

Learning Naïve Bayes Tree for Conditional Probability Estimation

Han Liang^{1*}, Yuhong Yan²

¹Faculty of Computer Science, University of New Brunswick
Fredericton, NB, Canada E3B 5A3

²National Research Council of Canada
Fredericton, NB, Canada E3B 5X9
yuhong.yan@nrc.gc.ca

Abstract. Naïve Bayes Tree uses decision tree as the general structure and deploys naïve Bayesian classifiers at leaves. The intuition behind it is that naïve Bayesian classifiers work better than decision trees when the sample data set is small. Therefore, after several attribute splits when constructing a decision tree, it is better to use naïve Bayesian classifiers at the leaves than to continue splitting the attributes. Naïve Bayes Tree is used to improve classification accuracy and Area Under the Curve (AUC). In this paper, we propose a learning algorithm to improve the conditional probability estimation in the diagram of Naïve Bayes Tree. The motivation for this work is that, for cost-sensitive learning where costs are associated with conditional probabilities, the score function is optimized when the estimates of conditional probabilities are accurate. The learning algorithm is a greedy and recursive process where in each step, the conditional log likelihood (CLL) is used as the metric to expand the decision tree. When some bound conditions are met, the algorithm uses naïve Bayes to estimate the probabilities for leaf attributes given the class variable and the path attributes. The whole tree encodes the conditional probability in its structure and parameters. The additional benefit of accurate estimates of conditional probability is that both the classification accuracy and AUC could be improved. On a large suite of benchmark sample sets, our experiments show that the CLL tree (CLL-Tree) outperforms the state-of-art learning algorithms, such as Naïve Bayes Tree and naïve Bayes significantly in yielding accurate conditional probability estimation and improving classification accuracy and AUC.

1 Introduction

Classification is a fundamental issue of machine learning in which a classifier is induced from a set of labeled training samples represented by a vector of attribute values and a class label. We denote attribute set $\mathbf{A} = \{A_1, A_2, \dots, A_n\}$, 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.

* This work was done when the author was a visiting worker at Institute for Information Technology, National Research Council of Canada.

Thus, a training sample $E = (\mathbf{a}, c)$, where $\mathbf{a} = (a_1, a_2, \dots, a_n)$, and a_i is the value of attribute A_i . A classifier is a function f that maps a sample E to a class label c , i.e. $f(\mathbf{a}) = c$. The inductive learning algorithm returns a function h that approximates f . The function h is called a hypothesis. The classifier can predict the assignment of C for an unlabeled testing sample $E_t = (\mathbf{b})$, i.e. $h(\mathbf{b}) = c_t$.

Various inductive learning algorithms, such as decision trees, Bayesian networks, and neural networks, can be categorized into two major approaches: probability-based approach and decision boundary-based approach. In a generative probability learning algorithm, a probability distribution $p(A, C)$ is learned from the training samples as a hypothesis. Then we can theoretically compute the probability of any E in the probability space. A testing sample $E_t = (\mathbf{b})$ is classified into the class c with the maximum posterior class probability $p(c|\mathbf{b})$ (or simply class probability), as shown below.

$$h(\mathbf{b}) = \arg \max_{c \in C} p(c|\mathbf{b}) = \arg \max_{c \in C} p(c, \mathbf{b})/p(\mathbf{b}) \quad (1)$$

Decision tree learning algorithms are well known as decision boundary-based. Though their probability estimates are poor, the algorithms can make good decisions on which side of the boundary a sample data falls. Decision trees work better when the sample data set is large. It is because, after several splits of attributes, the number of samples at the subspaces is too few on which to base the decision, while naïve Bayesian classifier works better in this case. Therefore, instead of continuing to split the attributes, naïve Bayesian classifiers are deployed at the leaves. [5] proposed this hybrid model called Naïve Bayes Tree (NBTree). It is reported that NBTree outperforms C4.5 and naïve Bayes in classification accuracy and AUC.

In this paper, we try to benefit from both decision tree and probability models in that we propose to use NBTree to improve the conditional probability estimation given the support attributes, i.e. $p(C|\mathbf{A})$. Accurate conditional probability is important in many aspects. First, in cost-sensitive classification, knowing the accurate conditional probability is crucial in making a decision. Determining only the decision boundary is not enough. Second, improving conditional probability can possibly improve classification accuracy, though it is not a necessary condition. Third, improving conditional probability can improve AUC which is a metric used for ranking.

Our proposed learning algorithm is a greedy and recursive procedure similar to NBTree. In each step of expanding the decision tree, the Conditional Log Likelihood (CLL) is used as the score function to select the best attribute to split, where let the CLL of a classifier B , given a (sub) sample set S be

$$CLL(\mathbf{B}|\mathbf{S}) = \sum_{s=1}^n \log P_B(C|\mathbf{A}) \quad (2)$$

The splitting process ends when some conditions are met. Then for the samples at leaves, naïve Bayesian classifiers are generated. This kind of NBTrees optimizes the estimation of conditional probability. We call the generated tree

CLL Tree (CLLTree). We present that on a large suite of benchmark sample sets, our empirical results show that CLLTree significantly outperforms the state-of-art learning algorithms, such as NBTree and naïve Bayes in yielding accurate probability estimation, classification accuracy and AUC.

This paper is organized as follows: section 2 reviews the existing work of using decision tree to estimate probability; section 3 presents the principles of CLLTree; section 4 presents the learning algorithm; section 5 collects the experimental results and section 6 is the conclusion.

2 Related Work in Decision Tree Based Probability Estimation

Decision tree learning algorithms are a major type of effective learning algorithm in machine learning. In a decision boundary-based algorithm, an explicit decision boundary is extracted from the training samples, and a testing sample E is classified into class c if E falls into the decision area corresponding to c . However, traditional decision tree algorithms, such as C4.5 [10], have been observed to produce poor probability estimation [9]. Typically, the probability generated from decision tree is calculated from the sub sample sets at leaves corresponding to the conjunction of the conditions along the paths back to the root [8]. For example, if a leaf node defines a subset of 100 samples, 90 of which are in the positive class and others are in the negative class, then each sample is assigned the same probability of 0.9 (90/100) that it belongs to the positive class. More specifically, $\hat{p}(+|\mathbf{A}_p = \mathbf{a}_p)$ equals 90%, where \mathbf{A}_p is the set of attributes on the path. Viewed as probability estimators, decision trees consist of piecewise uniform approximations within regions defined by axis-paralleled boundaries. Aiming at this fact, Provost and Domingos [8] presented two methods to improve the probability estimation of decision tree. First, by using Laplace estimation, probability estimates can be smoothed from small sample data at the tree leaves. Second, by turning off pruning and "collapsing" in C4.5, decision trees can generate finer trees to give more precise probability estimation. The final version is called C4.4. In this paper, we compare our new algorithm with C4.4 and its variants.

Another alternative improvement to tackle the "uniform probability distribution" problem of decision trees is to stop splitting at a certain level and put another probability density estimator at each leaf. [5] proposed an NBTree that uses decision tree as the general structure and deploys naïve Bayes classifiers at the leaves. This learning algorithm first uses classification accuracy as the score function to do univariate splits and when splitting does not increase the score function, a naïve Bayesian classifier is created at the leaf. Thus, sample attributes are divided into two sets: $\mathbf{A} = \mathbf{A}_p \cup \mathbf{A}_l$, where \mathbf{A}_p is the set of path attributes and \mathbf{A}_l is the set of leaf attributes. [5] showed the improvement of classification accuracy but it did not mention the performance on probability estimation. [13] proposed one encode of $p(C, \mathbf{A})$ for NBTree. The proposed Conditional Independent Tree (CITree) denotes $p(\mathbf{A}, C)$ as below:

$$p(\mathbf{A}, C) = \alpha p(C|\mathbf{A}_p(L))p(\mathbf{A}_l(L)|\mathbf{A}_p(L), C) \quad (3)$$

where α is a normalization factor. The term $p(C|\mathbf{A}_p(L))$ is the joint conditional distribution of path attributes and the term $p(\mathbf{A}_l(L)|\mathbf{A}_p(L), C)$ is the leaf attributes presented by naïve Bayes. From the conditional independence assumption of naïve Bayes, the following equation stands:

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

CITree explicitly defines conditional dependence among the path attributes and independence among the leaf attributes. The local conditional independence assumption of CITree is a relaxation of the (global) conditional independence assumption of naïve Bayes. Further study in [11] reveals that the local conditional independence explains the replication problem. And one CITree can be decomposed into a set of trees to eliminate the replicated subtrees. The complexity of the sum of the trees is not greater than the original one.

Building decision trees with accurate probability estimation, called Probability Estimation Trees (PETs), has received a great deal of attention recently [8]. The difference of PET and CITree is that PET represents the conditional probability distribution of the path attributes, while a CITree represents a joint distribution over all attributes.

Another related work involves *Bayesian networks* [7]. Bayesian networks are directed acyclic graphs that encode conditional independence among a set of random variables. Each variable is independent of its non-descendants in the graph given the state of its parents. Tree Augmented Naïve Bayes (TAN), proposed by [3], approximates the interaction between attributes by using a tree structure imposed on the naïve Bayesian framework. We need to point out that, although TAN takes advantage of tree structure, it is not similar to a decision tree. Indeed, decision trees divide a sample space into multiple subspaces and local conditional probabilities are independent among those subspaces. Therefore, attributes in decision trees can repeatedly appear, while TAN describes the joint probabilities among attributes, so each attribute appears only once. In decision trees, $p(C|\mathbf{A})$ is decomposable when a (sub) sample set is split into subspaces, but it is non-decomposable in TAN.

3 Learning Naïve Bayes Tree for Conditional Probability Estimation

3.1 The performance evaluation metrics

Accurate conditional probability $p(C|\mathbf{A})$ is important for many applications. Since it is justified that $\log p$ is a monotonic function of p and we use conditional log likelihood (CLL) for calculation, we mix the usage of CLL and conditional

probability hereafter. In cost-sensitive classification, the optimal prediction for a sample \mathbf{b} is the class c_i that minimize [2]

$$h(\mathbf{b}) = \arg \min_{c_i \in C} \sum_{c_j \in C - c_i} p(c_j | \mathbf{b}) C(c_i, c_j) \quad (5)$$

One can see that the score function in cost-sensitive learning directly relies on the conditional probability. It is not like the classification problem where only the decision boundary is important. Accurate estimation of conditional probability is necessary for cost-sensitive learning.

Better conditional probability estimation means better classification accuracy (ACC) (c.f. Equation 1). ACC is calculated as the percentage of the correctly classified samples over all the samples: $ACC = \frac{1}{N} \sum I(h(\mathbf{a}) = c)$, here N is the number of samples. However, improving conditional probability estimation is not a necessary condition for improving ACC. ACC can be scaled up through other ways, e.g. boundary-based approaches. On the other side, even if conditional probability is greatly improved, it may still lead to wrong classification. For instance, presume that $E+$ is a positive sample while it is misclassified into the negative class with a class probability estimate $\hat{p}(+|E+) = 0.3$. If the classification threshold is 0.5, an algorithm that improves this class probability estimate to $\hat{p}(+|E+) = 0.4$, still gives the incorrect result. Therefore, in this case, CLL does not improve ACC. However, at least, more precise estimation of CLL would not worsen ACC.

Ranking is different from both classification and probability estimation. For example, assume that $E+$ and $E-$ are a positive and a negative sample respectively, and that the actual class probabilities are $p(+|E+) = 0.9$ and $p(-|E-) = 0.1$. An algorithm that gives class probability estimates $\hat{p}(+|E+) = 0.55$ and $\hat{p}(+|E-) = 0.54$, gives a correct order of $E+$ and $E-$ in the ranking. Notice that the probability estimates are poor and the classification for $E-$ is incorrect. However, if a learning algorithm produces accurate class probability estimates, it certainly produces a precise ranking. Thus, aiming at learning a model to yield accurate conditional probability estimation will usually lead to a model yielding precise probability-based ranking.

In this paper, we use three different metrics CLL, ACC and AUC to evaluate learning algorithms.

3.2 The representation of CLL in CLLTree

The representation of conditional probability in the diagram of CLLTree is as follows:

$$\log(p(C|\mathbf{A})) = \log(p(C|\mathbf{A}_1, \mathbf{A}_p)) = \log(p(C|\mathbf{A}_p)) + \log(p(\mathbf{A}_1|C, \mathbf{A}_p)) - \log(p(\mathbf{A}_1|\mathbf{A}_p)) \quad (6)$$

\mathbf{A}_p divides a (sub) sample set into several subsets. All decomposed terms are the conditional probability of \mathbf{A}_p . $p(C|\mathbf{A}_p)$ is the conditional probability on the path attributes; $p(\mathbf{A}_1|C, \mathbf{A}_p)$ is the naïve Bayesian classifier at a leaf; and $p(\mathbf{A}_1|\mathbf{A}_p)$ is the joint probability of \mathbf{A}_1 under condition of \mathbf{A}_p .

In each step of generating the decision tree, CLL is calculated based on Equation 6. Assuming A_{li} denotes a leaf attribute, here, $p(C|\mathbf{A}_p)$ is calculated by the ratio of the number of samples that have the same class value to all the samples at a leaf; $p(\mathbf{A}_l|C, \mathbf{A}_p)$ can be represented by $\prod_{i=1}^m p(A_{li}|C, \mathbf{A}_p)$ (m is the number of attributes at a leaf node), and each $p(A_{li}|C, \mathbf{A}_p)$ can be calculated by the ratio of the number of samples that have the same attribute value of A_{li} and the same class value to the number of samples that have the same class value; likewise, $p(\mathbf{A}_l|\mathbf{A}_p)$ can also be represented by $\prod_{i=1}^m p(A_{li}|\mathbf{A}_p)$, and each $p(A_{li}|\mathbf{A}_p)$ can be calculated by the ratio of the number of samples that have the same attribute value of A_{li} to the number of samples at that leaf. The attribute to optimize CLL is selected as the next level node to extend the tree. We exhaustively build all possible trees in each step and keep only the best one for the next level expansion. Supposing finite k attributes are available. When expanding the tree at level q , there are $k - q + 1$ attributes to be chosen. This is a greedy way. CLLTree makes an assumption on probability, i.e. the probability dependency on the path attributes and the probability independency on the leaf attributes. Besides, it also has another assumption on the structure that each node has only one parent.

4 A New Algorithm for Learning CLLTree

From the discussion in the previous sections, CLLTree can represent any joint distribution. Therefore, the probability estimation based on CLLTree is accurate. But the structure learning of a CLLTree could theoretically be as time-consuming as learning an optimal decision tree. A good approximation of a CLLTree, which gives relatively accurate estimates of class probabilities, is desirable in many applications. Similar to a decision tree, building a CLLTree could be a greedy and recursive process. On each iteration, choose the “best” attribute as the root of the (sub) tree, split the associated data into disjoint subsets corresponding to the values of that attribute, and recur this process for each subset until certain criteria are satisfied. If the structure of a CLLTree is determined, a leaf naïve Bayes is a perfect model to represent the local conditional distribution at leaves. The algorithm is described below.

In our algorithm, we adopt a heuristic search process in which we choose an attribute with the greatest improvement on the performance of the resulting tree. Precisely speaking, on each iteration, each candidate attribute is chosen as the root of the (sub) tree, the resulting tree is evaluated, and we choose the attribute that achieves the highest CLL value. We consider two criteria for halting the search process. For one, we could stop splitting when none of the alternative attributes significantly improve probability estimation, in the form of CLL. Or, to make a leaf naïve Bayes work accurately, there are at least 30 samples at the current leaf. We define a split to be significant if the relative reduction in CLL is greater than 5%. Note that we train a leaf naïve Bayes by adopting an inner 5-fold cross-validation on the sub sample set S which fall into the current leaf. For example, if an attribute has 3 attribute values which will

Algorithm 1 Learning Algorithm $CLLTree(T, E, A)$

 T : CLLTree S : a set of labeled samples A : a set of attributes

```

for each attribute  $a \in A$  do
  Partition  $S$  into  $S_1, S_2, \dots, S_k$ , where  $k$  is the number of possible values of
  attribute  $a$ . Each sub set is corresponding to a value of  $a$ . For continuous
  attributes, a threshold is set up in this step.
  Create a naïve Bayes for each  $S_i$ .
  Evaluate the split on the attribute  $a$  in terms of CLL.
  Choose the attribute  $A_t$  with the highest split CLL.
if the split CLL is not improved greatly than the CLL of attribute  $A_t$  then
  create a leaf naïve Bayes for this attribute.
else
  for all values  $S_a$  of  $A_t$  do
     $CLLTree(T_a, S_a, A - A_t)$ .
  add  $T_a$  as a child of  $T$ 
Return  $T$ 

```

result in three leaf naïve Bayes, the inner 5-fold cross-validations will be run in three leaves. Furthermore, we compute CLL by putting the samples from all the leaves together rather than computing the CLL for each leaf separately.

It is also worth noting, however, the different biases between learning a CLL-Tree and learning a traditional decision tree. In decision tree, the building process is directed by the purity of the (sub) sample set measured by information gain, and the crucial point in selecting an attribute is whether the resulting split of the samples is “pure” or not. However, such a selection strategy does not necessarily lead to the truth of improving the probability estimation of a new sample. In building a CLLTree, we intend to choose the attributes that maximize the posterior class probabilities $p(C|\mathbf{A})$ among the samples at the current leaf as much as possible. That means, even though there possibly exists the high impurity of its leaves, it could still be a good CLLTree. Therefore, traditional decision tree learning algorithms are not straightforwardly suitable for learning CLLTree.

5 Experimental Methodology and Results

For the purpose of our study, we used 33 well-recognized sample sets from many domains recommended by Weka [12]. There is a brief description of these sample sets in Table 1. All sample sets came from the UCI repository[1].

The preprocessing stages of sample sets were carried out within the Weka platform, mainly including the following three steps:

1. Applying the filter of ReplaceMissingValues in Weka to replace the missing values of attributes.

Table 1. Description of sample sets used by the experiments.

Data Set	Size	Attr.	Classes	Missing	Numeric
anneal	898	39	6	Y	Y
anneal.ORIG	898	39	6	Y	Y
audiology	226	70	24	Y	N
balance	625	5	3	N	Y
breast	286	10	2	Y	N
breast-w	699	10	2	Y	N
colic	368	23	2	Y	Y
colic.ORIG	368	28	2	Y	Y
credit-a	690	16	2	Y	Y
credit-g	1000	21	2	N	Y
diabetes	768	9	2	N	Y
glass	214	10	7	N	Y
heart-c	303	14	5	Y	Y
heart-h	294	14	5	Y	Y
heart-s	270	14	2	N	Y
hepatitis	155	20	2	Y	Y
hypoth.	3772	30	4	Y	Y
iris	150	5	3	N	Y
kr-vs-kp	3196	37	2	N	N
labor	57	17	2	Y	Y
letter	20000	17	26	N	Y
lymph	148	19	4	N	Y
mushroom	8124	23	2	Y	N
p.-tumor	339	18	21	Y	N
segment	2310	20	7	N	Y
sick	3772	30	2	Y	Y
soybean	683	36	19	Y	N
splice	3190	62	3	N	N
vehicle	846	19	4	N	Y
vote	435	17	2	Y	N
vowel	990	14	11	N	Y
waveform	5000	41	3	N	Y
zoo	101	18	7	N	Y

- Applying the filter of Discretize in Weka to discretize numeric attributes. Therefore, all the attributes are treated as nominal.
- It is well known that, if the number of values of an attribute is almost equal to the number of samples in a sample set, this attribute does not contribute any information to classification. So we used the filter of Remove in Weka to delete these attributes. Three occurred within the 33 sample sets, namely Hospital Number in sample set horse-colic.ORIG, Instance Name in sample set Splice and Animal in sample set zoo.

To avoid the zero-frequency problem, we used the Laplace estimation. More precisely, assuming that there are n_c samples that have the class label as c , t total samples, and k class values in a sample set. The frequency-based probability estimation calculates the estimated probability by $p(c) = \frac{n_c}{t}$. The Laplace estimation calculates it as $p(c) = \frac{n_c+1}{t+k}$. In the Laplace estimation, $p(a_i|c)$ is calculated by $p(a_i|c) = \frac{n_{ic}+1}{n_c+v_i}$, where v_i is the number of values of attribute A_i and n_{ic} is the number of samples in class c with $A_i = a_i$.

In our experiments, two groups of comparisons have been performed. We compared CLLTree with naïve Bayesian related algorithms, such as NBTree, NB, TAN; and with PETs variant algorithms, such as C4.4, C4.4-B(C4.4 with bag-

ging), C4.5-L(C4.5 with Laplace estimation) and C4.5-B(C4.5 with bagging). We implemented CLLTree within the Weka framework [12], and used the implementation of other learning algorithms in Weka. In all experiments, the experimental result for each algorithm was measured via a ten-fold cross validation. Runs with various algorithms were carried out on the same training sets and evaluated on the same test sets. In particular, the cross-validation folds were the same for all the experiments on each sample set. Finally, we conducted two-tailed t -test with a significantly different probability of 0.95 to compare our algorithm with others. That is, we speak of two results for a sample set as being “significantly different” only if the difference is statistically significant at the 0.05 level according to the corrected two-tailed t -test [6].

Table 2 and 4 show the experimental results in terms of CLL and AUC. The corresponding summaries of t -test results are demonstrated in Table 3 and 5. Multi-class AUC has been calculated by M-measure[4] in our experiments. Table 6 and 7 are used to display the ACC comparison and t -test results respectively. In all t -test tables, each entry $w/t/l$ means that the algorithm in the corresponding row wins in w sample sets, ties in t sample sets, and loses in l sample sets, in contrast with the algorithm in the corresponding column.

Now, our observations are summarized as follows.

1. CLLTree outperforms NBTree in terms of CLL and AUC significantly, and slightly better in ACC. The detailed results in CLL (Table 3) show that CLLTree wins in 10 sample sets, ties in 23 sample sets and loses in 0 sample sets. In AUC (Table 5), CLLTree wins in 5 sample sets, ties in 27 sample sets and loses only in one. In addition, CLLTree surpasses NBTree in the ACC performance as well. It wins in 3 sample sets and loses in 1 sample set.
2. CLLTree is the best among the rest of learning algorithms in AUC. Compared with C4.4, it wins in 19 sample sets, ties in 14 sample sets and loses in 0 sample sets. Since C4.4 is the state-of-art decision tree algorithm designed specifically for yielding accurate ranking, this comparison also provides evidence to support CLLTree. Compared with naïve Bayes, our algorithm also wins in 9 sample sets, ties in 21 sample sets and loses in 3 sample sets.
3. In terms of the average classification accuracy (Table 6), CLLTree achieves the highest ACC among all algorithms. Compared with naïve Bayes, it wins in 11 sample sets, ties in 21 sample sets and loses in 1 sample set. The average ACC for naïve Bayes is 82.82%, lower than that of CLLTree. Furthermore, CLLTree also outperforms TAN significantly. It wins 6 sample sets, ties in 24 sample sets and loses in 3 sample sets. The average ACC for TAN is 84.64%, which is lower than our algorithm as well. And last, CLLTree is also better than C4.5, the implementation of traditional decision trees, in 8 sample sets.
4. Although C4.4 outperforms CLLTree in CLL, our algorithm is definitely better than C4.4 in the overall performance. C4.4 sacrifices its tree size to improve probability estimation, which could produce the “overfitting” problem and will be noise sensitive. Therefore, in a practical perspective, CLLTree is more suitable for many real applications.

From our experiments, we also made one interesting observation that in terms of training time CLLTree is tremendously better than NBTree. It wins in 30 sample sets, ties in 2 sample sets and loses in 1 sample set. We did not present the experimental results in this paper due to space limitation.

6 Conclusion

In this paper, we have proposed a novel algorithm CLLTree to improve probability estimation in NBTree. The empirical results prove our expectation that CLL and AUC are significantly improved and ACC is slightly better compared to other classic learning algorithms. There is still room to improve probability estimation. For example, after the structure is learned, we can use parameter learning algorithms to tune the conditional probability estimates on the path attributes. And we can find the right tree size for our model, i.e. possibly using model-selection criteria to decide when to stop the splitting.

References

1. C. Blake and C.J. Merz. Uci repository of machine learning database.
2. Charles Elkan. The foundations of cost-sensitive learning. In *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence*, 1991.
3. N. Friedman, D. Geiger, and M. Goldszmidt. Bayesian network classifiers. *Machine Learning*, 29, 1997.
4. D. J. Hand and R. J. Till. A simple generalisation of the area under the roc curve for multiple class classification problems. *Machine Learning*, 45, 2001.
5. Ron Kohavi. Scaling up the accuracy of naive-bayes classifiers: a decision-tree hybrid. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, 1996.
6. C. Nadeau and Y. Bengio. Inference for the generalization error. *Machine Learning*, 52(40), 2003.
7. J. Pearl. *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann, 1988.
8. F. J. Provost and P. Domingos. Tree induction for probability-based ranking. *Machine Learning*, 52(30), 2003.
9. F. J. Provost, T. Fawcett, and R. Kohavi. The case against accuracy estimation for comparing induction algorithms. In *Proceedings of the Fifteenth International Conference on Machine Learning*. Morgan Kaufmann, 1998.
10. J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann: San Mateo, CA, 1993.
11. J. Su and H. Zhang. Representing conditional independence using decision trees. In *Proceedings of the Twentieth National Conference on Artificial Intelligence (AAAI-05)*. AAAI Press, 2005.
12. I. H. Witten and E. Frank. *Data Mining – Practical Machine Learning Tools and Techniques with Java Implementation*. Morgan Kaufmann, 2000.
13. H. Zhang and J. Su. Conditional independence trees. In *Proceedings of the 15th European Conference on Machine Learning (ECML2004)*. Springer, 2004.

Table 2. Experimental results for CLLTree versus Naïve Bayes Tree (NBTree), naïve Bayes (NB) and Tree Augmented Naïve Bayes (TAN); C4.4, C4.4 with bagging (C4.4-B) and C4.5 with Laplace estimation (C4.5-L): Conditional Log Likelihood (CLL) & standard deviation.

Data Set	CLLTree	NBTree	NB	TAN	C4.4	C4.4-B
anneal	-10.78±15.51	-18.46±17.63	-14.22±6.16	-6.29±5.36	-7.84±2.58	-13.74±1.78
anneal.ORIG	-22.28±11.36	-33.33±16.32 ●	-23.58±5.60	-19.55±6.90	-22.17±3.74	-40.13±4.44 ●
audiology	-75.58±47.43	-95.28±41.89	-65.91±24.28	-67.19±24.11	-15.37±3.39 ○	-35.95±3.02 ○
balance-scale	-29.81±1.51	-31.75±1.51 ●	-31.75±1.51 ●	-34.78±3.10 ●	-52.78±4.03 ●	-46.71±2.46 ●
breast-cancer	-18.88±4.25	-20.47±5.23	-18.37±4.49	-18.17±3.60	-18.56±2.70	-17.07±2.13
breast-w	-11.43±6.38	-17.47±13.63	-18.28±14.16	-12.14±6.76	-11.17±3.39	-10.13±2.50
colic	-30.82±13.42	-34.42±17.34	-30.63±11.38	-26.22±9.35	-17.80±4.31 ○	-15.18±3.36 ○
colic.ORIG	-24.96±10.60	-38.50±17.60 ●	-21.24±5.74	-22.36±6.24	-17.66±3.19 ○	-16.09±2.26 ○
credit-a	-26.98±6.93	-34.52±11.89 ●	-28.79±8.10	-28.07±7.06	-28.06±4.92	-26.58±4.14
credit-g	-52.61±6.18	-62.44±23.22	-52.79±6.35	-56.16±8.09	-61.03±5.85 ●	-53.68±3.87
diabetes	-40.30±7.20	-42.70±9.11	-40.78±7.49 ●	-42.51±8.23	-43.05±4.79	-40.19±4.08
glass	-26.06±8.73	-31.06±9.62	-24.08±5.42	-26.15±6.27	-21.02±2.70	-29.77±1.98
heart-c	-17.92±8.48	-15.70±7.49	-13.91±6.71	-14.01±6.09	-15.85±3.68	-25.93±2.79 ●
heart-h	-15.93±7.24	-14.73±5.94	-13.49±5.37	-12.96±4.06	-14.78±3.16	-24.12±3.09 ●
heart-statlog	-12.01±4.74	-16.31±9.29	-12.25±4.96	-14.60±5.39 ●	-14.00±3.33	-12.61±2.15
hepatitis	-9.38±5.98	-9.18±5.78	-8.53±5.98	-8.16±4.72	-6.81±2.51	-6.20±1.64
hypothyroid	-95.50±13.68	-98.23±14.58	-97.14±13.29	-93.72±12.69	-90.14±5.73	-104.87±5.70 ●
iris	-2.73±2.55	-2.69±2.90	-2.56±2.35	-3.12±2.30	-3.63±1.35	-4.01±1.23 ●
kr-vs-kp	-18.39±14.46	-28.01±18.07	-93.48±7.65 ●	-60.27±7.38 ●	-8.65±3.50 ○	-7.92±2.75 ○
labor	-1.50±2.90	-1.03±2.27	-0.71±0.99	-2.23±3.43	-2.22±1.28	-2.13±0.95
letter	-1853.63±168.85	-2193.71±159.62 ●	-2505.15±98.42 ●	-1272.27±68.92 ○	-1048.56±30.99 ○	-2927.76±40.84 ●
lymph	-9.16±7.43	-8.48±5.51	-6.22±3.96	-7.15±5.24	-7.75±2.64	-9.85±1.85
mushroom	0.00±0.01	-0.14±0.14	-105.77±23.25 ●	-0.19±0.45	-2.10±0.19 ●	-2.18±0.20 ●
primary-tumor	-74.57±12.93	-74.19±14.56	-65.56±8.27 ○	-69.75±8.85	-50.98±3.70 ○	-82.41±3.45
segment	-61.82±22.64	-111.94±45.14 ●	-124.32±33.74 ●	-40.15±13.46 ○	-48.76±7.07	-97.61±6.49 ●
sick	-24.51±9.35	-45.55±19.82 ●	-46.05±11.99 ●	-28.91±8.80 ●	-21.10±5.56 ○	-19.66±4.67 ○
soybean	-17.39±11.73	-28.63±15.19	-26.25±11.03	-8.06±3.84 ○	-18.39±3.31	-61.37±4.69 ●
splice	-46.58±12.76	-47.11±13.57	-46.53±12.85	-46.89±11.95	-66.48±8.24 ●	-78.71±6.91 ●
vehicle	-98.66±21.48	-137.97±32.69 ●	-172.12±27.55 ●	-57.52±10.16 ○	-55.24±4.50 ○	-70.21±3.09 ○
vote	-7.78±5.33	-7.35±5.41	-27.25±13.85 ●	-7.91±5.39	-6.90±3.56	-6.10±3.25
vowel	-38.23±21.06	-45.93±16.44	-89.80±11.38 ●	-21.87±8.84 ○	-71.55±6.18 ●	-152.25±4.88 ●
waveform-5000	-228.39±19.84	-309.13±43.99 ●	-378.00±32.64 ●	-254.80±23.42 ●	-318.55±12.98 ●	-351.30±9.38 ●
zoo	-2.14±2.63	-1.29±1.68	-1.22±1.06	-1.07±1.44	-2.74±1.28	-4.59±1.00 ●

●, ○ statistically significant degradation or improvement compared with CLLTree

Table 3. Summary on *t*-test of experimental results: CLL comparisons on CLL-Tree, NBTree, NB, TAN, C4.4 and C4.4-B. An entry *w/t/l* means that the algorithm at the corresponding row wins in *w* sample sets, ties in *t* sample sets, and loses in *l* sample sets, compared to the algorithm at the corresponding column.

	C4.4-B	C4.4	TAN	NB	NBTree
C4.4	19/7/7				
TAN	16/12/5	8/17/8			
NB	14/9/10	5/14/14	3/18/12		
NBTree	10/15/8	5/16/12	2/20/11	7/22/4	
CLLTree	14/13/6	6/19/8	5/23/5	11/21/1	10/23/0

Table 4. Experimental results for CLLTree versus Naïve Bayes Tree (NBTree), naïve Bayes (NB) and Tree Augmented Naïve Bayes (TAN); C4.4 and C4.4 with bagging (C4.4-B): Area Under the Curve (AUC) & standard deviation.

Data Set	CLLTree	NBTree	NB	TAN	C4.4	C4.4-B
anneal	95.97±1.33	96.31±1.10	96.18±1.07	96.59±0.13	93.67±6.25	94.48±5.78
anneal.ORIG	93.73±7.40	93.55±7.60	94.50±4.03	95.26±2.80	91.01±8.07 ●	93.21±7.91
audiology	70.36±1.17	70.17±1.26	70.02±1.09	70.25±1.05	64.04±2.38 ●	69.05±1.84 ●
balance-scale	84.69±4.31	84.64±4.34	84.64±4.34	78.34±5.04 ●	61.40±6.73 ●	66.55±6.03 ●
breast-cancer	68.00±10.45	67.98±10.28	70.18±10.88	66.18±10.71	60.53±10.08 ●	64.55±10.49
breast-w	98.64±1.04	99.25±0.72 ○	99.25±0.73 ○	98.72±1.04	98.22±1.25	98.83±1.01
colic	82.08±8.30	86.78±7.24	84.36±6.95	85.04±6.45	83.96±7.41	88.20±6.63 ○
colic.ORIG	81.95±7.24	79.83±8.33	81.18±7.47	81.93±6.93	83.00±6.55	85.98±5.38
credit-a	92.06±3.11	91.34±3.25	91.86±3.11	91.35±3.21	89.59±3.81 ●	90.53±3.59 ●
credit-g	79.14±4.11	77.53±5.24	79.10±4.14	77.92±4.82	70.07±4.70 ●	74.21±4.74 ●
diabetes	82.57±4.97	82.11±5.06	82.61±4.97	81.33±5.19	76.20±5.10 ●	79.10±5.45 ●
glass	82.17±5.93	79.13±6.39	78.42±6.03 ●	78.32±6.12 ●	80.11±6.91	80.30±7.12
heart-c	83.89±0.62	84.00±0.58	84.11±0.56	84.03±0.59	83.27±0.75 ●	83.65±0.64
heart-h	83.87±0.58	83.90±0.59	84.00±0.56	83.88±0.55	83.30±0.64 ●	83.64±0.62
heart-statlog	91.34±4.84	89.83±5.82	91.34±4.85	88.19±5.75 ●	82.81±8.28 ●	86.51±6.67 ●
hepatitis	83.48±12.95	85.69±11.66	89.36±10.03	86.06±10.46	79.50±14.28	82.43±13.79
hypothyroid	88.23±6.67	87.66±6.75	88.10±6.49	87.84±7.10	80.62±8.24 ●	81.44±7.56 ●
iris	98.72±2.02	98.85±2.00	98.99±1.69	98.49±2.44	98.67±1.98	98.77±2.09
kr-vs-kp	99.82±0.20	99.44±0.60	95.19±1.19 ●	98.06±0.63 ●	99.93±0.08	99.97±0.05 ○
labor	95.29±14.62	96.63±11.83	98.67±5.39	93.75±14.34	87.17±18.05	90.79±15.51
letter	99.36±0.10	98.51±0.17 ●	96.91±0.21 ●	99.12±0.09 ●	95.52±0.32 ●	98.41±0.21 ●
lymph	89.12±2.74	88.94±2.81	90.25±1.57	89.16±3.28	86.30±4.58	88.17±3.11
mushroom	100.00±0.00	100.00±0.00	99.80±0.07 ●	100.00±0.00	100.00±0.00	100.00±0.00
primary-tumor	75.33±2.88	74.71±2.75	75.58±2.75	75.43±2.67	68.53±3.05 ●	73.05±3.07 ●
segment	99.40±0.24	99.11±0.33 ●	98.35±0.45 ●	99.63±0.18 ○	99.08±0.42 ●	99.49±0.28
sick	98.44±1.47	94.46±3.47 ●	95.87±2.31 ●	98.31±1.04	99.03±0.66	99.23±0.46
soybean	99.81±0.29	99.72±0.32	99.79±0.24	99.87±0.23	98.02±1.62 ●	98.95±1.26 ●
splice	99.45±0.27	99.44±0.31	99.46±0.27 ○	99.40±0.35	98.06±0.71 ●	98.74±0.58 ●
vehicle	86.68±2.53	85.86±3.30	80.58±3.04 ●	91.14±1.89 ○	85.96±2.75	89.02±2.16 ○
vote	98.50±1.44	98.61±1.57	97.15±2.00 ●	98.78±1.22	97.43±2.37	98.31±2.18
vowel	99.35±0.60	98.59±0.80 ●	95.98±1.07 ●	99.64±0.25	91.57±2.34 ●	96.44±1.58 ●
waveform-5000	94.74±0.67	93.71±0.96 ●	95.32±0.68 ○	93.87±0.80 ●	81.36±1.41 ●	90.04±1.28 ●
zoo	88.64±2.95	89.02±2.99	88.88±2.83	88.93±2.82	80.26±6.35 ●	80.88±6.71 ●
average	89.83±3.58	89.55±3.65	89.58±3.12	89.54±3.34	85.70±4.49	87.97±4.11

●, ○ statistically significant degradation or improvement compared with CLLTree

Table 5. Summary on *t*-test of experimental results: AUC comparisons on CLLTree, NBTree, NB, TAN, C4.4 and C4.4-B.

	C4.4-B	C4.4	TAN	NB	NBTree
C4.4	0/15/18				
TAN	12/19/2	18/13/2			
NB	14/12/7	21/7/5	4/20/9		
NBTree	8/20/5	19/12/2	3/25/5	7/25/1	
CLLTree	14/16/3	19/14/0	6/25/2	9/21/3	5/27/1

Table 6. Experimental results for CLLTree versus Naïve Bayes Tree (NBTree), naïve Bayes (NB) and Tree Augmented Naïve Bayes (TAN); C4.5, C4.5 with Laplace estimation (C4.5-L), and C4.5 with bagging (C4.5-B): Classification Accuracy (ACC) & standard deviation.

Data Set	CLLTree	NBTree	NB	TAN	C4.5	C4.5-L	C4.5-B
anneal	99.06±1.00	98.40±1.53	94.32±2.23	● 98.34±1.18	98.65±0.97	98.76±0.99	98.76±0.89
anneal.ORIG	89.94±3.33	91.27±3.03	88.16±3.06	90.88±2.55	90.36±2.51	90.23±2.54	91.78±2.44
audiology	78.40±8.63	76.66±7.47	71.40±6.37	● 72.68±7.02	● 77.22±7.69	76.69±7.68	80.67±7.31
balance-scale	91.44±1.30	91.44±1.30	91.44±1.30	86.22±2.82	● 64.14±4.16	● 64.14±4.16	● 73.30±5.38
breast-cancer	72.14±7.19	71.66±7.92	72.94±7.71	70.09±7.68	75.26±5.04	75.26±5.04	73.09±5.75
breast-w	95.08±2.48	97.23±1.76	97.30±1.75	○ 94.91±2.37	94.01±3.28	93.81±3.29	95.34±2.71
colic	78.08±7.38	82.50±6.51	78.86±6.05	80.57±5.90	84.31±6.02	○ 84.50±6.04	○ 84.56±6.21
colic.ORIG	75.57±6.49	74.83±7.82	74.21±7.09	76.11±6.04	80.79±5.66	○ 80.08±5.61	○ 82.64±5.44
credit-a	85.13±4.10	84.86±3.92	84.74±3.83	84.43±4.51	85.06±4.12	84.97±4.09	85.83±4.20
credit-g	76.01±3.76	75.54±3.92	75.93±3.87	75.86±3.58	72.61±3.49	● 72.25±3.48	● 73.89±3.78
diabetes	75.63±4.81	75.28±4.84	75.68±4.85	75.09±4.96	73.89±4.70	73.88±4.64	73.91±4.63
glass	58.69±10.15	58.00±9.42	57.69±10.07	58.43±8.86	58.14±8.48	58.28±8.52	57.98±8.99
heart-c	80.54±6.83	81.10±7.24	83.44±6.27	82.85±7.20	79.14±6.44	79.41±6.58	79.48±6.20
heart-h	81.41±6.56	82.46±6.26	83.64±5.85	82.14±6.20	80.10±7.11	80.03±7.15	80.90±7.08
heart-statlog	83.59±5.32	82.26±6.50	83.78±5.41	79.37±6.87	● 79.78±7.71	79.85±7.95	79.44±6.52
hepatitis	81.20±9.78	82.90±9.79	84.06±9.91	82.40±8.68	81.12±8.42	81.12±8.42	81.38±7.74
hypothyroid	92.90±0.73	93.05±0.65	92.79±0.73	93.23±0.68	93.24±0.44	93.24±0.44	93.25±0.45
iris	93.73±6.82	95.27±6.16	94.33±6.79	91.67±7.18	96.00±4.64	96.00±4.64	95.53±5.02
kr-vs-kp	98.93±0.65	97.81±2.05	87.79±1.91	● 92.05±1.49	● 99.44±0.37	● 99.44±0.37	○ 99.42±0.38
labor	93.93±10.94	95.60±8.39	96.70±7.27	90.33±10.96	84.97±14.24	84.97±14.24	85.23±13.11
letter	86.24±0.72	83.49±0.81	● 70.09±0.93	● 83.11±0.75	● 81.31±0.78	● 80.51±0.78	● 83.69±0.85
lymph	82.79±9.81	82.21±8.95	85.97±8.88	84.07±8.93	78.21±9.74	78.21±9.74	78.97±9.06
mushroom	100.00±0.00	100.00±0.00	95.52±0.78	● 99.99±0.03	100.00±0.00	100.00±0.00	100.00±0.00
primary-tumor	46.17±5.90	45.84±6.61	47.20±6.02	46.76±5.92	41.01±6.59	● 41.01±6.59	● 43.42±6.08
segment	93.13±1.34	92.64±1.61	89.03±1.66	● 94.54±1.60	○ 93.42±1.67	93.19±1.69	93.97±1.46
sick	97.80±0.82	97.86±0.69	96.78±0.91	● 97.61±0.73	98.16±0.68	98.18±0.67	98.17±0.71
soybean	93.07±2.57	92.30±2.70	92.20±3.23	95.24±2.28	○ 92.63±2.72	92.55±2.61	93.66±2.70
splice	95.39±1.15	95.42±1.14	95.42±1.14	95.39±1.16	94.17±1.28	● 94.08±1.29	● 94.51±1.28
vehicle	68.83±4.01	68.91±4.58	61.03±3.48	● 73.71±3.48	○ 70.74±3.62	70.38±3.69	71.93±4.07
vote	94.65±3.11	94.78±3.32	90.21±3.95	● 94.57±3.23	96.27±2.79	96.27±2.79	96.32±2.65
vowel	91.59±3.53	88.01±3.71	● 66.09±4.78	● 93.10±2.85	75.57±4.58	● 73.29±4.69	● 79.44±3.73
waveform-5000	84.40±1.64	81.62±1.76	● 79.97±1.46	● 80.72±1.78	● 72.64±1.81	● 72.21±1.79	● 75.54±1.83
zoo	93.86±6.42	94.55±6.54	94.37±6.79	96.73±5.45	92.61±7.33	92.61±7.33	93.51±7.16
average	85.13±4.52	85.02±4.51	82.82±4.43	84.64±4.39	82.87 ±4.51	82.70±4.53	83.92±4.41

●, ○ statistically significant degradation or improvement compared with CLLTree

Table 7. Summary on *t*-test of experimental results: ACC comparisons on CLL-Tree, NBTree, NB, TAN, C4.5, C4.5-L and C4.5-B.

	C4.5	C4.5-L	C4.5-B	TAN	NB	NBTree
C4.5-L	3/30/0					
C4.5-B	6/27/0	7/26/0				
TAN	8/22/3	10/19/4	3/25/5			
NB	8/13/12	8/14/11	5/15/13	3/19/11		
NBTree	7/24/2	8/24/1	5/25/3	3/26/4	11/22/0	
CLLTree	8/23/2	7/23/3	4/26/3	6/24/3	11/21/1	3/29/1