Representing Conditional Independence Using Decision Trees

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

Abstract

While the representation of decision trees is fully expressive theoretically, it has been observed that traditional decision trees has the replication problem. This problem makes decision trees to be large and learnable only when sufficient training data are available. In this paper, we present a new representation model, *conditional independence trees* (CITrees), to tackle the replication problem from probability perspective. We propose a novel algorithm for learning CITrees. Our experiments show that CITrees outperform naive Bayes (Langley, Iba, & Thomas 1992), C4.5 (Quinlan 1993), TAN (Friedman, Geiger, & Goldszmidt 1997), and AODE (Webb, Boughton, & Wang 2005) significantly in classification accuracy.

Introduction

In decision tree learning algorithms, a decision tree is induced from a set of labeled training examples represented by a set of attribute values and a class label. We denote a set of attributes by a bold-face upper-case letter, for example, $\mathbf{A} = (A_1, A_2, \cdots, A_n)$, and an assignment of values to each attribute in an attribute set by a corresponding bold-face lower-case letter, for example, $\mathbf{a}$. 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.

While decision trees perform quite well in classification, the representation structure of traditional decision trees suffers from the replication problem (Oliver 1993; Pagallo & Haussler 1990). Namely, we are forced to represent disjunctive concepts in different branches, which leads to duplication of subtrees. This problem causes decision trees to be large and learnable only when sufficient training data are available. Let us see the following example.

Example 1: Assume that the target function is a Boolean concept $C = (A1 \wedge A2) \vee (A3 \wedge A4)$. The decision tree for C is shown in Figure 1. You can see the duplicate subtrees of $(A3 \wedge A4)$.

Another related problem is 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 (Oliver 1993; Pagallo & Haussler 1990) and so the prediction could be inaccurate.

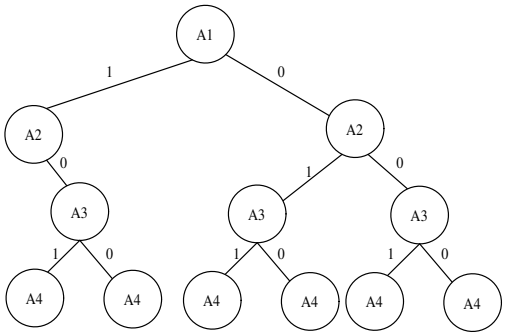

Figure 1: A decision tree without leaf nodes for $(A1 \wedge A2) \vee (A3 \wedge A4)$.

Thus, a more compact structure of decision trees is desirable. Intuitively, decomposing a large decision tree into small trees is a natural approach to solving the two problems. But under the paradigm of traditional decision trees, it is difficult to combine the predictions from multiple trees.

Although decision trees are not generally treated as a probabilistic model, they can be used to represent probability distributions, called probabilistic trees (Buntine 1991). In a probabilistic tree, a leaf L represents the conditional probability $P(\mathbf{x_p}(L)|c)$ [1], where $\mathbf{X_p}(L)$ are the attributes that occur on the path from the root to L, called the path attributes of L. Thus, a probabilistic tree represents a conditional probability distribution given the class. Assume that the attribute set $\mathbf{A}$ can be partitioned into disjoint subsets of attributes and the attributes in one subset are conditional independent of all the attributes in other subsets. We can use a probabilistic tree to represent the conditional distribution for each subset and, roughly speaking, the full conditional distribu-

[1] In a traditional probabilistic tree (Buntine 1991), a leaf L represents $P(c|\mathbf{x_p}(L))$, instead of $P(\mathbf{x_p}(L)|c)$ used in this paper. Since $P(c|\mathbf{x_p}(L))$ is estimated by using the fraction of examples of class C in the training data associated with L, $P(c)$ is estimated from the entire training data, and $P(\mathbf{x_p}(L))$ is irrelevant to C, $P(\mathbf{x_p}(L)|c)$ can be easily computed from $P(c|\mathbf{x_p}(L))$ and $P(c)$.

tion $P(\mathbf{A}|C)$ is equal to the product of conditional probability distributions of all trees. Notice that $P(\mathbf{A}|C)$ and the prior probability $P(C)$ determine a unique classifier.

It is easy to decompose the decision tree for **Example 1** into two probabilistic trees corresponding to two concepts $(A1 \wedge A2)$ and $(A3 \wedge A4)$, respectively, shown in Figure 2. Notice that all the probabilities are estimated from the truth table, in which each possible assignment of truth values to A_1, A_2, A_3, and A_4 occurs exactly once. Figure 2 represents the conditional independence that $P(A_1, A_2, A_3, A_4|C) = P(A_1, A_2|C)P(A_3, A_4|C)$ (Strictly, that is not true.). It is easy to verify that Figure 2 represents the target concept $C = (A1 \wedge A2) \vee (A3 \wedge A4)$.

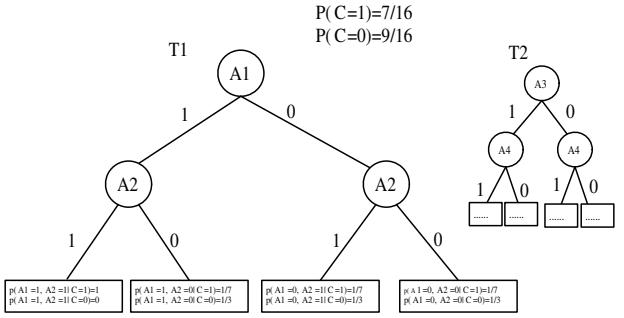

Figure 2: Two probabilistic trees representing $(A1 \wedge A2) \vee (A3 \wedge A4)$, in which the probabilities in T_2 are similar to that in T_1.

Figure 2 shows us that a compact decision tree presentation can be achieved by representing conditional independence in probabilistic trees. This is the key idea of this paper.

Related Work

Although decision trees performs well in classification, their replication problem and fragmentation problem are also well known (Oliver 1993; Pagallo & Haussler 1990). Those two problems have been attacked from two major approaches: constructing compound attributes (Oliver 1993; Pagallo & Haussler 1990), and extending the tree structure of decision trees to the more complex graph structure, such as decision graphs (Kohavi 1994). However, no clear solution has emerged under the traditional decision tree paradigm.

It has also been observed that traditional decision tree algorithms, such as C4.5 (Quinlan 1993), produce poor probability estimates(Provost, Fawcett, & Kohavi 1998). A substantial amount of work has been done recently on learning decision trees with accurate probability estimates (Provost & Domingos 2003). Although, theoretically, decision trees can represent any probability distribution, those that yield accurate probabilities tend to be large because of the replication problem, and the probability estimates could still be poor because of the fragmentation problem. A more compact representation could be an effective solution for this problem.

Although decision trees are well-known as a decision boundary-based classifier, each leaf of a tree can represent a conditional probability distribution. One way to extend decision trees toward a probabilistic model is to deploy a local probability model on leaves of a decision tree (Smyth, Gray, & Fayyad 1996). Friedman and Goldszmidt (1996) propose to use decision trees to represent the local distributions in Bayesian networks, in which only the conditional probability distribution of a single attribute is represented. Recently, Zhang and Su (2004) propose to use a decision tree to represent conditional independence. They present a type of probabilistic trees, in which, given the path attributes, all other attributes on a leaf are independent. Their model, however, is not a general model that can represent any conditional independence. Jaeger (2004) propose a model *probabilistic decision graphs* that is based on ordered binary decision diagrams.

In this paper, we present a novel decision tree representation model, *conditional independence trees* (CITrees). Our basic idea is to iteratively explore and represent conditional attribute independencies at each step in constructing a decision tree, and thus decompose a traditional decision (sub)tree into smaller (sub)trees.

Representing Conditional Independence under Decision Tree Paradigm

The key feature of decision trees is the iterative and nested decomposition. The instance space is iteratively partitioned into subspaces in a decision tree. More precisely, each internal node N corresponds to a subspace defined by the values of the path attributes $\mathbf{X_p}(N)$. It is natural to explore the conditional independencies among attributes in the subspace. Here the conditioning variables are $\mathbf{X_p}(N)$ and C. Indeed, some attribute independencies do not exist in the entire instance space, but do exist in some subspaces. This type of conditional attribute independencies are called *context-specific independence* (Boutilier *et al.* 1996), differentiating it from the *global conditional independence* that is conditioned only by C. The structure of decision trees from top to bottom provides a natural structure for exploring and representing various context-specific independence among attributes in various granularities from coarse to fine.

As discussed in the first section, a leaf L in a probabilistic tree represents the conditional probability $P(\mathbf{x_p}(L)|c)$. Let us denote all the attributes not in $\mathbf{X_p}(L)$ by $\mathbf{X_l}(L)$, called non-path attributes. If there is a representation of the conditional probability distribution over the non-path attributes at each leaf L, denoted by $P(\mathbf{x_l}(L)|\mathbf{x_p}(L), c)$, then each leaf represents a full conditional probability over all the attributes, conditioned by C, as shown in Equation 1. Thus, a probabilistic tree represents a full conditional distribution $P(\mathbf{A}|C)$.

$$P(\mathbf{a}|c) = P(\mathbf{x_l}(L)|\mathbf{x_p}(L), c)P(\mathbf{x_p}(L)|c). \quad (1)$$

Moreover, the subtree with root N represents a conditional distribution $P(\mathbf{X_l}(N)|\mathbf{x_p}(N), c)$. In the recursive process of learning a decision tree, if $\mathbf{X_l}(N)$ can be partitioned into disjoint and conditional independent subsets $\mathbf{A_{l_1}}(N)$, $\cdots$, $\mathbf{A_{l_k}}(N)$, we can construct k probabilistic (sub)trees, each of which represents $P(\mathbf{A_{l_i}}|\mathbf{x_p}(N), c)$, $i = 1, \cdots, k$, and their product represents $P(\mathbf{X_l}(N)|\mathbf{x_p}(N), c)$. Thus, a traditional

probabilistic (sub)tree can be represented by a set of smaller trees.

Definition 1 *Given two attributes A_i and A_j, and a set of attributes Θ_{ij}, A_i and A_j are said to be conditionally independent given Θ_{ij}, if*

$$P(A_i, A_j|\Theta_{ij}, C) = P(A_i|\Theta_{ij}, C)P(A_j|\Theta_{ij}, C). \quad (2)$$

Definition 2 *Given two sets of attributes U and Θ, and $\Theta \cap U = \Phi$ (the empty set), a subset U_I is called a conditional independence set given Θ, or simply CI set, if*

1. *for any attribute $A_j \notin U_I$, A_j is conditionally independent from any attribute $A_i \in U_I$ given Θ;*
2. *U_I is the minimum subset satisfying property 1.*

Assuming that $\{U_{I_1}, \cdots, U_{I_k}\}$ is a partition of CI sets of U given Θ, we have

$$P(U|\Theta, C) = \prod_{i=1}^{k} P(U_{I_i}|\Theta, C). \quad (3)$$

Our idea is that, at each step of constructing a probabilistic tree, we detect conditional attribute dependencies by discovering CI sets. We then build a tree for each CI set by choosing one attribute as the root, and repeat this process for each of its branches until certain criteria have been met. We use a rectangle to contain a set of trees, each of which corresponds to a CI set. A *conditional independent tree*, or simply CITree, is defined formally as follows.

Definition 3 *1. A probabilistic tree (without any rectangle) is a CITree.*

2. *A rectangle containing a set of probabilistic trees is a CITree.*
3. *If $T_1, \cdots, T_k$ are CITrees, then a tree consisting of a root and subtrees $T_1, \cdots, T_k$ is a CITree.*

Example 2: Figure 3 shows a CITree, in which the rectangle containing two subtrees T_2 and T_3 represents the following context-specific independence.

$$P(A_2, A_3|A_1 = 1, c) = P(A_2|A_1 = 1, c)P(A_3|A_1 = 1, c).$$

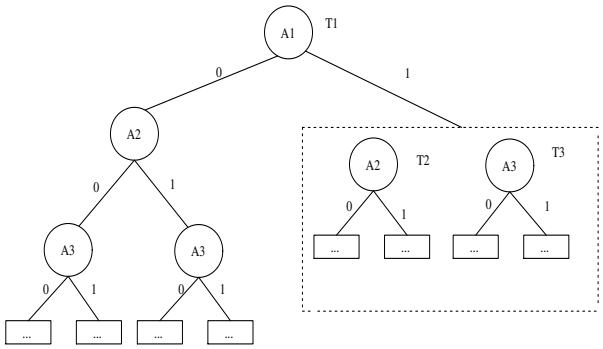

Figure 3: CITree for **Example 2**.

The conditional probability distribution represented by a CITree T is defined formally as follows.

Definition 4 *Assume that T is a (sub)CITree on attribute set $\mathbf{A} = (A_1, A_2, \cdots, A_n)$, the conditional distribution $P(T)$ represented by T is defined as follows.*

1. *If T is a probabilistic tree with root A (a CITree without rectangle), its conditional probability $P(T) = P(\mathbf{A}|\mathbf{X_p}(A), C)$.*
2. *If T is represented by a rectangle containing independent sub-CITrees $T_1, \cdots, T_k$,*

$$P(T) = \prod_{i=1}^{k} P(T_i). \quad (4)$$

3. *If T is a single (sub)CITree with root $A = \{a_1, \cdots, a_k\}$ and subtrees $T_s(a_i)$ corresponding to $A = a_i$, $i = 1, \cdots, k$,*

$$P(T) = P(T_s(a_i)), \textit{when } A = a_i. \quad (5)$$

A CITree provides, essentially, a more compact representation for the full conditional distribution, given C, than a probabilistic tree. Combined with $P(C)$, it also determines a classifier.

Bayesian networks (Pearl 1988) are a well-known probabilistic model. Generally speaking, a Bayesian network represents attribute independencies in the entire instance space, that is, global independence. It is not natural to use Bayesian networks to represent various context-specific independencies in different granularities. For example, assume that A_{k+2} depends on A_{k+1} only when $A_i = a_i$, $i = 1, \cdots, k$. To represent this fact in a Bayesian network, A_{k+2} should have $k + 1$ parents. In a CITree, this dependency is represented by only one path.

Another drawback of Bayesian networks is that it is intractable to learn the optimal structure of a Bayesian network from data. Thus, in practice, imposing restrictions on the structures of Bayesian networks, such as tree-augmented naive Bayes (TAN) (Friedman, Geiger, & Goldszmidt 1997), in which each attribute is allowed to have a parent only from other attributes, has been accepted. In contrast, CITrees have a tree structure, substantially simpler than a graph structure. It could be expected that learning a good CITree is easier than learning a good Bayesian network.

Notice the difference between CITrees and the work of Boutilier *et al.* (1996). In their work, context-specific independence is explored within the framework of Bayesian networks, not iteratively from various granularities as CITrees do. Moreover, a decision tree is used only to represent the local distribution of a single attribute.

Learning Conditional Independence Trees

To learn a CITree from data, the first issue is to explore conditional attribute independencies. More concretely, it is how to partition an attribute set into CI sets. We use conditional mutual information, defined in Equation 6, to detect the dependency between two attributes.

$$I_P(X; Y|Z) = \sum_{x,y,z} P(x, y, z) log \frac{P(x, y|z)}{P(x|z)P(y|z)}, \quad (6)$$

where x, y, and z are the values of variables X, Y, and Z respectively.

In practice, we focus on strong attribute dependencies, while ignoring weak attribute dependencies. We define a threshold based on the Minimum Description Length (MDL) principle to filter out weak dependencies, defined in Equation 7.

$$\delta(A_i, A_j) = \frac{log|D|}{2|D|} \times |T_{ij}|, \tag{7}$$

where $|D|$ is the size of the local training data, and $|T_{ij}| = |A_i| \times |A_j|$. Roughly speaking, $|T_{ij}|$ represents the size increase for representing the dependency between A_i and A_j in a CITree. In our implementation, A_i and A_j are put into a CI set, if and only if $I(A_i; A_j|C) > \delta(A_i, A_j)$. $\delta(A_i, A_j)$ is essentially a threshold. In **Example 1**, A_1, A_2, A_3 and A_4 can be grouped into two CI sets $\{A_1, A_2\}$ and $\{A_3, A_4\}$ using an appropriate threshold. Thus, Figure 2 is learnable.

After exploring the conditional attribute independencies in the local training data, we get CI set(s). For each CI set U_I, a (sub)CITree is built. We first introduce a few definitions that are used in choosing the root attribute in our algorithm.

Definition 5 *Given a CI set U_I and an attribute $A_i \in U_I$, the most influent attribute of A_i, denoted by A_i^{inf}, is defined as follows.*

$$A_i^{inf} = \arg \max_{A_j \in U_I, j \neq i} I(A_i, A_j|C). \tag{8}$$

Definition 6 *In a CI set U_I , an attribute A_i is called a composite attribute, if it satisfies the following equation.*

$$I(C; A_i|A_i^{inf}) > I(C; A_i) + \delta(A_i, A\_i^{inf}). \tag{9}$$

Intuitively, a composite attribute is an attribute that is more useful in discriminating the class variable combined with other attribute.

Definition 7 *The discriminating score of attribute A_i, denoted by $\psi(A_i)$, is defined as follows:*

$$\psi(A_i) = \max\{I(C; A_i), I(C; A_i|A_i^{inf})\}, \tag{10}$$

where $I(C, A_i)$ is the mutual information between C and A_i, defined in Equation 11.

$$I(C; A_i) = \sum_{C, A_i} P(C, A_i) log \frac{P(C, A_i)}{P(C)P(A_i)}. \tag{11}$$

In Definition 7, we take the combination of two attributes into account. In reality, it often happens that the combination of attributes is more useful in discriminating the class variable C.

Definition 8 *The most discriminating attribute for the class variable C, denoted by A_{dis}, is defined as follows.*

$$A_{dis} = \arg \max_{A_i} \psi(A_i). \tag{12}$$

Definition 9 *Given an attribute A_i, the influenced set of A_i, denoted by $\Psi(A_i)$, consists of all the attributes that are composite attributes with A_i as the most influent attribute.*

Definition 10 *The influence score of attribute A_i, denoted by $\omega(A_i)$, is defined as follows.*

$$\omega(A_i) = \sum_{A_j \in \Psi(A_i)} (I(C; A_j|A_i) - I(C; A_j) - \delta(A_i, A_j)). \tag{13}$$

Definition 10 reflects the influence of attribute A_i on other attributes in forming composite attributes.

Now we are ready to talk about choosing the root attribute. One straightforward way is to consider each attribute individually and choose the attribute based on Equation 11. In fact, C4.5 adopts this strategy. A more sophisticated strategy is to consider the combination of attributes. Our strategy is to consider the most discriminating attribute and its most influent attribute. An interesting observation from our experiments is that choosing which one first from the most discriminating attribute and its most influent attribute is crucial, if the most discriminating attribute is composite. Thus, we define the influence score for an attribute to distinguish them. The process for choosing the root attribute consists of two steps: computing the most discriminating attribute A_{dis} based on the discriminating scores of attributes, and then choose one from A_{dis} and its most influent attribute A_{dis}^{inf} based on their influence scores. Notice that we use a threshold in Equation 9 and 13 to handle the overfitting issue.

Building a CItree is also a greedy and recursive process, similar to building a decision tree. At each step, after discovering CI sets, choose the "best" attribute for each CI set as the root of the (sub)tree, split the local training data into disjoint subsets corresponding to the values of the root attribute, and then recur this process for each subset until all attributes have been used.

Algorithm CITrees(**S**, **U**)

Input : a set **S** of labeled examples, and a set **U** of attributes.

Output : a CITree.

1. **If U** is empty **Return** an empty tree.
2. Identify a partition $\{U_{I_1}, \cdots, U_{I_k}\}$ of CI sets of **U**.
3. **For** each CI set U_{I_i}
4. Create an empty tree T_i.
5. Choose the attribute A_{dis} using Equation 12.
6. **If** A_i is composite and $\omega(A_{dis}^{inf}) > \omega(A_{dis})$
7. **Then** $A_{root} = A_{dis}^{inf}$
8. **else**$A_{root} = A_{dis}$
9. Make A_{root} the root of tree T_i.
10. **For** all values a of A_{root}
11. T_a = CITrees($\mathbf{S}_a$, $\mathbf{A} - \{A_{root}\}$).
12. Add T_a as a child of A_{root}.
13. **If** $(k = 1)$ **Return** T_1.
14. **Else Return** a rectangle containing $T_1, \cdots, T_k$.

Experiments

We conducted experiments to compare our algorithm CITrees with C4.5 (Quinlan 1993), naive Bayes (Langley, Iba, & Thomas 1992), TAN (Friedman, Geiger, & Goldszmidt 1997), and AODE (averaged one-dependence estimators) (Webb, Boughton, & Wang 2005). AODE is a newly developed model which demonstrates a remarkable performance. We implemented CITrees within the Weka framework (Witten & Frank 2000), and used the implementations of naive Bayes, C4.5(J48), TAN, and AODE in Weka. We chose the 33 UCI data sets from Weka. In our experiment, the accuracy of an algorithm on each data set has been obtained via 10 runs of 10-fold stratified cross validation. Numeric attributes are discretized using ten-bin discretization implemented in Weka. Missing values are also processed using the mechanism in Weka. In our implementation, we used the Laplace estimation to avoid the zero-frequency problem. We conducted a two-tailed t-test with a 95% confidence level to compare each pair of algorithms on each data set.

Table 1 shows the accuracies of the algorithms on each data set, and the average accuracy and standard deviation on all data sets are summarized at the bottom of the table. Table 2 shows the results of the two-tailed t-test, 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. The detailed results displayed in Table 1 and Table 2 show that the performance of CITrees is overall the best among the algorithms compared in this paper. Now, we summarize the highlights briefly as follows:

1. CITrees perform better than C4.5 (7 wins and 1 loss), TAN (7 wins and 1 loss), and AODE (9 wins and 3 losses).
2. CITrees significantly outperform naive Bayes (12 wins and 0 loss).
3. CITrees achieve the highest average accuracy among all algorithms.

Figure 4 shows an example of a CITree learned from the "Vowel" data set. In this example, the CITree algorithm generated a CITree with the "SpeakerNumbers" attribute as the root, and all other attributes are conditionally independent given the value of "SpeakerNumbers". This CITree has only 151 nodes, significantly smaller than the tree of 600 nodes generated by C4.5. More important, the accuracy of CITree is 94.35%, significantly higher than C4.5's 75.57%.

Table 2: Summary of the experimental results.

	NB	AODE	TAN	C4.5
CITree	12-21-0	9-21-3	7-25-1	7-25-1
NB		1-18-14	3-19-11	8-11-14
AODE			5-24-4	11-18-4
TAN				10-19-4

Conclusions

In this paper, we present a novel decision tree representation model CITrees. CITrees provide a compact representation by iteratively exploring and representing context-specific attribute independencies from various granularities. Roughly speaking, a traditional probabilistic (sub)tree can be decomposed into smaller CITrees. CITrees can be viewed as a general probabilistic representation model, just as Bayesian networks. However, CITrees can efficiently represent context-free independence from various granularities, whereas Bayesian networks can efficiently represent global independence. We proposed an algorithm for learning CITrees and conducted experiments to compare it with other state-of-the-art algorithms.

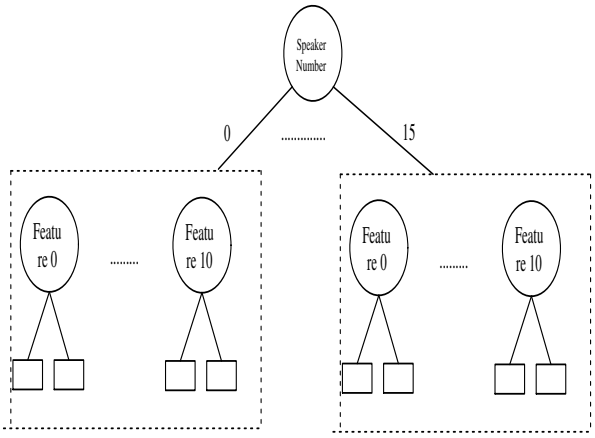

Figure 4: The CITree induced from the "Vowel" data set.

Since the structure of CITrees is significantly simpler than the structure of Bayesian networks, we believe that the CITree learning algorithm could be considerably simpler than Bayesian network learning algorithms. There is thus considerable potential room for improving the CITree learning algorithm presented in this paper. Another interesting question is: Can CITrees be used as a general inference model just as Bayesian networks are? This paper only addresses the classification problem.

References

Boutilier, C.; Friedman, N.; Goldszmidt, M.; and Koller, D. 1996. Context-specific independence in Bayesian networks. In *Proceedings of the 12th Annual Conference on Uncertainty in AI (UAI)*, 416–422.

Buntine, W. 1991. Theory refinement on Bayesian networks. In *Proceedings of the Seventh Conference on Uncertainty in Artificial Intelligence*. Morgan Kaufmann. 52–60.

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

Friedman, N.; Geiger, D.; and Goldszmidt, M. 1997. Bayesian network classifiers. *Machine Learning* 29:131–163.

Jaeger, M. 2004. Probabilistic decision graphs - combining verification and AI techniques for probabilistic inference.

Table 1: Experimental results on accuracy.

Dataset	NB	C4.5	TAN	AODE	CITree
Anneal	94.32 ± 2.23	98.65 ± 0.97	98.34 ± 1.18	96.83 ± 1.66	99.29 ± 0.87
Audiology	71.4 ± 6.37	77.22 ± 7.69	72.68 ± 7.02	71.66 ± 6.42	79.18 ± 7.6
Autos	63.67 ± 11.36	80.44 ± 8.46	76.98 ± 9.21	75.09 ± 10.23	80.84 ± 8.4
Balance	91.44 ± 1.3	64.14 ± 4.16	86.22 ± 2.82	89.78 ± 1.88	91.44 ± 1.29
Breast-cancer	72.94 ± 7.71	75.26 ± 5.04	70.09 ± 7.68	72.73 ± 7.01	70.91 ± 7.54
Wisconsin-breast	97.3 ± 1.75	92.19 ± 3.16	95.05 ± 2.24	96.91 ± 1.84	97.2 ± 1.83
Horse-colic	79.65 ± 5.9	84.77 ± 5.89	80.55 ± 6.23	81.26 ± 5.83	80.71 ± 5.82
Credit-rating	84.75 ± 3.83	85.32 ± 4.42	84.22 ± 4.41	85.78 ± 3.75	84.81 ± 4.03
German-credit	75.93 ± 3.87	72.61 ± 3.49	75.86 ± 3.58	76.45 ± 3.88	74.58 ± 3.62
Pima-diabetes	75.68 ± 4.85	73.89 ± 4.7	75.09 ± 4.96	76.57 ± 4.53	75.52 ± 4.71
Glass	57.69 ± 10.07	58.14 ± 8.48	58.43 ± 8.86	61.73 ± 9.69	60.43 ± 8.94
Cleveland	83.58 ± 6.4	79.44 ± 6.43	82.78 ± 6.98	83.07 ± 7.05	82.38 ± 6.86
Heart-statlog	83.78 ± 5.41	79.78 ± 7.71	79.37 ± 6.87	83.63 ± 5.32	81.96 ± 5.73
Hepatitis	83.53 ± 10.47	81.5 ± 8.24	82.13 ± 9.17	83.55 ± 9.73	84.07 ± 9.09
Hypothyroid	92.79 ± 0.73	93.24 ± 0.44	93.23 ± 0.68	93.56 ± 0.61	93 ± 0.69
Ionosphere	90.86 ± 4.33	87.47 ± 5.17	92.23 ± 4.36	91.74 ± 4.28	91.97 ± 4.26
Iris	94.33 ± 6.79	96 ± 4.64	91.67 ± 7.18	94 ± 5.88	94.2 ± 6.47
Chess	87.79 ± 1.91	99.44 ± 0.37	92.05 ± 1.49	91.03 ± 1.66	99.03 ± 0.52
Labor	95.67 ± 8.28	84.97 ± 14.24	87.67 ± 13.77	94.73 ± 8.79	93.4 ± 10.57
Letter	70.09 ± 0.93	81.31 ± 0.78	83.11 ± 0.75	85.54 ± 0.68	87.22 ± 0.72
Lymphography	85.97 ± 8.88	78.21 ± 9.74	84.07 ± 8.93	85.46 ± 9.32	85.7 ± 8.02
Mushroom	95.52 ± 0.78	100 ± 0	99.99 ± 0.03	99.95 ± 0.07	100 ± 0
Primary-tumor	47.2 ± 6.02	41.01 ± 6.59	46.76 ± 5.92	47.87 ± 6.37	43.78 ± 6.59
Segment	89.03 ± 1.66	93.42 ± 1.67	94.54 ± 1.6	92.92 ± 1.4	94.52 ± 1.54
Sick	96.78 ± 0.91	98.16 ± 0.68	97.61 ± 0.73	97.52 ± 0.72	97.82 ± 0.83
Sonar	76.35 ± 9.94	71.09 ± 8.4	73.66 ± 10.04	79.91 ± 9.6	77.99 ± 9.96
Soybean	92.2 ± 3.23	92.63 ± 2.72	95.24 ± 2.28	93.31 ± 2.85	94.08 ± 2.78
Splice	95.42 ± 1.14	94.17 ± 1.28	95.39 ± 1.16	96.12 ± 1	94.56 ± 1.46
Vehicle	61.03 ± 3.48	70.74 ± 3.62	73.71 ± 3.48	71.65 ± 3.59	72.38 ± 3.76
Vote	90.21 ± 3.95	96.27 ± 2.79	94.57 ± 3.23	94.52 ± 3.19	95.52 ± 2.89
Vowel	66.09 ± 4.78	75.57 ± 4.58	93.1 ± 2.85	89.64 ± 3.06	94.35 ± 2.5
Waveform	79.97 ± 1.46	72.64 ± 1.81	80.72 ± 1.78	84.24 ± 1.6	81.43 ± 1.65
Zoo	93.98 ± 7.14	92.61 ± 7.33	93.69 ± 7.75	94.66 ± 6.38	95.75 ± 6
Mean	82.33 ± 4.78	82.49 ± 4.71	84.26 ± 4.82	85.25 ± 4.54	85.75 ± 4.47

Int. J. of Uncertainty, Fuzziness and Knowledge-based Systems 12:19–24.

Kohavi, R. 1994. Bottom-up induction of oblivious read-once decision graphs. In *Proceedings of the 5th European Conference on Machine Learning*. Springer. 154–169.

Langley, P.; Iba, W.; and Thomas, K. 1992. An analysis of Bayesian classifiers. In *Proceedings of the Tenth National Conference of Artificial Intelligence*. AAAI Press. 223–228.

Oliver, J. J. 1993. Decision graphs an extension of decision trees. In *Proceedings of the Fourth International Workshop on Artificial Intelligence and Statistics*.

Pagallo, G., and Haussler, D. 1990. Boolean feature discovery in empirical learning. *Machine Learning* 5(1):71–100.

Pearl, J. 1988. *Probabilistic reasoning in intelligent systems:networks of plausible inference*. Morgan Kauhmann.

Provost, F. J., and Domingos, P. 2003. Tree induction for probability-based ranking. *Machine Learning* 52(3):199–215.

Provost, F.; Fawcett, T.; and Kohavi, R. 1998. The case against accuracy estimation for comparing induction algorithms. In *Proceedings of the Fifteenth International Conference on Machine Learning*. Morgan Kaufmann. 445–453.

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

Smyth, P.; Gray, A.; and Fayyad, U. 1996. Retrofitting decision tree classifiers using kernel density estimation. In *Proceedings of the Twelfth International Conference on Machine Learning*. Morgan Kaufmann. 506–514.

Webb, G. I.; Boughton, J.; and Wang, Z. 2005. Not so naive bayes: Aggregating one-dependence estimators. *Journal of Machine Learning* 58(1):5–24.

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

Zhang, H., and Su, J. 2004. Conditional independence trees. In *Proceedings of the 15th European Conference on Machine Learning*. Springer. 513–524.