Conditional Independence Trees

Harry Zhang and Jiang Su

Faculty of Computer Science, University of New Brunswick
P.O. Box 4400, Fredericton, NB, Canada E3B 5A3
hzhang@unb.ca,
WWW home page: http://www.cs.unb.ca/profs/hzhang/

Abstract. It has been observed that traditional decision trees produce poor probability estimates. In many applications, however, a probability estimation tree (PET) with accurate probability estimates is desirable. Some researchers ascribe the poor probability estimates of decision trees to the decision tree learning algorithms. To our observation, however, the representation also plays an important role. Indeed, the representation of decision trees is fully expressive theoretically, but it is often impractical to learn such a representation with accurate probability estimates from limited training data. In this paper, we extend decision trees to represent a joint distribution and conditional independence, called conditional independence trees (CITrees), which is a more suitable model for PETs. We propose a novel algorithm for learning CITrees, and our experiments show that the CITree algorithm outperforms C4.5 and naive Bayes significantly in classification accuracy.

1 Introduction

Classification is a fundamental issue of machine learning, in which a classifier is induced from a set of labeled training examples represented by a vector of attribute values and a class label. We denote a vector of attributes by an bold-face upper-case letter $\mathbf{A}$, $\mathbf{A} = (A_1, A_2, \cdots, A_n)$, and an assignment of value to each attribute in $\mathbf{A}$ by a corresponding bold-face lower-case letter $\mathbf{a}$. We use C to denote the class variable and c to denote its value. Thus, a training example $E = (\mathbf{a}, c)$, where $\mathbf{a} = (a_1, a_2, \cdots, a_n)$, and a_i is the value of attribute A_i. A classifier is a function that maps an example to a class label.

There are numerous inductive learning algorithms, such as decision trees, Bayesian networks, and neural networks, that can be categorized into two major approaches: probability-based approach and decision boundary-based approach. In a probability-based learning algorithm, a probability distribution $p(\mathbf{A}, C)$ is learned from the training data, and an example E is classified into the class c with the maximum posterior class probability $p(c|E)$ (or simply class probability), as shown below.

$$C_{pb}(E) = \arg\max_{c} p(c|E). \qquad (1)$$

Various probability-based learning algorithms have been developed, which are different in the way of estimating $p(c|E)$. For example, a naive Bayes classifier (or simply naive Bayes), shown in Equation 2, is a successful one widely used in many applications.

$$C_{nb}(E) = \arg\max_{c} p(c) \prod_{i=1}^{n} p(a_i|c). \tag{2}$$

A naive Bayes is based on the crucial assumption that all the attributes are independent given the value of the class variable, called conditional independence assumption and shown in Equation 3. Obviously, this assumption is rarely true in reality.

$$p(\mathbf{a}|c) = \prod_{i=1}^{n} p(a_i|c). \tag{3}$$

In a decision boundary-based algorithm, an explicit decision boundary is extracted from the training data, and an example E is classified into class c if E falls into the decision area corresponding to c. Decision tree algorithms are well-known as decision boundary-based. While decision trees perform quite well in classification, it is also found that their probability estimates are poor [9]. Building decision trees with accurate probability estimates, called probability estimation trees (PETs), has received a great deal of attention recently [10]. Some researchers ascribe the poor probability estimates of decision trees to the decision tree learning algorithms. Thus, many techniques have been proposed to improve the learning algorithms in producing accurate probability estimates[10].

To our observation, however, the representation also plays an important role. Indeed, the representation of decision trees is fully expressive theoretically, but it is often impractical to learn such a representation with accurate probability estimates from limited training data.

In a decision tree, the class probability $p(c|E)$ is estimated by the fraction of the examples of class c in the leaf into which E falls. Thus, the class probabilities of all the examples in the same leaf are equal. This is an obstacle in building an accurate PET, because two contradictory factors are in play at the same time. On one hand, traditional decision tree algorithms, such as C4.5, prefer a small tree. Thus, a leaf has more examples and the class probability estimates are more reliable. A small tree, however, has a small number of leaves, thus more examples will have the same class probability. That prevents the learning algorithm from building an accurate PET. On the other hand, if the tree is large, not only may the tree overfit the training data, but the number of examples in each leaf is also small, and thus the probability estimates would not be accurate and reliable. Such a contradiction does exist in traditional decision trees.

Our motivation is to extend the representation of traditional decision trees not only to represent accurate probabilities but also to be easily learnable from limited data in practice. Naturally, if an accurate PET is built, its classification accuracy should also be high, since an accurate approximation of $p(c|E)$ is found and can be used for classification. Thus, we use classification accuracy to evaluate learning algorithms in this paper.

The rest of the paper is organized as follows. Section 2 introduces the related work on learning decision trees with accurate probability estimates. Section 3 presents a novel model for PETs and a corresponding algorithm for learning PETs. In Section 4, we present empirical experiments. The paper concludes with discussion and some directions for future work.

2 Related Work

Since traditional decision tree algorithms, such as C4.5, have been observed to produce poor probability estimates of probabilities [9], a substantial amount of work has been done recently on accurate PETs [10]. Provost and Domingos [10] point out that the reason behind the poor estimates of decision trees is not the decision tree representation, but the inductive algorithm. They propose a few techniques to modify the C4.5 learning algorithm.

First, they turn off the pruning and collapsing in C4.5, since they notice that a larger tree tends to have more accurate probability estimates.

Second, they propose to use Laplace correction to smooth probability estimates. The reason is the fragmentation problem: As the splitting process proceeds, the data associated with each descendant node becomes small. Eventually, when the depth of the tree is large, there is very little data with each leaf node [6]. Thus, the probability estimates based on frequency are not accurate. This issue is more serious after turning off the pruning and collapsing mechanism.

The resulting algorithm is called C4.4. They also find out that bagging, an ensemble method, improves the probability estimates of decision trees significantly.

Ling and Yan also propose a method to improve the probability estimates of decision trees [7]. They present a method to generate the class probability of an example, in which an average of the probability estimates from all leaves of the tree is used, instead of only using the leaf into which it falls. Thus, each leaf contributes to the class probability estimate of an example in different degree.

In learning a decision tree, a critical step is to choose the “best” attribute in each step. The entropy-based splitting criteria, such as information gain and gain ratio, have been widely used. There are also other splitting criteria proposed. One is Bayesian approach [3], which searches for a decision tree with the maximum posterior probability given the training examples.

Although decision trees are well-known as a nonparametric and decision-boundary based classifier, each leaf of a tree actually represents a conditional probability distribution. These types of decision trees are called probabilistic decision trees. Jordan [5] analyzes decision trees within a probabilistic framework. A decision tree actually represents a sequence of probabilistic decisions, each conditional on the attribute values and previous decisions. Thus, Bayesian theory can be used in analyzing the performance of the tree. A learning algorithm based on EM (Expectation-Maximization) has been proposed for maximum likelihood parameter estimation in a hidden Markov decision tree.

A questionable point of traditional decision trees (including probabilistic trees) is that only the attributes along the path from the root to a leaf are used in both classification and probability estimation. Since a small tree is preferred by traditional decision tree learning algorithms, many attributes may not be used. This is a more serious issue in learning PETs than classification. Kohavi proposes to deploy a naive Bayes in each leaf, and the resulting decision tree is called an NBTree [6]. The algorithm for learning an NBTree is similar to C4.5. After a tree is grown, a naive Bayes is constructed for each leaf using the data associated with that leaf. An NBTree classifies an example by sorting it to a leaf and applying the naive Bayes in that leaf to assign a class label to it. Actually, deploying a model at leaves to calibrate the probability estimates of a decision tree has been proposed by Symth, Gray and Fayyad [11]. They also notice that every example from a particular leaf has the same probability estimate, and thus suggest to place a kernel-based probability density estimator at each leaf.

Our work is inspired by the works of Kohavi, and Symth, Gray and Fayyad, but from different point of view. Indeed, if a local model that incorporates the attributes not occurring on the path is deployed at each leaf, together with the conditional probability of the attributes occurring on the path, the resulting tree represents accurate probabilities. If the structure of standard decision trees is learned and used the same way as in C4.5, however, the leaf models would not directly and explicitly benefit from the structure, and thus would still play a role of smoothing. Our motivation is how to learn and use the structure of a tree to explore conditional independences among attributes, such that a simple leaf model, like a naive Bayes, gives accurate probability estimates. Then, the resulting model is more compact and more easily learnable, while its representation is still accurate.

3 Understanding Decision Trees from Probabilistic Perspective

Even though there theoretically exists a decision tree with accurate probability estimates for any given problem, such a tree tends to be large and learnable only when sufficient (huge) training data are available. In practice, a small tree is preferred. Thus, poor probability estimates are yielded. Therefore, the representation of a decision tree should be extended to represent accurate probabilities and be learnable from limited training data.

3.1 Probabilistic Decision Trees

Figure 1 shows an example of a probabilistic tree, in which each leaf L represents a conditional distribution $p(C|\mathbf{A_p}(L))$, where $\mathbf{A_p}(L)$ are the attributes that occur in the path from the root to L. For simplicity, the attributes that occur in the path is called the path attributes of L, and all other attributes are called the leaf attributes of L, denoted by $\mathbf{A_l}(L)$.

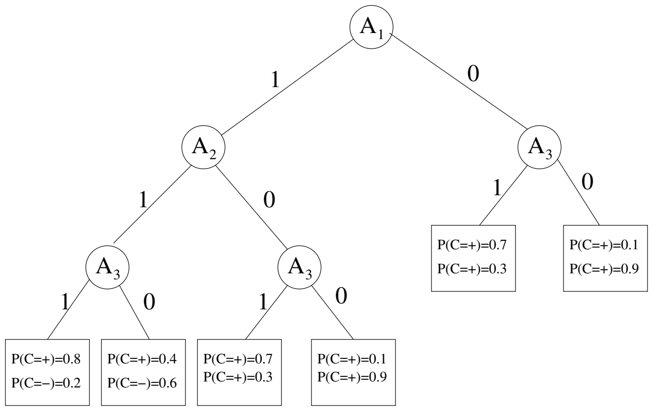

Fig. 1. An example of an probabilistic tree

In practice, $p(C|\mathbf{A_p}(L))$ is often estimated by using the fraction of examples of class C in L, and the classification of a decision tree is based on $p(C|\mathbf{A_p}(L))$. Thus, from the probabilistic point of view, a decision tree can be also viewed as a probability-based classifier, defined as below.

$$C_{dt}(E) = \arg\max_{c} p(c|\mathbf{a_p}(L)), \tag{4}$$

where L is the leaf into which E falls, $\mathbf{a_p}(L)$ is the value of the path attributes of L, and $C_{dt}(E)$ is the classification given by the decision tree.

Comparing Equation 4 with Equation 1, $p(c|\mathbf{a_p}(L))$ is actually used as an approximation of $p(c|E)$ in a decision tree. Thus, all the examples falling into the same leaf have the same class probability. Due to the fact that traditional decision tree learning algorithms prefer a small tree, a leaf tends to have more examples with the same probability. Therefore, decision trees are prone to be poor PETs.

3.2 Conditional Independence Trees

In a probabilistic tree, a leaf L represents the conditional probability distribution $p(C|\mathbf{A_p}(L))$. If there is a representation of the conditional probability distribution over the leaf attributes at each leaf, called the local conditional distribution and denoted by $p(\mathbf{A_l}(L)|\mathbf{A_p}(L), C)$, then each leaf represents a full joint distribution over all the attributes, as shown in the equation below.

$$p(\mathbf{A}, C) = \alpha p(C|\mathbf{A_p}(L))p(\mathbf{A_l}(L)|\mathbf{A_p}(L), C), \tag{5}$$

where α is a normalization factor.

Definition 1. *A probabilistic decision tree T is called a joint probabilistic tree, if each of its leaves represents both the conditional probability distribution $p(C|\mathbf{A_p}(L))$ and $p(\mathbf{A_l}(L)|\mathbf{A_p}(L), C)$.*

Definition 2. *A joint probability tree T is called a conditional independence tree, or simply CITree, if the local conditional independence assumption, shown in Equation 6, is true for each leaf L.*

$$p(\mathbf{A_l}(L)|\mathbf{A_p}(L), C) = \prod_{i=1}^{m} p(A_{li}|C, \mathbf{A_p}(L)), \tag{6}$$

where $\mathbf{A_l} = (A_{l1}, A_{l2}, \cdots, A_{lm})$ are the leaf attributes of L.

According to Definition 2, the structure of a CITree represents the conditional independences among attributes, and its leaves represent a joint distribution. A CITree is different from a probabilistic tree in the following aspects.

1. A CITree represents a joint distribution over all the attributes, but a probabilistic tree represents only the conditional probability distribution of the path attributes.
2. A CITree explicitly defines conditional dependences among attributes.

Comparing Equation 6 with Equation 3, we notice that the local conditional independence assumption of CITrees is a relaxation of the (global) conditional independence assumption of the naive Bayes. Thus, the local conditional independence assumption is more realistic in applications. In addition, the local conditional independence represented in a CITree is also different from the conditional independence in a Bayesian network. In a Bayesian network, An attribute A_1 is conditionally independent of attribute A_2 given A_3 means that for all the values of A_3, A_1 is independent of A_2. In a CITree, however, the conditional independence is that A_1 is independent of A_2, given a specified value of A_3. The granularity in a CITree is finer than that in a Bayesian network.

It is interesting to notice that, after growing a CITree, if a naive Bayes is deployed on each leaf using only the data associated with it, the naive Bayes, called leaf naive Bayes, represents the actual joint distribution. A leaf naive Bayes in leaf L is shown below.

$$C_{lnb}(E) = \arg\max_{c} p_L(c) \prod_{i=1}^{m} p_L(a_{li}|c), \tag{7}$$

where $p_L(c)$ denotes the probability of examples in L being in c, and $p_L(a_{li}|c)$ is the probability that the examples of class c have $A_{li} = a_{li}$ in L. It is obvious that $p_L(c) = p(c|\mathbf{a_p}(L))$ and $p_L(a_{li}|c) = p(a_{li}|c, \mathbf{a_p}(L))$. So $p_L(c)\prod_{i=1}^{m} p_L(a_{li}|c)$ is proportional to $p(c|E)$. Thus, if the structure of the CITree is found, the naive Bayes is a perfect model for leaves.

Generally, a CITree can be viewed as a combination of a decision tree and a naive Bayes. It is well-known that decision trees are fully expressive with the class of propositional language; that is, any Boolean function is represented by a decision tree. However, a naive Bayes has limited expressive power; that is, it can only represent linear Boolean functions [4]. Interestingly, any joint distribution can be represented by a CITree. According to the product rule,

$$p(A_1, A_2, \cdots, A_n, C) = p(C)p(A_1|C)P(A_2|A_1, C) \cdots P(A_n|A_1, \cdots, A_{n-1}, C). \tag{8}$$

It is trivial to build a CITree to represent $p(A_1, A_2, \cdots, A_n, C)$. Thus, CITrees are also fully expressive.

The representation of CITrees, however, is more compact than that of decision trees. To show this, let us consider only full dependences among attributes. An attribute A_i is said to fully depend on A_j, if $A_i = A_j$. Notice that if an attribute is conditionally independent of all other attributes, it does not occur on any path. If several attributes conditionally depend on one attribute, only that attribute occurs in the path. In the extreme case that the global conditional independent assumption is true, a CITree has only one node, which is just a global naive Bayes. Assume that there are n attributes. The maximum height of a CITree is $\frac{n}{2}$, which corresponds to that each attributes depends exactly on another attribute. The maximum height of a decision tree is n. Our experiments in Section 4 show that the average size of CITrees is much smaller than that of decision trees.

3.3 A Novel Algorithm for Learning CITree

From the discussion in the preceding section, a CITree can represent any joint distribution. Thus, a CITree is a perfect PET, and the classification based on CITree is accurate. But in practice, learning the structure of a CITree is just as time-consuming as learning an optimal decision tree. However, a good approximation of a CITree, which gives good estimates of class probabilities, is satisfiable in many applications. If the structure of a CITree is determined, a leaf naive Bayes is a perfect model representing the local conditional distributions at leaves.

Building a CITree could be also a greedy and recursive process, similar to building a decision tree. At each step, choose the "best" attribute as the root of the (sub)tree, split the associated data into disjoint subsets corresponding to the values of the attribute, and then recur this process for each subset until certain criteria are satisfied.

Notice as well, however, the difference between learning a CITree and learning a decision tree. In building a decision tree, we are looking for a sequence of attributes that leads to the least impurity in all leaves of the tree. The key in choosing an attribute is whether the resulting partition of the examples is "pure" or not. It is natural, since the most common class of a leaf is used as the class of all the examples in that leaf. However, such a selection strategy does not necessarily lead to the truth of the local conditional independence assumption. In building a CITree, we intend to choose the attributes that make the local conditional independence among the rest of attributes true as much as possible. That means that, even though the impurity of its leaves is high, it could still be a good CITree, as long as the leaf attributes are independent. Thus, traditional decision tree learning algorithms are not directly suitable for learning CITrees.

In learning a CITree, an attribute, given which all other attributes have the maximum conditional independence, should be selected at each step. Thus, we should select the attribute with the greatest influence on other attributes. Our idea is to try each possible attribute as the root, evaluate the resulting tree, and choose the attribute that achieves the highest classification accuracy.

Similar to C4.5, our learning algorithm has two separate steps: growing a tree and pruning. In growing a tree, each possible attribute is evaluated at each step, and the attribute that gives the most improvement in accuracy is selected. The algorithm is depicted below.

```
Algorithm CITree ($\mathbf{T}$, $\mathbf{S}$, $\mathbf{A}$)
Input : CITree $\mathbf{T}$, a set $\mathbf{S}$ of labeled examples, a set of attributes $\mathbf{A}$
Output : a CITree.
    1. Evaluate the current CITree $\mathbf{T}$.
    2. For all attributes $A$ in $\mathbf{A}$
        – Partition $\mathbf{S}$ into $\mathbf{S_1}, \cdots, \mathbf{S_k}$, each of which corresponds to a value of $A$.
        – Create a leaf naive Bayes for each $\mathbf{S_i}$.
        – Evaluate the resulting CITree.
    3. Choose the attribute $A_{opt}$ with the highest accuracy.
    4. For all values $a$ of $A_{opt}$
       CITree($\mathbf{T}_a$, $\mathbf{S}_a$, $\mathbf{A} - \{A_{opt}\}$).
       Add $\mathbf{T}_a$ as a child of $\mathbf{T}$.
    5. Return $\mathbf{T}$.
```

Note that we train a leaf naive Bayes by using the examples in this leaf, and the accuracy is the accuracy of classifying those examples using the leaf naive Bayes.

In the algorithm described above, we grow a tree as large as possible until we are out of data or attributes, and then start a pruning process with two steps:

1. Conduct the pessimistic error-based post-pruning in C4.5.
2. Apply pruning based on the accuracy of leaf naive Bayes, in which the children of a node are removed only if the resulting pruned tree (making it a leaf node and deploying a naive Bayes at it) performs no worse than the original tree.

4 Experiments

We conduct experiments to compare our algorithm CITree with C4.5 and naive Bayes. Our algorithm is implemented within the Weka framework [12]. We use the implementation of naive Bayes and C4.5(J48) in Weka. We have chosen 33 datasets from the UCI repository [8], described in Table 1. In our experiment, the average accuracy on each dataset has been obtained using 3-fold cross validation 10 times. Numeric attributes are discretized using ten-bin discretization implemented in Weka[12]. Missing values are also processed using the mechanism in Weka.

Table 1. Description of the datasets used in the experiments.

dataset	Size	Number of Attribute	missing value	Class
Letter	20000	17	N	26
Mushroom	8124	22	Y	2
Waveform	5000	41	N	3
Sick	3772	30	Y	2
Hypothyroid	3772	30	Y	4
Chess End-Game	3196	36	N	2
Splice	3190	62	N	3
Segment	2310	20	N	7
German Credit	1000	24	N	2
Vowel	990	14	N	11
Anneal	898	39	Y	6
Vehicle	846	19	N	4
Pima Indians Diabetes	768	8	N	2
Wisconsin-breast-cancer	699	9	Y	2
Credit Approval	690	15	Y	2
Soybean	683	36	Y	19
Balance-scale	625	5	N	3
Vote	435	16	Y	2
Horse Colic	368	28	Y	2
Ionosphere	351	34	N	2
Primary-tumor	339	18	Y	22
Heart-c	303	14	Y	5
Breast cancer	286	9	Y	2
Heart-statlog	270	13	N	2
Audiology	226	70	Y	24
Glass	214	10	N	7
Sonar	208	61	N	2
Autos	205	26	Y	7
Hepatitis Domain	155	19	Y	2
Iris	150	5	N	3
Lymph	148	19	N	4
Zoo	101	18	N	7
Labor	57	16	N	2

Table 2 shows the average accuracy obtained by the three algorithms. The comparison of the three algorithms on these datasets, in which a paired t-test with a confidence of 95% has been used, are summarized in Table 3. Our observations are summarized below.

1. The CITree algorithm outperforms the naive Bayes significantly: It wins in 7 datasets, ties in 26 datasets and loses in 0 dataset. The average accuracy for CITree is 83.26%, higher than the average accuracy 81.83% of naive Bayes. That fact is understandable, since the conditional independences among attributes have been explored and represented in CITrees. Thus, the class probability estimates of a CITree are expected to be more accurate than those of naive Bayes.
2. The CITree algorithm also outperforms C4.5 significantly: It wins in 7 datasets, ties in 25 datasets and loses in 1 datasets. The average accuracy for decision trees is 80.69%, lower than CITree's. The CITree algorithm builds a tree from a viewpoint different from C4.5's. Since C4.5's good performance in classification is well-known, this comparison provides evidence to support CITree's.
3. The sizes of CITrees are significantly smaller than the sizes of decision trees over all the datasets. Here the size of a tree is the number of nodes. The average tree size for CITrees is 11, and for C4.5 it is 391. This verifies that a CITree is much more compact than a decision tree. However, the efficiency of the CITree algorithm is lower than C4.5. Roughly speaking, the average training time of the CITree algorithm is 10 time slower than C4.5.

5 Conclusions

In this paper, we propose a model CITree for accurate probability representation, the structure of which explicitly represents conditional independences among attributes. We show that CITrees are more expressive than naive Bayes and more compact than decision trees. A CITree can be implemented by using naive Bayes at leaves. We present a novel algorithm which builds a tree by exploring the conditional independence among attributes, different from traditional decision tree learning algorithms. Our experiments show that CITrees outperform C4.5 and naive Bayes significantly in classification accuracy. The results provide evidence that a CITree yields more accurate probability estimates.

Our goal of this research is to build accurate PETs. Although accuracy to some degree reflects the quality of probability estimates, it is interesting to know directly the errors of the probability estimates by using artificial data. In our future research, we will also investigate other performance measures that more precisely reflect the errors between the true probability and the estimated probability, such as the area under the ROC curve [2].

References

1. Bennett, P. N.: Assessing the calibration of Naive Bayes' posterior estimates. Technical Report No. CMU-CS00-155 (2000)

Table 2. Experimental results on accuracy. In this table, the dataset are sorted in a decreasing order of their size.

Dataset	CITree	NB	C4.5	CITreeSize	Treesize(c4.5)
Letter	81.17 ± 0.35	69.89 ± 0.65	78.27 ± 0.57	133	8737
Mushroom	99.93 ± 0.12	95.59 ± 0.69	100 ± 0	9	30
Waveform	80.12 ± 1.75	79.56 ± 0.68	71.96 ± 1.36	23	1153
Sick	96.9 ± 0.46	96.9 ± 0.46	97.86 ± 0.42	8	45
Hypothyroid	93.06 ± 0.41	93.06 ± 0.41	93.17 ± 0.23	1	24
Chess End-Game	96.86 ± 0.87	87.95 ± 0.85	99.17 ± 0.25	26	55
Splice	92.5 ± 1.64	95.15 ± 0.73	93.46 ± 0.88	14	214
Segment	91.06 ± 0.79	88.44 ± 0.83	91.48 ± 1.36	12	386
German Credit	75.03 ± 1.25	75.03 ± 1.25	71.68 ± 1.02	8	134
Vowel	79.74 ± 8.16	63.45 ± 3.39	67.61 ± 2.43	22	649
Anneal	94.9 ± 2.28	94.08 ± 1.23	98.72 ± 0.66	5	75
Vehicle	67.36 ± 2.83	60.57 ± 2.3	68.34 ± 2.73	15	412
Pima Indians Diabetes	74.31 ± 3.16	75.35 ± 2.49	74.89 ± 2.03	5	56
Soybean	91.19 ± 1.44	91.36 ± 1.47	88.61 ± 1.26	8	81
Wisconsin-breast-cancer	96.89 ± 0.91	97.14 ± 0.68	94.33 ± 1.16	8	46
Credit Approval	85.03 ± 1.33	84.77 ± 1.3	85.62 ± 1.35	6	42
Balance-scale	89.39 ± 1.2	89.39 ± 1.2	66.54 ± 2.9	1	85
Vote	93.98 ± 1.64	89.04 ± 1.97	95.06 ± 1.28	6	10
Horse Colic	80.6 ± 2.75	79.71 ± 3.57	83.88 ± 1.6	8	15
Primary-tumor	42.09 ± 4.35	48.31 ± 2.36	39.57 ± 1.77	6	83
Ionosphere	89.01 ± 3.39	90.35 ± 2.24	88.34 ± 2.93	9	48
Heart-c	84.27 ± 3.71	84.37 ± 2.73	78.06 ± 2.89	1	49
Breast cancer	72.73 ± 2.86	72.73 ± 2.86	71.92 ± 3.4	2	7
Heart-statlog	82.88 ± 3.61	83.31 ± 2.7	77.64 ± 5.04	1	87
Audiology	70.95 ± 3.99	71.21 ± 4.08	77.03 ± 2.89	1	49
Glass	58.14 ± 4.59	58.55 ± 4.26	58.56 ± 3.95	4	78
Sonar	76.13 ± 2.63	76.69 ± 2.79	66.79 ± 6.25	7	47
Autos	64.6 ± 10.4	61.15 ± 6.3	66.15 ± 6.16	9	145
Hepatitis Domain	81.7 ± 3.81	81.7 ± 3.81	81.71 ± 3.9	1	4
Iris	95.29 ± 3.6	95.29 ± 3.6	95.49 ± 2.45	1	11
Lymph	82.35 ± 3.02	82.92 ± 2.48	77.73 ± 6.57	2	31
Zoo	96.14 ± 2.87	96.14 ± 2.87	92.33 ± 3.52	1	15
Labor	91.25 ± 3.38	91.25 ± 3.38	70.8 ± 6.6	1	6
Average	83.26	81.83	80.69	11	391

Table 3. Summary of the experimental results. An entry *w-t-l* means that the algorithm at the corresponding row wins in *w* datasets, ties in *t* datasets, and loses in *l* datasets, compared to the algorithm at the corresponding column.

	C4.5	NB
CITree	7-25-1	7-26-0
C4.5		7-17-9

2. 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
3. Buntine, W.: Learning Classification Trees. Statistics and Computing **2** (1992) 63-73
4. Domingos, P., Pazzani M.: Beyond Independence: Conditions for the Optimality of the Simple Bayesian Classifier. Machine Learning **29** (1997) 103-130
5. Jordan, M. I., A Statistical Approach to Decision Tree Modeling. Proceedings of the Eleventh International Conference on Machine Learning, Morgan Kaufmann (1994) 363-370
6. Kohavi, R.: Scaling Up the Accuracy of Naive-Bayes Classifiers: A Decision-Tree Hybrid. Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96). AAAI Press (1996) 202-207
7. Ling, C. X., Yan, R. J.: Decision Tree with Better Ranking. Proceedings of the 20th International Conference on Machine Learning. Morgan Kaufmann (2003) 480-487
8. Merz, C., Murphy, P., Aha, D.: UCI repository of machine learning databases. Dept of ICS, University of California, Irvine (1997). http://www.ics.uci.edu/m̃learn/MLRepository.html
9. 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 (1998) 445-453
10. Provost, F. J., Domingos, P.: Tree Induction for Probability-Based Ranking. Machine Learning **52(3)** (2003) 199-215
11. Symth, P., Gray, A., Fayyad, U.: Retrofitting decision tree classifiers using kernel density estimation. Proceedings of the Twelfth International Conference on Machine Learning. Morgan Kaufmann (1996) 506–514
12. Witten, I. H., Frank, E.: Data Mining –Practical Machine Learning Tools and Techniques with Java Implementation. Morgan Kaufmann (2000)