A Fast Decision Tree Learning Algorithm

Jiang Su and **Harry Zhang**
Faculty of Computer Science
University of New Brunswick, NB, Canada, E3B 5A3
{jiang.su, hzhang}@unb.ca

Abstract

There is growing interest in scaling up the widely-used decision-tree learning algorithms to very large data sets. Although numerous diverse techniques have been proposed, a fast tree-growing algorithm without substantial decrease in accuracy and substantial increase in space complexity is essential. In this paper, we present a novel, fast decision-tree learning algorithm that is based on a conditional independence assumption. The new algorithm has a time complexity of $O(m \cdot n)$, where m is the size of the training data and n is the number of attributes. This is a significant asymptotic improvement over the time complexity $O(m \cdot n^2)$ of the standard decision-tree learning algorithm C4.5, with an additional space increase of only $O(n)$. Experiments show that our algorithm performs competitively with C4.5 in accuracy on a large number of UCI benchmark data sets, and performs even better and significantly faster than C4.5 on a large number of text classification data sets. The time complexity of our algorithm is as low as naive Bayes'. Indeed, it is as fast as naive Bayes but outperforms naive Bayes in accuracy according to our experiments. Our algorithm is a core tree-growing algorithm that can be combined with other scaling-up techniques to achieve further speedup.

Introduction and Related Work

Decision-tree learning is one of the most successful learning algorithms, due to its various attractive features: simplicity, comprehensibility, no parameters, and being able to handle mixed-type data. In decision-tree learning, a decision tree is induced from a set of labeled training instances represented by a tuple of attribute values and a class label. Because of the vast search space, decision-tree learning is typically a greedy, top-down and recursive process starting with the entire training data and an empty tree. An attribute that best partitions the training data is chosen as the splitting attribute for the root, and the training data are then partitioned into disjoint subsets satisfying the values of the splitting attribute. For each subset, the algorithm proceeds recursively until all instances in a subset belong to the same class. A typical tree-growing algorithm, such as C4.5 (Quinlan 1993), has a time complexity of $O(m \cdot n^2)$, where m is the size of the training data and n is the number of attributes.

Since large data sets with millions of instances and thousands of attributes are not rare today, the interest in developing fast decision-tree learning algorithms is rapidly growing. The major reason is that decision-tree learning works comparably well on large data sets, in addition to its general attractive features. For example, decision-tree learning outperforms naive Bayes on larger data sets, while naive Bayes performs better on smaller data sets (Kohavi 1996; Domingos & Pazzani 1997). A similar observation has been noticed in comparing decision-tree learning with logistic regression (Perlich, Provost, & Simonoff 2003).

Numerous techniques have been developed to speed up decision-tree learning, such as designing a fast tree-growing algorithm, parallelization, and data partitioning. Among them, a large amount of research work has been done on reducing the computational cost related to accessing the secondary storage, such as SLIQ (Mehta, Agrawal, & Rissanen 1996), SPRINT (Shafer, Agrawal, & Mehta 1996), or Rainforest (Gehrke, Ramakrishnan, & Ganti 2000). An excellent survey is given in (Provost & Kolluri 1999). Apparently, however, developing a fast tree-growing algorithm is more essential. There are basically two approaches to designing a fast tree-growing algorithm: searching in a restricted model space, and using a powerful search heuristic.

Learning a decision tree from a restricted model space achieves the speedup by avoiding searching the vast model space. One-level decision trees (Holte 1993) are a simple structure in which only one attribute is used to predict the class variable. A one-level tree can be learned quickly with a time complexity of $O(m \cdot n)$, but its accuracy is often much lower than C4.5's. Auer *et al.* (1995) present an algorithm $T2$ for learning two-level decision trees. However, It has been noticed that $T2$ is no more efficient than even C4.5 (Provost & Kolluri 1999).

Learning restricted decision trees often leads to performance degradation in some complex domains. Using a powerful heuristic to search the unrestricted model space is another realistic approach. Indeed, most standard decision-tree learning algorithms are based on heuristic search. Among them, the decision tree learning algorithm C4.5 (Quinlan 1993) has been well recognized as the reigning standard. C4.5 adopts information gain as the criterion (heuristic) for splitting attribute selection and has a time complexity of $O(m \cdot n^2)$. Note that the number n of attributes corresponds to the depth of the decision tree, which is an important factor contributing to the computational cost for tree-growing. Although one empirical study suggests that on average the learning time of ID3 is linear with the number of attributes (Shavlik, Mooney, & Towell 1991), it has been also noticed that C4.5 does not scale well when there are many attributes (Dietterich 1997).

The motivation of this paper is to develop a fast algorithm searching the unrestricted model space with a powerful heuristic that can be computed efficiently. Our work is inspired by naive Bayes, which is based on an unrealistic assumption: all attributes are independent given the class. Because of the assumption, it has a very low time complexity of $O(m \cdot n)$, and still performs surprisingly well (Domingos & Pazzani 1997). Interestingly, if we introduce a similar assumption in decision-tree learning, the widely used information-gain heuristic can be computed more efficiently, which leads to a more efficient tree-growing algorithm with the same asymptotic time complexity of $O(m \cdot n)$ with naive Bayes and one-level decision trees. That is the key idea of this paper.

A Fast Tree-Growing Algorithm

To simplify our discussion, we assume that all attributes are non-numeric, and each attribute then occurs exactly once on each path from leaf to root. We will specify how to cope with numeric attributes later. In the algorithm analysis in this paper, we assume that both the number of classes and the number of values of each attribute are much less than m and are then discarded. We also assume that all training data are loaded to the main memory.

Conditional Independence Assumption

In tree-growing, the heuristic plays a critical role in determining both classification performance and computational cost. Most modern decision-tree learning algorithms adopt a (im)purity-based heuristic, which essentially measures the purity of the resulting subsets after applying the splitting attribute to partition the training data. Information gain, defined as follows, is widely used as a standard heuristic.

$$IG(\mathbf{S}, X) = Entropy(\mathbf{S}) - \sum_{x} \frac{|\mathbf{S}_x|}{|\mathbf{S}|} Entropy(\mathbf{S}_x), \quad (1)$$

where $\mathbf{S}$ is a set of training instances, X is an attribute and x is its value, $\mathbf{S}_x$ is a subset of $\mathbf{S}$ consisting of the instances with $X = x$, and $Entropy(\mathbf{S})$ is defined as

$$Entropy(\mathbf{S}) = -\sum_{i=1}^{|C|} P_{\mathbf{S}}(c_i) log P_{\mathbf{S}}(c_i), \quad (2)$$

where $P_{\mathbf{S}}(c_i)$ is estimated by the percentage of instances belonging to c_i in $\mathbf{S}$, and $|C|$ is the number of classes. $Entropy(\mathbf{S}_x)$ is similar.

Note that tree-growing is a recursive process of partitioning the training data and $\mathbf{S}$ is the training data associated with the current node. Then, $P_{\mathbf{S}}(c_i)$ is actually $P(c_i|\mathbf{x_p})$ on the entire training data, where $\mathbf{X_p}$ is the set of attributes along the path from the current node to the root, called path attributes, and $\mathbf{x_p}$ is an assignment of values to the variables in $\mathbf{X_p}$. Similarly, $P_{\mathbf{S}_x}(c_i)$ is $P(c_i|\mathbf{x_p}, x)$ on the entire training data.

In the tree-growing process, each candidate attribute (the attributes not in $\mathbf{X_p}$) is examined using Equation 1, and the one with the highest information gain is selected as the splitting attribute. The most time-consuming part in this process is evaluating $P(c_i|\mathbf{x_p}, x)$ for computing $Entropy(\mathbf{S}_x)$. It must pass through each instance in $\mathbf{S}_x$, for each of which it iterates through each candidate attribute X. This results in a time complexity of $O(|\mathbf{S}| \cdot n)$. Note that the union of the subsets on each level of the tree is the entire training data of size m, and the time complexity for each level is thus $O(m \cdot n)$. Therefore, the standard decision-tree learning algorithm has a time complexity of $O(m \cdot n^2)$.

Our key observation is that we may not need to pass through $\mathbf{S}$ for each candidate attribute to estimate $P(c_i|\mathbf{x_p}, x)$. According to probability theory, we have

$$\begin{aligned} P(c_i|\mathbf{x_p}, x) &= \frac{P(c_i|\mathbf{x_p})P(x|\mathbf{x_p}, c_i)}{P(x|\mathbf{x_p})} \\ &= \frac{P(c_i|\mathbf{x_p})P(x|\mathbf{x_p}, c_i)}{\sum_{i=1}^{|C|} P(c_i|\mathbf{x_p})P(x|\mathbf{x_p}, c_i)}. \end{aligned} \quad (3)$$

Assume that each candidate attribute is independent of the path attribute assignment $\mathbf{x_p}$ given the class, i.e.,

$$P(X|\mathbf{x_p}, C) = P(X|C). \quad (4)$$

Then we have

$$P(c_i|\mathbf{x_p}, x) \approx \frac{P(c_i|\mathbf{x_p})P(x|c_i)}{\sum_{i=1}^{|C|} P(c_i|\mathbf{x_p})P(x|c_i)}. \quad (5)$$

The information gain obtained by Equation 5 and Equation 1 is called *independent information gain* (IIG) in this paper.

Note that in Equation 5, $P(x|c_i)$ is the percentage of instances with $X = x$ and $C = c_i$ on the entire training data that can be pre-computed and stored with a time complexity of $O(m \cdot n)$ before the tree-growing process with an additional space increase of $O(n)$, and $P(c_i|\mathbf{x_p})$ is the percentage of instances belonging to class c_i in $\mathbf{S}$ that can be computed by passing through $\mathbf{S}$ once taking $O(|\mathbf{S}|)$. Thus, at each level, the time complexity for computing $P(c_i|\mathbf{x_p}, x)$ using Equation 5 is $O(m)$, while C4.5 takes $O(m \cdot n)$ using Equation 3.

To compute IIG, $\frac{|\mathbf{S}_x|}{|\mathbf{S}|}$ in Equation 1 must also be computed. If we examine the partition for each candidate attribute X, the corresponding time complexity would be $O(m \cdot n)$. Fortunately, $\frac{|\mathbf{S}_x|}{|\mathbf{S}|} = P(x|\mathbf{x_p})$, which can be approximated by the denominator of Equation 5 taking $O(m)$. In C4.5, it takes $O(m \cdot n)$.

The time complexity for selecting the splitting attribute using IIG is similar to using information gain in C4.5. For the current node, $IIG(\mathbf{S}, X)$ must be calculated for each candidate attribute, which takes $O(n)$ for each node. The total time complexity for splitting attribute selection on the entire tree is then $O(k \cdot n)$, where k is the number of internal nodes on the tree. In the worst case, each leaf of the tree contains a single instance, and the number of internal nodes is $O(m)$. Thus, the total time complexity for splitting attribute selection is $O(m \cdot n)$. Note that C4.5 does the same thing and has the exact same time complexity for splitting attribute selection.

The total time for tree-growing is the sum of the time for probability estimation, partition, and splitting attribute selection. Therefore, the time complexity for tree-growing using IIG is $O(m \cdot n)$, as low as the time complexity for learning a naive Bayes and a one-level tree. Compared with the time complexity $O(m \cdot n^2)$ of C4.5 based on information gain, it is indeed a significant asymptotic improvement.

The conditional independence assumption in Equation 4 seems similar to the well-known conditional independence assumption in naive Bayes that assumes all attributes are independent given the class, defined as follows.

$$P(X_1, X_2, \cdots, X_n|C) = \prod_{i=1}^{n} P(X_i|C). \tag{6}$$

However, the assumption in Equation 4 is weaker than the assumption in Equation 6. First, Equation 6 defines an assumption on the entire instance space (global independence assumption). But Equation 4 defines an assumption on a subspace specified by a particular assignment $\mathbf{x_p}$. It is essentially context-specific independence (Friedman & Goldszmidt 1996). Apparently, context-specific independence has finer granularity than global independence in the sense that some attributes are not globally independent, but they are independent in certain context. Second, in Equation 4, we do not assume that candidate attributes not in $\mathbf{X_p}$ are independent. Naive Bayes, however, assumes that all attributes are independent, including candidate attributes.

It is well-known that the independence assumption for naive Bayes is unrealistic, but it still works surprisingly well even in the domains containing dependencies. Domingos and Pazanni (1997) ascribe that to the zero-one loss function in classification. That is, for a given instance e, as long as the class with the largest posterior probability $P(c|e)$ in naive Bayes remains the same with the true class, the classification of naive Bayes would be correct. Interestingly, their explanation is also suitable to decision-tree learning. In tree-growing, one attribute is selected at each step. Similarly, as long as the attribute with the largest IIG is the same with the one with the largest information gain, it would not affect attribute selection. So accurate information gain does not matter. To see how this could happen, let us consider the following example.

Given a *Boolean* concept $C = (A1 \wedge A2) \vee A3 \vee A4$, assume that the training data consist of all 16 instances. The decision tree generated by C4.5 is shown in Figure 1. It is interesting that the tree generated using IIG is the same. When choosing the root, IIG is information gain. In the other steps in tree-growing, IIG might not be equal to information gain. But the attribute with the highest information gain remains the same. For example, when choosing the attribute immediately under the root, the information gain values for A_1, A_2 and A_4 are 0.04, 0.04 and 0.54, and the IIG values for A_1, A_2 and A_4 are 0.02, 0.02 and 0.36. Apparently, A_4 is the attribute with both the highest information gain and IIG, although IIG is not equal to information gain. Note that the independence assumption for IIG is not true in this example.

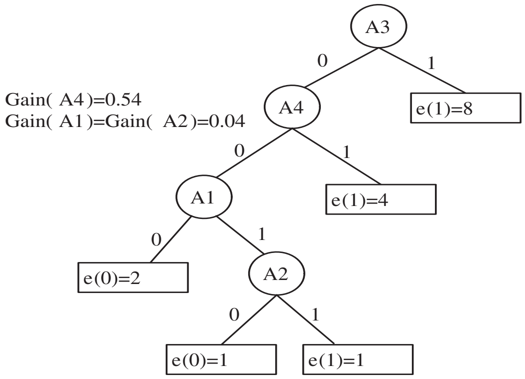

Figure 1: Decision tree generated for *Boolean* concept $(A1 \wedge A2) \vee A3 \vee A4$, in which $e(c)$ is the number of instances in class c falling into a leaf.

Another reason to support the independence assumption in Equation 4 is the accuracy of probability estimation. It is well-known that decision-tree learning suffers from the *fragmentation* problem: As the splitting process proceeds, the data associated with each descendant node becomes small. Eventually, when the depth of a tree is large, there is very little data with each leaf node and so the estimate for $P(c_i|\mathbf{x_p}, x)$ could be inaccurate. In computing IIG, since $P(x|c_i)$ is estimated on the entire training data and $P(c_i|\mathbf{x_p})$ is estimated before the partitioning using attribute X, this problem could be less serious.

Actually, the assumption in Equation 4 can be relaxed. To simplify our discussion, let us just consider binary classes $\{+, -\}$. Then, we have

$$\begin{aligned} P(+|\mathbf{x_p}, x) &= \frac{P(+|\mathbf{x_p})P(x|\mathbf{x_p}, +)}{P(+|\mathbf{x_p})P(x|\mathbf{x_p}, +) + P(-|\mathbf{x_p})P(x|\mathbf{x_p}, -)}, \\ &= \frac{P(+|\mathbf{x_p})}{P(+|\mathbf{x_p}) + P(-|\mathbf{x_p})\frac{P(x|\mathbf{x_p}, -)}{P(x|\mathbf{x_p}, +)}}. \end{aligned}$$

Obviously, the probability estimate generated from Equation 5 is equal to the one from Equation 3, if

$$\frac{P(x|\mathbf{x_p}, +)}{P(x|\mathbf{x_p}, -)} = \frac{P(x|+)}{P(x|-)}. \tag{7}$$

Naive Tree Algorithm

The tree-growing algorithm based on the conditional independence assumption, called *naive tree algorithm (NT)*, is illustrated as follows.

Algorithm NT(**Π**, **S**)

Input: **Π** is a set of candidate attributes, and **S** is a set of labeled instances.

```
Output: A decision tree $\mathbf{T}$.
1. If ($\mathbf{S}$ is pure or empty) or ($\mathbf{\Pi}$ is empty) Return $\mathbf{T}$.
2. Compute $P_{\mathbf{S}}(c_i)$ on $\mathbf{S}$ for each class $c_i$.
3. For each attribute $X$ in $\mathbf{\Pi}$, compute $IIG(\mathbf{S}, X)$ based on Equation 1 and 5.
4. Use the attribute $X_{max}$ with the highest $IIG$ for the root.
5. Partition $\mathbf{S}$ into disjoint subsets $\mathbf{S}_x$ using $X_{max}$.
6. For all values $x$ of $X_{max}$
   • $\mathbf{T}_x$=$NT$($\mathbf{\Pi} - X_{max}$, $\mathbf{S}_x$)
   • Add $\mathbf{T}_x$ as a child of $X_{max}$.
7. Return $\mathbf{T}$.
```

Before we call the NT algorithm, a set of probabilities $P(X|C)$ should be computed on the entire training data for each attribute and each class, which takes $O(m \cdot n)$. According to the analysis in the preceding section, the total time complexity of the NT algorithm is $O(m \cdot n)$. Note that the major difference between NT and C4.5 is in Step 3. C4.5 computes the information gain using $P(c_i|\mathbf{x_p}, x)$ for each candidate attribute X that needs to pass through each instance for each candidate attribute, taking $O(m \cdot n)$. In NT, $P(c_i|\mathbf{x_p}, x)$ is approximated using Equation 5 that takes only $O(m)$. Consequently, the time complexity of NT is as low as the time complexity of naive Bayes and one-level trees.

Note that in the NT algorithm above, we do not cope with numeric attributes for simplicity. In the implementation of the NT algorithm we process numeric attributes in the following way.

1. In preprocessing, all numeric attributes are discretized by k-bin discretization, where $k = \sqrt{m}$.
2. In selecting the splitting attribute, numeric attributes are treated the same as non-numeric attributes.
3. Once a numeric attribute is chosen, a splitting point is found using the same way as in C4.5. Then, the corresponding partitioning is done in Step 5, and a recursive call is made for each subset in Step 6. Note that a numeric attribute could be chosen again in the attribute selection on descendant nodes.

Note that we do not change the way to choose the splitting attribute in NT when there are numeric attributes, and that we use the same way as in C4.5 to choose a splitting point only after a numeric attribute has been selected. Thus the time complexity for selecting splitting attributes for NT keeps the same, and its advantage over C4.5 also remains the same. However, the time complexity of NT is higher than $O(m \cdot n)$. In fact, the time complexity of C4.5 is also higher than $O(m \cdot n^2)$ in the presence of numeric attributes.

In a traditional decision-tree learning algorithm, an attribute X is more likely to be selected if it is independent from the path attributes $\mathbf{X_p}$, because it provides more discriminative power than an attribute depending on $\mathbf{X_p}$. In other words, an attribute Y dependent on $\mathbf{X_p}$ would have less chance to be chosen than X. Using the independence assumption in Equation 4, such kind of attributes Y might have more chance to be chosen. In our tree-growing algorithm, we keep checking this kind of attributes. In our implementation, if an attribute leads to all subsets except one being empty, it will not be chosen as the splitting attribute and will be removed from the candidate attribute set. Intuitively, the tree sizes generated by NT tend to be larger than C4.5's. This issue can be overcome using post pruning. In our implementation, the pruning method in C4.5 is applied after a tree is grown. Interestingly, the tree sizes generated by NT are not larger than C4.5's after pruning according to the experimental results depicted in next section.

Experiments

Experiment Setup

We conducted experiments under the framework of Weka (version 3.4.7) (Witten & Frank 2000). All experiments were performed on a Pentium 4 with 2.8GHZ CPU and 1G RAM. We design two experiments. The first one is on 33 UCI benchmark data sets, which are selected by Weka and represent a wide range of domains and data characteristics. The data sets occur in a descending order of their sizes in Table 1. Missing values are processed using the mechanism in Weka, which replaces all missing values with the modes and means from the training data. The purpose of this experiment is to observe the general performance of NT and the influence of its independence assumption on accuracy in practical environments.

The second experiment is on 19 text classification benchmark data sets (Forman & Cohen 2004). These data sets are mainly from Reuters, TREC and OHSUMED, widely used in text classification research. In fact, decision-tree learning is one of the competitive text classification algorithms (Dumais *et al.* 1998), and its classification time is very low. From the preceding sections, we know that when the number of attributes is large, NT of $O(m \cdot n)$ should be significantly faster than C4.5 of $O(m \cdot n^2)$. Text classification usually involves thousands of attributes, from which we can observe the advantage of NT over C4.5 in running time (training time).

We compare NT with C4.5 and naive Bayes. Naive Bayes is well known as one of the most practical text classification algorithms, with very low time complexity and considerably good accuracy as well. We use the implementation of C4.5 and naive Bayes in Weka, and implement NT under the framework of Weka. The source code of NT is available at: *http://www.cs.unb.ca/profs/hzhang/naivetree.rar*.

In our experiments, we use three evaluation measures: accuracy, running time, and tree size. Certainly, we want the new algorithm to be competitive with C4.5 in accuracy with substantial decrease in running time. In decision-tree learning, a compact tree usually offers more insightful results than a larger tree, and tree size is also a widely-used evaluation criterion in decision trees learning.

We use the following abbreviations in the tables below.

$\mathbf{NT}$: the NT algorithm. $\mathbf{NT}(T)$ denotes its training time.

$\mathbf{C4.5}$: the traditional decision-tree learning algorithm (Quinlan 1993). $\mathbf{C4.5}(T)$ denotes its training time.

NB: naive Bayes. **NB**(T) denotes its training time.

R(S): the ratio of the tree size learned by NT to C4.5's. If the ratio is greater than 1, it means that NT grows a larger tree than does C4.5.

R(T): the ratio of the running time of NT to the running time of C4.5. If the ratio is greater than 1, it means that the running time of NT is longer than C4.5's.

Experimental Results on UCI Data Sets

Table 1 shows the accuracy, tree size ratio and training time ratio of each algorithm on each data set obtained via 10 runs of 10-fold stratified cross validation, and the average accuracy and standard deviation on all data sets are summarized at the bottom of the table. Note that we reported the running time ratio, instead of running time, because the running time on small data sets is very close to zero. Table 2 shows the results of the two-tailed t-test in accuracy with a 95% confidence interval, in which each entry $w/t/l$ means that the algorithm in the corresponding row wins in w data sets, ties in t data sets, and loses in l data sets, compared to the algorithm in the corresponding column.

Table 1: Experimental results on UCI data sets.

Data set	**NT**	**C4.5**	**NB**	**R**(S)	**R**(T)
Letter	85.66 ± 0.76	88.03 ± 0.71	64.07 ± 0.91	1.18	0.25
Mushroom	100 ± 0	100 ± 0	95.52 ± 0.78	1.10	1.02
Waveform	76.78 ± 1.74	75.25 ± 1.9	80.01 ± 1.45	1.13	0.23
Sick	98.54 ± 0.62	98.81 ± 0.56	92.89 ± 1.37	0.71	0.27
Hypothyroid	99.34 ± 0.41	99.58 ± 0.34	95.29 ± 0.74	0.81	0.76
Chess	97.34 ± 1	99.44 ± 0.37	87.79 ± 1.91	0.89	0.99
Splice	94.24 ± 1.23	94.08 ± 1.29	95.42 ± 1.14	0.81	0.31
Segment	95.34 ± 1.24	96.79 ± 1.29	80.17 ± 2.12	1.40	0.33
German-C	73.01 ± 3.93	71.13 ± 3.19	75.16 ± 3.48	0.42	0.36
Vowel	76.14 ± 4.17	80.08 ± 4.34	62.9 ± 4.38	1.15	0.45
Anneal	98.47 ± 1.22	98.57 ± 1.04	86.59 ± 3.31	1.11	0.16
Vehicle	69.36 ± 4.24	72.28 ± 4.32	44.68 ± 4.59	0.94	0.49
Diabetes	73.96 ± 4.62	74.49 ± 5.27	75.75 ± 5.32	0.77	0.41
Wisconsin-b	94.09 ± 2.77	94.89 ± 2.49	96.12 ± 2.16	0.93	0.44
Credit-A	85.17 ± 3.93	85.75 ± 4.01	77.81 ± 4.21	1.30	0.37
Soybean	91.9 ± 2.74	92.55 ± 2.61	92.2 ± 3.23	1.22	0.72
Balance	78.92 ± 4.04	77.82 ± 3.42	90.53 ± 1.67	0.94	0.62
Vote	95.17 ± 2.93	96.27 ± 2.79	90.21 ± 3.95	0.62	0.76
Horse-Colic	85.27 ± 5.04	83.28 ± 6.05	77.39 ± 6.34	0.47	0.47
Ionosphere	88.89 ± 4.74	89.74 ± 4.38	82.17 ± 6.14	1.00	0.25
Primary-t	38.76 ± 6.15	41.01 ± 6.59	47.2 ± 6.02	0.86	0.91
Heart-c	78.98 ± 7.31	76.64 ± 7.14	82.65 ± 7.38	0.88	0.56
Breast-Can	72.84 ± 5.49	75.26 ± 5.04	72.94 ± 7.71	2.30	0.95
Heart-s	82.89 ± 7.29	78.15 ± 7.42	83.59 ± 5.98	0.63	0.34
Audiology	75.67 ± 6.56	76.69 ± 7.68	71.4 ± 6.37	1.19	1.14
Glass	70.94 ± 9.51	67.63 ± 9.31	49.45 ± 9.5	1.02	0.47
Sonar	71.81 ± 10.74	73.61 ± 9.34	67.71 ± 8.66	1.35	0.24
Autos	74.02 ± 9.18	78.64 ± 9.94	57.67 ± 10.91	1.48	0.54
Hepatitis	80.16 ± 8.89	77.02 ± 10.03	83.29 ± 10.26	0.53	0.67
Iris	95.4 ± 4.51	94.73 ± 5.3	95.53 ± 5.02	1.03	1.07
Lymph	75.17 ± 9.57	75.84 ± 11.05	83.13 ± 8.89	0.85	0.70
Zoo	93.07 ± 6.96	92.61 ± 7.33	94.97 ± 5.86	1.20	0.63
Labor	83.77 ± 14.44	81.23 ± 13.68	93.87 ± 10.63	0.83	0.79
Mean	83.37 ± 4.79	83.57 ± 4.86	79.58 ± 4.92	1.00	0.56

Table 2: T-test summary in accuracy on UCI data sets.

	C4.5	**NB**
NT	2-27-4	15-11-7
C4.5		15-9-9

Table 1 and 2 show that the classification performance of NT is better than naive Bayes, and is very close to C4.5. We summarize the highlights briefly as follows:

1. NT performs reasonably well compared with C4.5. The average accuracy of NT is 83.37%, very close to C4.5's 83.57%, and the average tree size of NT is equal to C4.5's (the average ratio is 1.0).
2. On data sets "Letter", "Vowel" and "Chess", strong and high-order attribute dependencies have been observed (Kohavi 1996). We can see that naive Bayes performs severely worse than C4.5. However, the accuracy decrease of NT is only 2%-4%. This evidence supports our argument that the independence assumption in NT is weaker than naive Bayes', and shows that NT performs considerably well even when the independence assumption is seriously violated.
3. NT achieves higher accuracy and smaller tree size than C4.5 in data sets "German-C", "Heart-s" and "Labor". This means that C4.5 is not always superior to NT. We will see that NT even outperforms C4.5 in text classification in next section .
4. NT inherits the superiority of C4.5 to naive Bayes in accuracy on larger data sets. Since it has the same time complexity with naive Bayes, NT could be a more practical learning algorithm in large-scale applications.
5. NT is faster than C4.5. The average training time ratio is 0.56. But its advantage is not very clear. According to our discussion in the preceding section, NT would have a more clear advantage in running time when the data set is large and has a large number of attributes.

Experimental Results in Text Classification

We conducted the second experiment on text classification benchmark data sets. Table 3 shows the accuracy, training time in seconds and tree size ratio for each algorithm on each data set obtained via 10-fold stratified cross validation. Table 4 shows the results of the two-tailed t-test in accuracy with a 95% confidence interval. We summarize the highlights briefly as follows:

1. NT performs slightly better than C4.5 with 4 wins and 1 loss. The average accuracy of NT is 79.88%, higher than C4.5's 78%. On most data sets (14 out of 19), NT achieves higher accuracy . In addition, the tree size learned by NT is smaller than C4.5's on each data set, and the average tree size ratio is only 0.73. Note that, generally speaking, the conditional independence assumption is violated in text classification.
2. It is clear that NT is remarkably faster than C4.5. The average training time for NT is 93 seconds, thirty times less than C4.5's 2823 seconds.
3. It is not a surprise that NT performs better than naive Bayes, with the average accuracy 79.88% vs. the 75.63% of naive Bayes. The equally important point is that NT is as fast as naive Bayes in training. Note that decision-tree learning has a significant advantage over naive Bayes in terms of classification time.

In summary, our experimental results in text classification show:

Table 3: Experimental results on text classification data sets.

Data	$\mathbf{NT}$	$\mathbf{C4.5}$	$\mathbf{NB}$	$\mathbf{NT}(T)$	$\mathbf{NB}(T)$	$\mathbf{C4.5}(T)$	$\mathbf{R}(S)$
Ohscal	72.48	71.35	70.86	560	643	32694	0.42
La1	75.5	77.22	83.21	419	414	7799	0.55
La2	77.33	76.39	84.55	467	390	6820	0.58
Fbis	76.74	74.06	67.4	24	21	457	0.63
Cora36	39.39	46.94	61.78	106	25	1764	0.88
Re1	83.47	79	75.14	30	15	512	0.74
Re0	76.01	74.73	71.02	16	10	252	0.74
Wap	67.44	62.44	80.45	70	45	2180	0.73
Oh0	81.86	82.56	87.34	7	7	129	0.69
Oh10	78.19	73.62	80.19	8	8	161	0.54
Oh15	72.19	73.7	80.81	7	6	138	0.80
Oh5	84.3	81.58	82.45	5	6	88	0.68
Tr31	94.07	92.77	88.45	12	28	188	0.96
Tr11	86.95	77.58	67.63	5	7	73	0.74
Tr12	81.44	83.05	72.85	3	4	33	0.85
Tr21	88.68	82.44	52.34	4	7	81	0.54
Tr23	97.57	91.12	59.29	1	3	12	0.98
Tr41	93.73	91.35	91.81	8	18	153	0.89
Tr45	90.29	90.14	79.42	10	16	101	0.98
Mean	79.88	78.00	75.63	93	88	2823	0.73

1. NT scales well in the problems with large number of attributes.
2. NT could be a practical text classification algorithm, because it is as fast as naive Bayes in training and significantly faster in testing, and still enjoys higher classification accuracy.

Conclusion

In this paper, we present a novel, efficient algorithm for decision-tree learning. Its time complexity is $O(m \cdot n)$, as low as learning a naive Bayes and a one-level tree. It is also a significant asymptotic reduction on the time complexity $O(m \cdot n^2)$ of C4.5. Our new algorithm is based on a conditional independence assumption, similar to but weaker than the conditional independence assumption of naive Bayes. Experiments show that the new algorithm scales up well to large data sets with large number of attributes: It performs significantly faster than C4.5 while maintaining competitive accuracy. One drawback of the new algorithm is its manner to handle missing values, which is clearly inferior to C4.5's and will be a topic for our future research.

Although we focus on the information gain-based decision-tree learning in this paper, estimating the purity of subsets generated by a splitting attribute X, i.e., computing $P_{\mathbf{S}_x}(c_i)$, is essential in many other decision-tree learning algorithms. So the method proposed in this paper is also suitable to those algorithms. Note also that the NT algorithm is a core tree-growing algorithm, which can be combined with other scaling-up techniques to achieve further speedup. In addition, NT could be a practical algorithm used in various applications with large amount of data, such as text classification.

Table 4: T-test summary in accuracy on text classification data sets.

	$\mathbf{C4.5}$	$\mathbf{NB}$
$\mathbf{NT}$	4-14-1	9-4-6
$\mathbf{C4.5}$		7-6-6

References

Auer, P.; Holte, R. C.; and Maass, W. 1995. Theory and applications of agnostic PAC-learning with small decision trees. In *Proceedings of the 12th International Conference on Machine Learning*. Morgan Kaufmann. 21–29.

Dietterich, T. G. 1997. Machine learning research: Four current directions. *AI Magazine* 18:4:97–136.

Domingos, P., and Pazzani, M. 1997. Beyond independence: Conditions for the optimality of the simple Bayesian classifier. *Machine Learning* 29:103–130.

Dumais, S.; Platt, J.; Heckerman, D.; and Sahami, M. 1998. Inductive learning algorithms and representations for text categorization. In *Proceedings of the Seventh International Conference on Information and Knowledge Management*. 148–155.

Forman, G., and Cohen, I. 2004. Learning from little: Comparison of classifiers given little training. In *Proceeding of PKDD2004*. 161–172.

Friedman, N., and Goldszmidt, M. 1996. Learning Bayesian networks with local structure. In *Proceedings of the Twelfth Conference on Uncertainty in Artificial Intelligence*. 252–262.

Gehrke, J. E.; Ramakrishnan, R.; and Ganti, V. 2000. Rainforest - A framework for fast decision tree construction of large datasets. *Data Mining and Knowledge Discovery* 4:2/3:127–162.

Holte, R. 1993. Very simple classification rules perform well on most commonly used datasets. *Machine Learning* 11:63–91.

Kohavi, R. 1996. 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*. AAAI Press. 202–207.

Mehta, M.; Agrawal, R.; and Rissanen, J. 1996. SLIQ: A fast scalable classifier for data mining. In *Proceedings of the Fifth International Conference on Extending Database Technology*. 18–32.

Perlich, C.; Provost, F.; and Simonoff, J. S. 2003. Tree induction vs. logistic regression: A learning-curve analysis. *Machine Learning Research* 4:211–255.

Provost, F. J., and Kolluri, V. 1999. A survey of methods for scaling up inductive algorithms. *Data Min. Knowl. Discov* 3(2):131–169.

Quinlan, J. 1993. *C4.5: Programs for Machine Learning*. Morgan Kaufmann: San Mateo, CA.

Shafer, J. C.; Agrawal, R.; and Mehta, M. 1996. SPRINT: A scalable parallel classifier for data mining. In *Proceedings of the 22nd International Conference on Very Large Databases*. Morgan Kaufmann. 544–555.

Shavlik, J.; Mooney, R.; and Towell, G. 1991. Symbolic and neural network learning algorithms: An experimental comparison. *Machine Learning* 6:111–143.

Witten, I. H., and Frank, E. 2000. *Data Mining –Practical Machine Learning Tools and Techniques with Java Implementation*. Morgan Kaufmann.