J., & Zhang, H. (2003). AUC: a statistically consistent and more discriminating measure than accuracy. *Proceedings of 18th International Conference on Artificial Intelligence (IJCAI-2003)*. To appear.

Margineantu, D. D., & Dietterich, T. G. (2001). Improved class probability estimates from decision tree models. In C. Holmes (Ed.), *Nonlinear estimation and classification*. The Mathematical Sciences Research Institute, University of California Berkeley.

Pazzani, M., Merz, C., Murphy, P., Ali, K., Hume, T., & Brunk, C. (1994). Reducing misclassification costs. In *Proceedings of the 11th international conference on machine learning*, 217–225. Morgan Kaufmann.

Provost, F., & Domingos, P. (2003). Tree induction for probability-based ranking. *Machine Learning*. To appear.

Provost, F., Fawcett, T., & Kohavi, R. (1998). The case against accuracy estimation for comparing induction algorithms. In *Proceedings of the fifteenth international conference on machine learning*, 445–453. Morgan Kaufmann.

Quinlan, J. (1993). *C4.5: Programs for machine learning*. Morgan Kaufmann: San Mateo, CA.

Shavlik, J., Mooney, R., & Towell, G. (1991). Symbolic and neural learning algorithms: An experimental comparison. *Machine Learning*, *6*, 111 – 144.

Smyth, P., Gray, A., & Fayyad, U. (1995). Retrofitting decision tree classifiers using kernel density estimation. *Proceedings of the 12th International Conference on machine Learning* (pp. 506–514).