Full Bayesian Network Classifiers

Jiang Su JIANG.SU@UNB.CA
Harry Zhang HZHANG@UNB.CA
Faculty of Computer Science, University of New Brunswick, Canada

Abstract

The structure of a Bayesian network (BN) encodes variable independence. Learning the structure of a BN, however, is typically of high computational complexity. In this paper, we explore and represent variable independence in learning conditional probability tables (CPTs), instead of in learning structure. A full Bayesian network is used as the structure and a decision tree is learned for each CPT. The resulting model is called *full Bayesian network classifiers (FBCs)*. In learning an FBC, learning the decision trees for CPTs captures essentially both variable independence and context-specific independence. We present a novel, efficient decision tree learning, which is also effective in the context of FBC learning. In our experiments, the FBC learning algorithm demonstrates better performance in both classification and ranking compared with other state-of-the-art learning algorithms. In addition, its reduced effort on structure learning makes its time complexity quite low as well.

1. Introduction

A Bayesian network (BN) (Pearl, 1988) consists of a directed acyclic graph G and a set P of probability distributions, where nodes and arcs in G represent random variables and direct correlations between variables respectively, and P is the set of local distributions for each node. A local distribution is typically specified by a conditional probability table (CPT). BNs are often used for the classification problem. In the classification learning problem, a learner attempts to construct a classifier from a given set of labeled training examples that are represented by a tuple of attribute variables used collectively to predict the value of the class variable. This paper focusses on learning BN classifiers.

Learning BNs has become an active research in the past decade (Heckerman, 1999). The goal of learning a BN is to determine both the structure of the network (structure learning) and the set of CPTs (parameter learning). Since the number of possible structures is extremely huge, structure learning often has high computational complexity. Thus, heuristic and approximate learning algorithms are the realistic solution. A variety of learning algorithms have been proposed, such as (Cooper & Herskovits, 1992; Heckerman et al., 1995; Lam & Bacchus, 1994; Cheng et al., 2002). Learning unrestricted BNs, however, seems to not necessarily lead to a classifier with good performance. For example, Friedman et al. (1997) observed that unrestricted BN classifiers do not outperform naive Bayes, the simplest BN classifier, in a large sample of benchmark data sets. Several factors contribute to that. First, the standard BN learning algorithms try to maximize the likelihood-based scoring function, rather than the conditional likelihood-based one (Friedman et al., 1997). The latter seems computationally intractable. Second, the resulting network tends to have a complex structure, and thus has high variance because of the inaccurate probability estimation caused by the limited amount of training data.

One practical approach for structure learning is to impose some restrictions on the structures of BNs, for example, learning tree-like structures, in which each node has at most one parent. This is essentially a strategy with high bias and low variance. When strong and complex variable dependencies do exist, however, some dependencies have to be discarded. That would damage the performance. It would be nice to have a learning algorithm with both low variance and low bias (with full power to represent arbitrary dependence). Nevertheless, structure learning is the bottleneck in BN learning.

The motivation of this paper is to overcome (or allevi-

Appearing in *Proceedings of the 23rd International Conference on Machine Learning*, Pittsburgh, PA, 2006.

ate) the bottleneck of structure learning in BN learning. Our idea is that we do not use the structure to represent variable independence. Instead, we assume that all variables are dependent and use a full BN as the structure of the target BN. We capture and represent variable independence, as well as context-specific independence, in learning CPTs. In fact, a full BN perfectly represents the joint distribution, and learning the structure of a full BN requires dramatically less computation than does learning the structure of an arbitrary BN. A full BN, however, usually has high variance because of its high structural complexity. This issue can be alleviated by exploiting a structured representation for CPTs. In this paper, we propose to build a full BN and then learn a decision tree as the representation of each CPT.

2. Related Work

Naive Bayes is the simplest BN classifier, in which each attribute node (corresponding to an attribute variable) has the class node (corresponding to the class variable) as its parent, but does not have any other parent, shown graphically in Figure 1. Naive Bayes does not represent any variable dependencies given the class variable. Extending its structure to explicitly represent variable dependencies is a direct way to overcome the limitation of naive Bayes. Tree augmented naive Bayes (TAN) is an extended tree-like naive Bayes (Friedman et al., 1997), in which the class node directly points to all attribute nodes and an attribute node can have only one parent from another attribute node in addition to the class node.

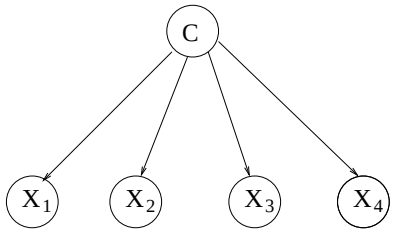

Figure 1. An example of naive Bayes

TAN is a specific case of general augmented naive Bayesian networks (ANBs), which can represent arbitrary variable dependencies. ANBs are the BN classifiers addressed in this paper. Note that learning an ANB is similar to learning a BN.

There are a variety of algorithms proposed for BN structure learning, mainly in two types:

1. Conditional independence based: These attempt to detect the conditional dependency between two nodes by performing tests of conditional independence on the data, then search for a network in terms of the detected dependencies and independencies (Cheng et al., 2002).

2. Score based: Define a score to evaluate how well a structure fits the data, and search for a structure that maximizes the score. There are a variety of ways to define the scoring function, such as the MDL score (Lam & Bacchus, 1994; Friedman et al., 1997), and the Bayesian Dirichlet score (Cooper & Herskovits, 1992; Heckerman et al., 1995). In the score-based approach, a greedy search process is often used, which starts from an initial structure and repeatedly applies local operations (arc addition, arc removal, and arc reversal) to maximally improve the score.

There is another type of independence, called context-specific independence (CSI) (Friedman & Goldszmidt, 1996), which holds only in certain contexts (given the assignment of values to some variables). Friedman and Goldszmidt (1996) propose to use decision trees to represent CPTs to capture CSI.

The most recent work in learning BN classifiers is AODE (averaged one-dependence estimators) (Webb et al., 2005). In AODE, an ensemble of TANs are learned and the classification is produced by aggregating the classifications of all qualified TANs. In AODE, a TAN is built for each attribute variable, in which the attribute variable is set to be the parent of all other attribute variables.

3. Representing Variable Independence in CPTs

The key idea of BNs is to explore and exploit variable independence to obtain a compact representation of the joint distribution. Note that the structure of a BN represents variable independence, which is defined formally as follows.

Definition 1 *Let* $\mathbf{X}$, $\mathbf{Y}$, $\mathbf{Z}$ *be subsets of the variable set* $\mathbf{W}$. $\mathbf{X}$ *and* $\mathbf{Y}$ *are said conditionally independent given* $\mathbf{Z}$*, denoted by* $I(\mathbf{X};\mathbf{Y}|\mathbf{Z})$*, if*

$$P(\mathbf{X}|\mathbf{Y},\mathbf{Z}) = P(\mathbf{X}|\mathbf{Z}), \textit{whenever } P(\mathbf{Y},\mathbf{Z}) > 0.$$

Context-specific independence (CSI) addresses independence with a finer granularity than variable independence, defined as follows (Friedman & Goldszmidt, 1996).

Definition 2 *Let* $\mathbf{X}$, $\mathbf{Y}$, $\mathbf{Z}$, $\mathbf{T}$ *be disjoint subsets of* $\mathbf{W}$. $\mathbf{X}$ *and* $\mathbf{Y}$ *are said to be contextually independent given* $\mathbf{Z}$ *and the context* $\mathbf{t}$ *(an assignment of values to the variables in* $\mathbf{T}$*), denoted by* $I_c(\mathbf{X}; \mathbf{Y}|\mathbf{Z}, \mathbf{t})$*, if*

$$P(\mathbf{X}|\mathbf{Y}, \mathbf{Z}, \mathbf{t}) = P(\mathbf{X}|\mathbf{Z}, \mathbf{t}), \textit{whenever } P(\mathbf{Y}, \mathbf{Z}, \mathbf{t}) > 0.$$

CSI corresponds to regularities in CPTs, and thus can be captured by the structured representation for CPTs. The most naive form of a CPT is a table, which enumerates all the conditional probabilities for each assignment of values to the node and its parents. It has been noticed that decision trees can be used as the representation of CPTs (Friedman & Goldszmidt, 1996) to obtain a more compact representation for CPTs. The decision trees in CPTs are called CPT-trees.

Let Π_X be the parents of X in a BN B. It has been noticed that, if $Y \in \Pi_X$ and Y does not occur in the CPT-tree of X, $I(X, Y|\Pi_X - Y)$ is true (Chickering et al., 1997). This observation implies that, not only CSI, but also variable independence, can be captured and represented by CPT-trees. Thus, we do not need to use the structure of a BN to represent variable independence. Instead, we assume that all variables are fully dependent in structure learning, and explore both variable independence and CSI in learning CPTs. We use a full BN as the structure of the BN. That is a structure in which the first variable is the parents of all other variables, and the second variable is the parents of all other variables except the first one, and so on. Note that a full BN is a perfect representation of the joint distribution, as follows.

$$P(X_1, X_2, \cdots, X_n, C) = P(C)P(X_1|C)P(X_2|X_1, C)\cdots P(X_n|X_{n-1}, \cdots, X_1, C).$$

We call a full BN with CPT-trees *a full Bayesian network classifier* (*FBC*). The structure of an *FBC* does not encode any variable independence, which can, however, be represented in CPTs. For example, the variable independencies represented by the naive Bayes in Figure 1 can be represented by an *FBC*, shown in Figure 2.

We can prove that any variable independencies encoded by a BN B can be represented by an *FBC* as follows. Since B is an acyclic graph, the nodes of B can be sorted on the basis of the topological ordering. Go through each node X in the topological ordering, and add arcs to all the nodes ranked after X and not in Π_X (the set of X's parents). The resulting network FB is a full BN. Build a CPT-tree for each node X in FB, such that any variable that is not in Π_X in B does not occur in the CPT-tree of X in FB. Then, FB essentially represents the same variable independencies as B. Thus, we have the following theorem.

Theorem 1 *For any BN* B*, there is an FBC* FB*, such that* B *and* FB *encode the same variable independencies.*

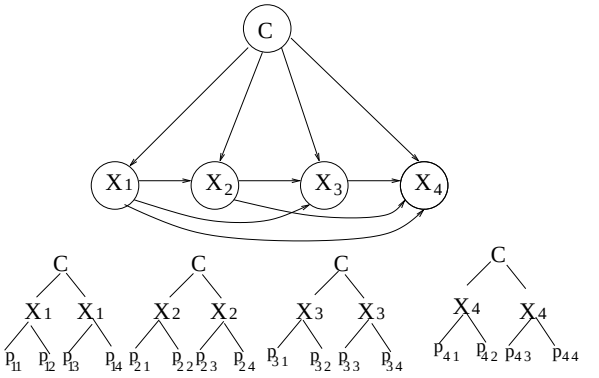

Figure 2. An *FBC* for naive Bayes, in which p_{ij} denotes the corresponding probability $p(X_i|C)$.

FBCs have some obvious advantages. First, in principle, structure learning is not even needed, since any full BN is a perfect representation of the joint distribution. Even though, in practice, different full BNs may perform differently because of the errors caused by the sample's finite size, the number of possible structures of full BNs is $n!$, significantly less than the number of possible structures of even TANs. Thus, it could be expected that learning a full BN has dramatically less computational cost than learning an arbitrary BN. Second, learning a decision tree for a CPT captures both variable independence and CSI. Thus, learning an *FBC* could be more efficient than the traditional approach that learns the structure first to capture variable independence and then learns decision trees for CPTs to capture CSI. In addition, the high variance for a full BN is avoidable in an *FBC* through the CPT-tree representation.

4. Learning Full Bayesian Network Classifiers

Learning an *FBC* consists of two parts: construct a full BN, and then learn a decision tree to represent the CPT of each variable. We implement a full BN using a Bayesian multinet (Heckerman, 1991). A multinet is a set of BNs, each of which corresponds to a value c of the class variable C. A multinet for an *FBC* is correspondingly a set of *FBC*s.

4.1. Learning the Structure of *FBC*

Given a training data set S, we partition S into $|C|$ subsets , each S_c of which corresponds to the class

value c, and then construct an *FBC* for S_c. Note that learning the structure of a full BN actually means learning an order of variables and then adding arcs from a variable to all the variables ranked after it. In fact, given the order of variables, learning a BN is relatively easier. However, learning the order of variables is still difficult. Recently, Teyssier and Koller (2005) propose to learn the order of variables through a sophisticated local search. In this paper, we adopt a straightforward method, which ranks a variable based on its total influence on other variables. The influence (dependency) between two variables can be measured by mutual information defined as follows.

$$M(X;Y) = \sum_{x,y} P(x,y) log \frac{P(x,y)}{P(x)P(y)}, \qquad (1)$$

where x and y are the values of variables X and Y respectively. Note that, since we compute the mutual information in each subset S_c of the training set, $M(X;Y)$ is actually the conditional mutual information $M(X;Y|c)$.

In practice, it is possible that the dependency between two variables, measured by Equation 1, is caused merely by noise. In that case, the dependency is not reliable and should not be taken into account. Thus, we need a threshold to judge if the dependency between two variables is reliable. One typical approach to defining the threshold is based on the *Minimum Description Length* (MDL) principle. Friedman and Yakhini (1996) show that the average cross-entropy error is asymptotically proportional to $\frac{logN}{2N}$, where N is the size of the training data. We adopt their result to define the threshold to filter out unreliable dependencies as follows.

$$\varphi(X_i, X_j) = \frac{logN}{2N} \times T_{ij}, \qquad (2)$$

where $T_{ij} = |X_i| \times |X_j|$, $|X_i|$ is the number of possible values of X_i, and $|X_j|$ is similar. Assume that we use the most naive form as the representation for CPTs. Then, if X_i and X_j are independent, the total size of two CPTs for X_i and X_j is $|X_i| + |X_j|$. If we represent the dependency between X_i and X_j, the total size of the two CPTs are $|X_i| \times |X_j| + |X_i|$ (assuming X_i is the parent of X_j). Thus, roughly speaking, T_{ij} represents the size increase for representing the dependency between X_i and X_j. In our implementation, the dependency between X_i and X_j is taken into account only if $M(X_i;X_j) > \varphi(X_i, X_j)$.

The total influence of a variable X_i on other variables, denoted by $W(X_i)$, is defined as follows.

$$W(X_i) = \sum_{j(j\neq i)}^{M(X_i;X_j)>\varphi(X_i,X_j)} M(X_i;X_j). \qquad (3)$$

Consequentially, we have the following learning algorithm based on $W(X_i)$.

Algorithm FBC-Structure(**S**, **X**)

Input : A set **S** of labeled examples, a set **X** of variables.

Output : A full BN **B**.

1. **B** = empty.
2. Partition the training data **S** into $|C|$ subsets $\mathbf{S}_c$ by the class value c.
3. **For** each training data set $\mathbf{S}_c$
   - Compute the mutual information $M(X_i;X_j)$ and the structure penalty $\varphi(X_i, X_j)$ between each pair of variables X_i and X_j based on Equation 1 and Equation 2.
   - Compute $W(X_i)$ for each variable X_i based on Equation 3.
   - **For** all variables X_i in **X**
     - Add all the variables X_j with $W(X_j) > W(X_i)$ to the parent set Π_{X_i} of X_i.
     - Add arcs from all the variables X_j in Π_{X_i} to X_i.
   - add the resulting network B_c to **B**.
4. Return **B**.

Note that the structure learning algorithm for *FBC* depicted above is $O(n^2 \cdot N)$, where n is the number of variables and N is the size of the training data.

4.2. Learning CPT-trees

After the structure of an *FBC* is determined, a CPT-tree should be learned for each variable X_i. Certainly, a traditional decision tree learning algorithm, such as C4.5, can be used to learn CPT-trees. But the time complexity is typically $O(n^2 \cdot N)$. Note that a decision tree should be learned for each variable. Thus the resulting *FBC* learning algorithm would have a complexity of $O(n^3 \cdot N)$.

We propose a fast decision tree learning algorithm for learning CPTs. Before the tree-growing process, all the variables X_j in Π_{X_i} are sorted in terms of mutual information $M(X_i, X_j)$ on the whole training data, which determines a fixed order of variables. In the tree

growing process, remove the variable X_j with the highest mutual information from Π_{X_i}, and compute the local mutual information $M^{\mathbf{S}}(X_i, X_j)$ on the current training data $\mathbf{S}$. If it is greater than the local threshold $\varphi^{\mathbf{S}}(X_i, X_j)$, X_j is used as the root, and partition the current training data $\mathbf{S}$ according to the values of X_j and repeat this process for each branch (subset). Otherwise, try another variable. The key point here is that, for each variable, we compute the local mutual information and local threshold just once. Whether failed or not, it is removed from Π_{X_i} and is never considered again. In contrast, the traditional decision tree learning algorithm computes the local mutual information of each unused variable at each step. Our algorithm is depicted as follows.

Algorithm Fast-CPT-Tree(Π_{X_i}, $\mathbf{S}$)

Input: Π_{X_i} is the set of parents of X_i, and $\mathbf{S}$ is a set of labeled examples.

Output: A decision tree $\mathbf{T}$ with X_i as the predictive variable.

1. Create an empty tree $\mathbf{T}$.
2. **If** ($\mathbf{S}$ is pure or empty) or (Π_{X_i} is empty) **Return T**.
3. $qualified = False$.
4. **While** ($qualified == False$) and (Π_{X_i} is not empty)
   - Choose the variables X_j with the highest $M(X_j; X_i)$.
   - Remove X_j from Π_{X_i}.
   - Compute the local mutual information $M^{\mathbf{S}}(X_i, X_j)$ on $\mathbf{S}$ based on Equation 1.
   - Compute the local structure penalty $\varphi^{\mathbf{S}}(X_i, X_j)$ on $\mathbf{S}$ based on Equation 2.
   - **If** $M^{\mathbf{S}}(X_j; X_i) > \varphi^{\mathbf{S}}(X_i, X_j)$ $qualified = True$
5. **If** $qualified == True$
   - Create a root X_j for $\mathbf{T}$.
   - Partition $\mathbf{S}$ into disjoint subsets $\mathbf{S}_x$, x is a value of X_j.
   - **For** all values x of X_j
     - $\mathbf{T}_x$=Fast-CPT-Tree(Π_{X_i},$\mathbf{S}_x$)
     - Add $\mathbf{T}_x$ as a child of X_j.
6. **Return T**.

The time complexity of the *Fast-CPT-Tree* algorithm is $O(n \cdot N)$, significantly less than $O(n^2 \cdot N)$, the time complexity of the traditional decision tree learning algorithm. In addition, it does not need to compute $M(X_i, X_j)$, since it has been done in the structure learning. That makes it even more efficient. In the context of CPT-tree learning, it is also effective according to our experiments. Thus, the resulting *FBC* learning algorithm has the time complexity of $O(n^2 \cdot N)$.

Table 1. Description of data sets.

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

5. Experiments

5.1. Experiment Setup

We conduct our experiments under the framework of Weka (Witten & Frank, 2000). In our experiments, we use the 33 UCI data sets, selected by Weka, which represent a wide range of domains and data characteristics. A brief description of the data sets is in Table 1. To better understand the effect of training data size on the algorithm performance, we sort the data sets by their sizes. Numeric variables are discretized using unsupervised ten-bin discretization implemented in Weka. Missing values are also processed using the mechanism in Weka, which replaces all missing values with the modes and means from the training data. In addition, all the preprocessing is done with the default parameters in Weka implementation.

We implemented *FBC* within the Weka framework. The source code is available at *http://www.cs.unb.ca/profs/hzhang/FBC.rar*. For all the other algorithms compared in our experiments, we

use their implementation in Weka. In our experiment, the performance of an algorithm on each data set has been observed via 10 runs of 10-fold stratified cross validation. In our experiments, we conducted the two-tailed t-test with a 95% confidence interval to compare each pair of algorithms on each data set.

5.2. Experimental Results

We observed both classification performance and ranking performance for each algorithm, measured by accuracy and AUC (the area under the Receiver Operating Characteristics curve) (Provost & Fawcett, 1997), respectively. Ranking performance of a classifier is crucial in many applications and has attracted a great deal of attention recently (Provost & Domingos, 2003).

Note that we tried to optimize the algorithms compared in our experiments and the details are depicted as follows.

FBC: *Full Bayesian network classifier.* We adopt Laplace smoothing with a default parameter 0.5 assigned by WEKA.

AODE: *Averaged One-Dependence Estimators* (Webb et al., 2005). All parameter setting is given in (Webb et al., 2005).

HGC: Hill climbing BN learning algorithm (Heckerman et al., 1995). Laplace smoothing with a default parameter 0.5 is applied. In addition, each node is allowed to have at most 5 parents, by which HGC achieves the best performance according to our observation.

TAN: An improved *ChowLiu* algorithm for learning TAN (Friedman et al., 1997). Laplace smoothing with a default parameter 0.5 is applied.

NBT: An algorithm to combine naive Bayes with decision trees by deploying a naive Bayes on each leaf of a decision tree (Kohavi, 1996).

C4.5L: The traditional decision trees induction algorithm C4.5 (Quinlan, 1993) using Laplace smoothing with a default parameter 1.0 (Provost & Domingos, 2003). In addition, pessimistic pruning is applied.

SMO: The *sequential minimal optimization* algorithm for training an SVM classifier using polynomial or RBF kernels (Platt, 1998; Keerthi et al., 2001). Linear kernel, rather than RBF kernel, is used. To obtain proper probability estimates, we use logistic regression models to calibrate the output probabilities.

Table 2 and Table 4 show the accuracies and AUC of the classifiers. Table 3 and Table 5 show 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.

From our experimental results, *FBC* demonstrates good performance in both classification and ranking. Overall, the performance of *FBC* is the best among the algorithms compared.

Table 3. Summary of the experimental results on accuracy.

	AODE	HGC	TAN	NBT	C4.5	SMO
FBC	8/22/3	4/27/2	6/27/0	6/27/0	11/19/3	6/24/2
AODE		5/20/8	5/24/4	5/24/5	11/18/4	5/22/6
HGC			7/23/3	6/26/1	9/23/1	5/26/2
TAN				4/26/3	12/17/4	4/22/7
NBT					8/24/1	3/28/2
C4.5						5/23/5

5.3. Computational Complexity

The efficiency of a learning algorithm is critical in many applications. The time complexities of the algorithms compared in our experiments are summarized in Table 6.

Note that *FBC* is among the most efficient algorithms in both training time and classification time. In addition, according to our experiments, *FBC* is among the fastest algorithms in terms of training time. For example, SMO took 586 seconds on the "Letter" data set, while NBT took 302 seconds and *FBC* only used 3 seconds. Due to the space limit, we do not systematically present the experimental results in running time.

6. Conclusion

In this paper, we propose the use of a full BN as the structure of the BN classifier and explore both variable independence and CSI in learning CPT-trees. We present an efficient decision tree learning algorithm for learning CPT-trees. The *FBC* learning algorithm proposed in this paper demonstrates good performance in both classification and ranking, and has a relatively low time complexity.

In learning an FBC, the variable order plays an important role. In this paper, we used a straightforward but efficient method to determine the order. A natural question is, can a more sophisticated method yielding a more accurate variable order improve the performance of FBC further? This is a topic for our future research.

Table 2. Experimental results on accuracy

Data Set	*FBC*	AODE	TAN	HGC	SMO	C4.5L	NBT
Letter	89.10±0.64	85.54±0.68•	83.11±0.75•	87.15±0.85 •	88.65±0.69	80.51±0.78•	83.49±0.81•
Mushroom	100.00±0	99.95±0.07•	99.99±0.03	100.00±0	100.00±0	100.00±0	100.00±0
Waveform	80.24±1.64	84.24±1.6 ○	80.72±1.78	82.26±1.62 ○	82.56±1.75○	72.21±1.79•	81.62±1.76
Sick	97.79±0.72	97.52±0.72	97.61±0.73	97.84±0.73	97.64±0.71	98.18±0.67	97.86±0.69
Hypothyroid	93.16±0.56	93.56±0.61○	93.23±0.68	93.39±0.55	93.48±0.5	93.24±0.44	93.05±0.65
Chess	96.36±1.16	91.03±1.66•	92.05±1.49•	95.64±1.29	94.78±1.39•	99.44±0.37○	97.81±2.05
Splice	95.12±1.11	96.12±1 ○	95.39±1.16	96.51±0.99 ○	92.92±1.45•	94.08±1.29•	95.42±1.14
Segment	95.01±1.38	92.92±1.4 •	94.54±1.6	94.79±1.29	94.38±1.78	93.19±1.69•	92.64±1.61•
German Credit	75.27±3.54	76.45±3.88	75.86±3.58	73.08±3.85	75.35±3.73	72.25±3.48•	75.54±3.92
Vowel	93.46±2.65	89.64±3.06•	93.10±2.85	92.70±2.87	86.86±3.55•	73.29±4.69•	88.01±3.71•
Anneal	99.04±1.07	96.83±1.66•	98.34±1.18	98.94±1	99.23±1.23	98.76±0.99	98.40±1.53
Vehicle	74.23±3.43	71.65±3.59•	73.71±3.48	68.10±4.14 •	68.80±3.83•	70.38±3.69•	68.91±4.58•
Pima Diabetes	74.85±4.33	76.57±4.53	75.09±4.96	75.92±5.34	74.43±4.36	73.88±4.64	75.28±4.84
Wisconsin-breast	97.25±1.77	96.91±1.84	95.05±2.24•	96.94±1.68	95.95±2.13	91.86±3.01•	97.17±1.75
Credit Approval	84.81±3.98	85.78±3.75	84.22±4.41	86.00±3.95	84.94±3.81	85.32±4.41	84.55±4.11
Soybean	94.41±2.39	93.31±2.85	95.24±2.28	95.13±2.43	93.00±3.05	92.55±2.61	92.30±2.7 •
Balance-scale	91.44±1.3	89.78±1.88•	86.22±2.82•	91.44±1.3	89.55±3.57	64.14±4.16•	91.44±1.3
Vote	94.36±3.34	94.52±3.19	94.57±3.23	94.87±2.92	96.00±2.95	96.27±2.79	94.78±3.32
Horse Colic	79.22±5.93	81.26±5.83	80.55±6.23	81.28±5.41	80.14±5.38	84.83±5.93○	82.58±5.65
Ionosphere	90.80±4.41	91.74±4.28	92.23±4.36	93.40±4.12	89.15±4.71	87.61±5.55	89.18±4.82
Primary-tumor	47.32±5.38	47.87±6.37	46.76±5.92	41.48±7.43 •	43.32±6.4	41.01±6.59•	45.84±6.61
Heart-c	83.34±6.41	83.07±7.05	82.78±6.98	81.83±7.14	82.57±7.26	79.61±6.49	81.23±6.68
Breast cancer	70.39±8.2	72.73±7.01	70.09±7.68	72.84±6.65	69.59±7.45	75.26±5.04○	71.66±7.92
Heart-statlog	83.81±5.39	83.63±5.32	79.37±6.87•	82.59±6.39	82.48±6.34	79.85±7.95	82.26±6.5
Audiology	74.05±6.68	71.66±6.42	72.68±7.02	73.36±7.47	80.38±7.5 ○	76.69±7.68	76.66±7.47
Glass	62.44±8.48	61.73±9.69	58.43±8.86	58.36±9.3	64.75±8.53	58.28±8.52	58.00±9.42
Sonar	77.38±10.2	79.91±9.6	73.66±10.1	69.16±11.42	75.45±9.57	70.99±8.99	71.40±8.8
Autos	76.70±9.62	75.09±10.2	76.98±9.21	79.52±9.37	79.39±8.24	77.86±9.02	77.78±9.59
Hepatitis	86.90±7.94	83.55±9.73	82.13±9.17•	80.19±10.1 •	80.86±10.2•	81.50±8.24•	81.69±9.25•
Iris	93.73±7.02	94.00±5.88	91.67±7.18	96.07±4.65	95.87±4.9	96.00±4.64	95.27±6.16
Lymph	85.20±8.58	85.46±9.32	84.07±8.93	82.61±9.42	81.64±9.28	78.21±9.74	82.21±8.95
Zoo	94.17±6.76	94.66±6.38	93.69±7.75	96.74±5.47	92.52±7.5	92.61±7.33	94.65±6.39
Labor	94.90±9.18	94.73±8.79	87.67±13.8	89.57±11.9	84.93±13.8•	84.97±14.2	94.70±9.29

○, • statistically significant improvement or degradation

Table 5. Summary of the experimental results on AUC.

	AODE	HGC	TAN	NBT	C4.5L	SMO
FBC	7/22/4	6/25/2	9/24/0	8/24/1	25/7/1	10/20/3
AODE		5/21/7	8/20/5	9/22/2	26/6/1	9/19/5
HGC			9/22/2	7/24/2	22/10/1	8/22/3
TAN				6/23/4	20/12/1	7/20/6
NBT					19/13/1	1/26/6
C4.5L						1/11/21

Table 6. Summary of the time complexities of algorithms.

	Training	Classification
FBC	$O(n^2 \cdot N)$	$O(n)$
AODE	$O(n^2 \cdot N)$	$O(n^2)$
HGC	$O(n^4 \cdot N)$	$O(n)$
TAN	$O(n^2 \cdot N)$	$O(n)$
NBT	$O(n^3 \cdot N)$	$O(n)$
C4.5	$O(n^2 \cdot N)$	$O(n)$
SMO	$O(N^{2.3})$	$O(n)$

References

Cheng, J., Greiner, R., Kelly, J., Bell, D., & Liu, W. (2002). Learning Bayesian networks from data: An information-theory based approach. *Artificial Intelligence Journal*, *137:1-2*, 43–90.

Chickering, D., Heckerman, D., & Meek, C. (1997). A Bayesian approach to learning Bayesian networks with local structure. In *Proceedings of Thirteenth conference on Uncertainty in Artificial Intelligence*, 80–89. Morgan Kaufmann.

Cooper, G., & Herskovits, E. (1992). A Bayesian method for the induction of probabilistic networks from data. *Machine Learning*, *9*, 309–347.

Friedman, N., Geiger, D., & Goldszmidt, M. (1997). Bayesian network classifiers. *Machine Learning*, *29*, 131–163.

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

Friedman, N., & Yakhini, Z. (1996). On the sample complexity of learning Bayesian networks. In *Proceedings of twelfth conference on uncertainty in artificial intelligence*, 274–282.

Heckerman, D. (1991). *Probabilistic similarity networks*. MIT Press.

Heckerman, D. (1999). A tutorial on learning with Bayesian networks. In M. I. Jordan (Ed.), *Learning in Graphical Models*, 301–354. MIT Press.

Table 4. Experimental results on AUC

Data Set	*FBC*	AODE	TAN	HGC	SMO	C4.5L	NBT
Letter	99.63±0.06	99.41±0.06•	99.12±0.09•	99.48±0.08•	99.49±0.07 •	98.51±0.17•	96.49±0.3 •
Mushroom	100.00±0	100.00±0	100.00±0	100.00±0	100.00±0	100.00±0	100.00±0
Waveform	93.83±0.76	96.46±0.56○	93.87±0.8	94.75±0.76○	94.85±0.75 ○	86.95±1.2 •	93.71±0.96
Sick	98.93±0.63	97.09±1.69•	98.31±1.04•	98.16±1.13•	93.94±4.25 •	93.79±4.19•	94.46±3.47•
Hypothyroid	87.81±6.9	87.34±7.23	87.84±7.1	86.91±6	86.60±7.36	63.68±6.65•	87.66±6.75
Chess	99.34±0.36	97.44±0.75•	98.06±0.63•	99.06±0.39•	99.13±0.4	99.88±0.12○	99.44±0.6
Splice	99.46±0.26	99.56±0.25○	99.40±0.35	99.52±0.34	98.35±0.62 •	98.09±0.84•	99.44±0.31
Segment	99.66±0.18	99.42±0.22•	99.63±0.18	99.63±0.17	99.46±0.32 •	98.88±0.45•	99.11±0.33•
German Credit	77.40±4.61	79.68±4.14○	77.92±4.82	74.10±5.18•	78.03±4.47	72.18±4.47•	77.53±5.24
Vowel	99.74±0.2	99.40±0.36•	99.64±0.25	99.61±0.29	98.60±0.68 •	90.56±2.5 •	98.59±0.8 •
Anneal	96.57±0.4	96.27±0.93	96.59±0.13	96.49±0.6	96.31±1.06	89.12±4.34•	96.31±1.1
Vehicle	90.90±1.88	89.91±2.03	91.14±1.89	88.45±2.18•	86.66±2.81 •	87.02±2.7 •	85.86±3.3 •
Pima Diabetes	81.54±4.9	82.96±4.83	81.33±5.19	83.80±5.3 ○	81.64±5.23	77.59±6.51•	82.11±5.06
Wisconsin-breast	99.29±0.7	99.29±0.72	98.73±1.06•	99.43±0.64	98.69±1.18	97.50±2.13•	99.28±0.72
Credit Approval	92.08±3.11	92.28±2.95	91.25±3.22	92.41±3.17	90.39±3.47 •	90.35±3.72•	91.32±3.31
Soybean	99.85±0.23	99.82±0.23	99.87±0.23	99.85±0.25	99.79±0.23 •	97.89±1.64•	99.72±0.32
Balance-scale	84.42±4.39	79.93±3.94•	78.34±5.04•	84.42±4.39	95.72±2.63 ○	56.56±8.43•	84.64±4.34○
Vote	98.49±1.47	98.67±1.24	98.78±1.22	98.77±1.42	98.94±1.62	97.59±2.26	98.61±1.57
Horse Colic	84.19±7.09	86.97±5.87○	84.90±6.54	86.39±5.61	84.04±6.53	85.02±6.87	87.20±6.46
Ionosphere	97.88±2.23	97.57±2.23	98.08±2.17	98.23±2.31	94.48±3.66 •	89.87±5.15•	94.31±4.22•
Primary-tumor	75.51±2.75	75.68±2.68	75.43±2.67	75.22±2.43	75.34±2.74	68.38±3.04•	74.71±2.75
Heart-c	84.09±0.6	84.11±0.58	84.02±0.59	83.97±0.6	83.99±0.59	83.29±0.65•	84.00±0.58
Breast cancer	68.69±10.3	71.18±10.0	66.18±10.7	64.20±11.6	64.98±12.2	62.26±9.25•	67.98±10.3
Heart-statlog	91.16±4.91	91.28±4.7	88.19±5.75•	90.27±5.2	89.64±5.66	84.91±7.88•	89.83±5.82
Audiology	70.42±1.01	70.05±1.08•	70.25±1.05	70.39±1.14	70.79±0.94	62.04±2.36•	70.17±1.26
Glass	81.20±6.17	78.86±6.21	78.32±6.12•	81.98±5.88	86.24±5.07 ○	80.00±5.96	79.13±6.39
Sonar	86.23±8.83	90.00±6.76	82.04±9.73	76.50±12.4•	82.91±10.11	73.80±10.9•	79.18±9.38•
Autos	94.26±3.06	93.92±2.74	94.38±2.5	95.53±1.92	94.76±2.63	91.42±3.66	94.60±2.72
Hepatitis	90.61±8.76	87.41±11.2	83.32±11.9•	85.15±11.3	81.29±13.2 •	70.32±14.6•	84.18±12.8•
Iris	98.87±1.74	99.16±1.42	98.49±2.44	99.14±1.53	98.13±2.9	98.58±2.09	98.85±2
Lymph	89.84±2.44	90.03±2.28	89.16±3.28	89.55±2.51	88.47±3.94	86.21±5.3	88.94±2.81
Zoo	88.88±2.83	88.93±2.82	86.67±4.54•	88.98±3.13	88.10±4.03	80.10±6.28•	89.02±2.99
Labor	97.96±6.77	97.88±8.04	93.29±13.9	95.87±9.56	92.42±15.1	84.15±18.1•	93.87±17.0

○, • statistically significant improvement or degradation

Heckerman, D., Geiger, D., & Chickering, D. M. (1995). Learning Bayesian networks: The combination of knowledge and statistical data. *Machine Learning*, *20*, 197–243.

Keerthi, S., Shevade, S., Bhattacharyya, C., & Murthy, K. (2001). Improvements to Platt's SMO algorithm for SVM classifier design. *Neural Computation*, *13(3)*, 637–649.

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*, 202–207. AAAI Press.

Lam, W., & Bacchus, F. (1994). Learning Bayesian belief networks: an approach based on the MDL principle. *Computational Intelligence*, *10(4)*, 269–293.

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

Platt, J. C. (1998). Fast training of support vector machines using sequential minimal optimization. *Advances in Kernel Methods–Support Vector Learning*. MIT Press.

Provost, F., & Fawcett, T. (1997). Analysis and visualization of classifier performance: comparison under imprecise class and cost distribution. In *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining*, 43–48. AAAI Press.

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

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

Teyssier, M., & Koller, D. (2005). Ordering-based search: A simple and effective algorithm for learning bayesian networks. In *Proceedings of the Twenty-first Conference on Uncertainty in Artificial Intelligence*, 584–590.

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

Witten, I. H., & Frank, E. (2000). *Data mining – practical machine learning tools and techniques with Java implementation*. Morgan Kaufmann.