Comparison of Lazy Bayesian Rule and Tree-Augmented Bayesian Learning

Zhihai Wang
School of Information Technology,
Deakin University, 3125, Australia
zhw@deakin.edu.au

Geoffrey I. Webb
School of CSSE, Monash University,
Victoria, 3800, Australia
webb@csse.monash.edu.au

Abstract

The naive Bayes classifier is widely used in interactive applications due to its computational efficiency, direct theoretical base, and competitive accuracy. However, its attribute independence assumption can result in sub-optimal accuracy. A number of techniques have explored simple relaxations of the attribute independence assumption in order to increase accuracy. Among these, the lazy Bayesian rule (LBR) and the tree-augmented naive Bayes (TAN) have demonstrated strong prediction accuracy. However, their relative performance has never been evaluated. This paper compares and contrasts these two techniques, finding that they have comparable accuracy and hence should be selected according to computational profile. LBR is desirable when small numbers of objects are to be classified while TAN is desirable when large numbers of objects are to be classified.

1. Introduction

The Naive Bayesian classifier is one of the most computationally efficient algorithms for machine learning and data mining. It has been shown in many domains to be surprisingly accurate compared to alternatives including decision tree learning, rule learning, neural networks, and instance-based learning [10], [11], [4], [8], [5], [12]. It is based on Bayes' theorem and an assumption that all attributes are mutually independent within each class.

Assume X is a finite set of instances, and $A = \{A_1, A_2, \cdots, A_n\}$ is a finite set of n attributes. An instance $x \in X$ is described by a vector $< a_1, a_2, \cdots, a_n >$, where a_i is a value of attribute A_i. C is called the class attribute. Prediction accuracy will be maximized if the predicted class $L(< a_1, a_2, \cdots, a_n >) = argmax_c(P(c| < a_1, a_2, \cdots, a_n >)$. Unfortunately, unless $< a_1, a_2, \cdots, a_n >$ occurs many times within X, it will not be possible to directly estimate $P(c| < a_1, a_2, \cdots, a_n >)$ from the frequency with which each class $c \in C$ co-occurs with $< a_1, a_2, \cdots, a_n >$ within X. Bayes' theorem provides an equality that might be used to help estimate $P(c_i|x)$ in such a circumstance:

$$P(c_i|x) = \frac{P(c_i)P(< a_1, a_2, \cdots, a_n > |c_i)}{P(< a_1, a_2, \cdots, a_n >)}. \quad (1)$$

If the n attributes are mutually independent within each class value, then the probability is directly proportional to:

$$P(c_i| < a_1, a_2, \cdots, a_n >) \propto P(c_i) \prod_{k=1}^{n} P(a_k|c_i). \quad (2)$$

Classification selecting the most probable class as estimated using formulas 1 and 2 is the well-known naive Bayesian classifier.

2. Approaches of improving naive Bayesian method

The attribute independence assumption makes the application of Bayes' theorem to classification practical in many domains, but this assumption rarely holds in real world problems. Notwithstanding Domingos and Pazzani's (1996) analysis that demonstrates that some violations of the independence assumption are not harmful to classification accuracy [4], previous research has shown that semi-naive techniques [10] and Bayesian networks [5] that explicitly adjust the naive strategy to allow for violations of the independence assumption, can improve upon the prediction accuracy of the naive Bayesian classifier in many domains.

One approach is to select attribute subsets. The selective Bayesian classifier [11] is a variant of the naive method that uses only a subset of the given attributes in making predictions. Kohavi and John (1997) use best-first search, based on accuracy estimates, to find a subset of attributes [9]. Their algorithm can wrap around any classifiers, including either the decision tree classifiers or the naive Bayesian classifiers. Another of the most important research approaches is directly to relax the independence assumptions.

0-7695-1754-4/02 $17.00 © 2002 IEEE

Kononenko (1991) proposed a semi-naive Bayesian classifier [10], which partitioned the attributes into disjoint groups and assumed independence only between attributes of different groups. Pazzani (1996) proposed an algorithm based on the wrapper model for the construction of Cartesian product attributes to improve the naive Bayesian classifier [13].

Friedman, Geiger and Goldszmidt (1997) compared the naive Bayesian method and Bayesian network, and showed that using unrestricted Bayesian networks did not generally lead to improvements in accuracy and even reduced accuracy in some domains [5]. They presented a compromise representation, called tree-augmented naive Bayes (TAN), in which the class node directly points to all attributes nodes and an attribute node can have only at most one additional parent to the class node. Based on this presentation, they utilized the concept of mutual information to efficiently find the best tree-augmented naive Bayesian classifier. Keogh and Pazzani (1999) took a different approach to constructing tree-augmented Bayesian networks [7]. They use the same representation, but use leave-one-out cross validation to estimate the classification accuracy of the network when an arc is to add. The two methods mainly differ in the criterion of attribute selection used to select dependence relations among the attributes while building a tree-augmented Bayesian network.

Zheng and Webb (2000) proposed the lazy Bayesian rule (LBR) learning technique [17]. LBR can be thought of as applying lazy learning techniques to naive Bayesian rule induction. At classification time, for each test example, it builds a most appropriate rule with a conjunction of conditions as its antecedent and a local naive Bayesian classifier as its consequent. LBR has been compared experimentally with a naive Bayesian classifier, a decision tree classifier, a Bayesian tree learning algorithm, a constructive Bayesian classifier, a selective naive Bayesian classifier, and a lazy decision tree algorithm in a wide variety of natural domains. In their extensive experiments, LBR obtained lower error than all the alternative algorithms.

Both LBR and TAN can be viewed as variants of naive Bayes that relax the attribute independence assumption. TAN relaxes this assumption by allowing each attribute to depend upon at most one other attribute in addition to the class. LBR allows an attribute to depend upon many other attributes, but all attributes depend upon the same set of other attributes. These two different approaches to relaxing the attribute independence assumption have not previously been compared. This paper compares the two techniques, focusing primarily on prediction accuracy but also investigating the significance of the use of lazy learning in LBR in comparison to the use of eager learning in TAN, and the criteria used for attribute selection.

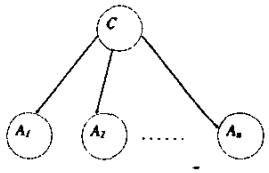

Figure 1. The Bayesian network structure of the naive Bayesian classier

3. The representation of dependencies

Bayesian networks have been a popular medium for graphically representing and manipulating attribute interdependencies. Bayesian networks are directed acyclic graphs (DAG) that allow for efficient and effective representation of joint probability distributions over a set of random variables. Each vertex in the graph represents a random variable, and each edge represents direct correlations between the variables. Each variable is independent of its non-descendants given its parents in the graph.

3.1. Basic Bayesian network

We now more formally introduce some notation for Bayesian networks. Let $A = \{A_1, A_2, \cdots, A_n\}$ be a finite set of discrete random variables where each variable A_i may take on values from a finite domain. A Bayesian network is an annotated directed acyclic graph that encodes a joint probability distribution over A. It can be formally defined as follows.

Definition 1. A Bayesian network is defined as a pair:

$$\mathbf{B} =< \mathbf{G}, \mathbf{\Theta} > \tag{3}$$

where $\mathbf{G} =< \mathbf{N}, \mathbf{E} >$ is a directed acyclic graph where each node $A \in \mathbf{N}$, corresponds to a random variable (an attribute in standard naive Bayesian terminology) and where each arc $E \in \mathbf{E}$ represents a direct dependence between variables. $\mathbf{\Theta}$ is the set of parameters that quantifies the Bayesian network, in which each $\theta \in \mathbf{\Theta}$ represents a conditional probability distribution for the corresponding node.

Bayesian networks provide a kind of direct and clear representation for the dependencies among the variables or attributes. These dependencies can be exploited to calculate the posterior probability $P(< a_1, a_2, \cdots, a_n > |c_i)$ in formula 1. A Bayesian network for the naive Bayesian classifier is the simple structure depicted in Figure 1, which has the class node as the parent node of all other nodes. No

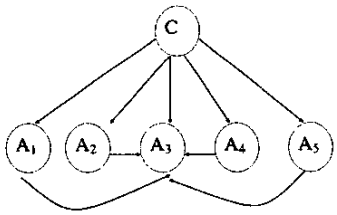

Figure 2. A Bayesian network for TAN

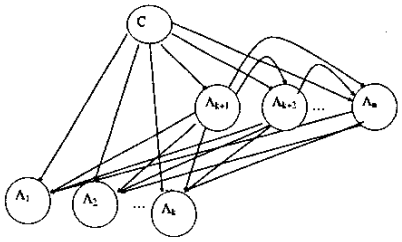

Figure 3. A Bayesian network for LBR

other arcs are allowed in its structure. That means every attribute is independent from the rest of the attributes given the state of the class variable. The naive Bayesian classifier has surprisingly outperformed many sophisticated classifiers over a large number of data sets, especially where the attributes are not strongly correlated. However, when the attribute independence assumption is violated, which appears to be very common, the performance of the naive Bayesian classifier might be poor.

3.2. Tree-augmented Bayesian networks

From the viewpoint of the representation of dependencies, we can think of the structure of the naive Bayesian network as being the most restrictive case of the attribute dependencies, in that it strictly allows no dependencies between attributes given the class variable. In order to improve the performance of the naive Bayesian classifier and reduce the negative impact of the independence assumption, many efforts have been made to extend the structure of the naive Bayesian classifier to account for dependence between attributes [14, 5, 7, 2, 3, 16]. Both of the algorithms of Friedman et al (1997) and Keogh et al (1999) are based on tree-augmented Bayesian networks, which can be viewed as a compromise between the restrictive naive Bayesian network and an unrestricted Bayesian network.

Definition 2. A tree-augmented Bayesian network is defined by the following conditions:

- Each attribute has the class attribute as a parent;
- Attributes may have one other attribute as a parent.

Figure 2 shows an example of an augmented Bayesian network. There are two reasons for this restriction [7]. First, it reduces the search space of classifiers that must be considered. Secondly, the probability estimates for a node become more unreliable as additional parents are added, because the size of the conditional probability tables increases exponentially with the number of parents. In a tree-augmented Bayesian network, a node without any parent, other than the class node, is called an orphan. Given a tree-augmented Bayesian network, if we extend arcs from node A_k to every orphan node A_i, then node A_k is said to be a super parent. Assume that $Parents(v)$ is the set of all parents of node v, and $Parents$ is the set of $Parents(v)$ for all nodes in a tree-augmented Bayesian network, i.e., $Parents = \{Parents(v) : \forall v \in V\}$. Actually, here $Parents$ just represents a tree-augmented Bayesian network. If v is an orphan, then $Parents(v) = \emptyset$.

3.3. Lazy Bayesian Rule Learning

Lazy learning algorithms exhibit three characteristics that distinguish them from other learning algorithms [1]. First, they defer processing of their inputs until they receive requests for information; they simply store their inputs for future use. Next, they reply to information requests by combining their stored data. Finally, they discard the constructed answer and any intermediate results. This provides another approach to alleviating the small disjunct problem of decision tree learning, and further improving the performance of naive Bayesian classification.

In general, any lazy learning approach to building a Bayesian network will build a new network based on each given test instance. For a given test instance and training instances, the network is initialised to a naive Bayesian network, which means all nodes in the network are orphans and $Parents = \emptyset$. For any training instance, each attribute only has two values to be used in the network. One is equal to the value of the test instance, and another value is not equal to the value of the test instance on this attribute. We don't need to differentiate any different value that is not equal to the value on the test instance.

3.4. The Bayesian network for a lazy Bayesian rule

The lazy Bayesian rule (LBR) learning technique [18, 17] can be thought of as applying lazy learning techniques to Bayesian rule induction. LBR is similar to $LazyDT$ (Lazy Decision Tree learning algorithms) [6], which can be considered to generate decision rules at classification time.

For each instance to be classified, $LazyDT$ builds one rule that is most appropriate to the instance by using an entropy measurement. The antecedent of the rule is a conjunction of conditions in the form of attribute-value pairs. The consequent of the rule is the class to be predicted, being the majority class of the training instances that satisfy the antecedent of the rule. LBR can be considered as a combination of the two techniques $NBTree$ and $LazyDT$. Before classifying a test instance, it generates a Bayesian rule that is most appropriate to the test instance. Alternatively, LBR can be viewed as a lazy approach to classification using the following variant of Bayes theorem,

$$P(C_i|V_1 \wedge V_2) = P(C_i|V_2)P(V_1|C_i \wedge V_2)/P(V_1|V_2) \quad (4)$$

Here, V_1 and V_2 are any two conjunctions of attribute values such that each a_i from belongs to exactly one of V_1 or V_2. The structure of a Bayesian network for a lazy Bayesian rule is shown in Figure 3, here $V_1 = \{A_1, A_2, \cdots, A_k\}$ and $V_2 = \{A_{k+1}, A_{k+2}, \cdots, A_n\}$. At classification time, for each instance to be classified, the attribute values in V are allocated to V_1 and V_2 in a manner that is expected to minimize estimation error. The antecedent of a Bayesian rule is the conjunction of attribute-value pairs from the set V_2, and the consequent of a Bayesian rule is a local naive Bayesian classifier created from those training instances that satisfy the antecedent of the rule, which only uses those attributes that belong to the set V_1. That is to say, we have known there are some dependencies among the attributes in V_2, but we don't need to clarify what they are. And a weaker assumption that all the attributes in V_1 are mutual independent given class value and V_2 is hold. During the generation of a Bayesian rule, the test instance to be classified is used to guide the selection of attributes for creating attribute-value pairs. The values in the attribute-value pairs are always the same as the corresponding attribute values of the test instance. The objective is to grow the antecedent of a Bayesian rule that ultimately decreases the errors of the local naive Bayesian classifier in the consequent of the rule.

4. Classification using Bayesian networks

On In this section, we will discuss some issues to be related to our Bayesian learning framework. In general speaking, a Bayesian network allows for modeling of arbitrarily complex dependencies between attributes. Sahami (1996) makes use of the notion of k-dependence Bayesian classifiers to illustrate the relationship of the naive Bayesian classifier and a full unrestricted Bayesian classifier [14].

Definition 3. A k-dependence Bayesian classifier is a Bayesian network which contains the structure of the naive Bayesian classifier and allows each attribute to have a maximum of k attribute nodes as its parents.

The naive Bayesian classifier is a 0-dependence Bayesian classifier. The full unrestricted Bayesian Network classifier is a $(n-1)$-dependence Bayesian classifier, where n is the number of domain attributes. Thus, Bayesian networks have much representational power at the cost of computationally expensive learning and inference. Restricting the number of parents to two could mitigate problems in estimating probabilities from the training data while allowing some amount of dependencies among attributes to be represented.

Finding the best Bayesian network is an intractable problem. What is the best Bayesian network for the training data? In building a Bayesian network, how to evaluate a new Bayesian network when an arc is added? Friedman, Geiger and Goldszmidt (1997) described the problem of learning a Bayesian network from training data as follows [5].

Statement 1. Given a training data set $X = \{x_1, x_2, \cdots, x_m\}$ of instances of $A^* = \{A_1, A_2, \cdots, A_n, C\}$, find a Bayesian network B that best matches the training set X.

However, the use of such criteria runs the risk of overfitting. Cerquides (1999) presented an alternative statement as follows [2].

Statement 2. Given a training data set $X = \{x_1, x_2, \cdots, x_m\}$ of a probability distribution P^*, find the Bayesian network B that best matches P^*.

In practice, this statement is still not operative with respect to building a Bayesian network. In order to efficiently express the classification purpose, we prefer the following statement.

Statement 3. Given a training data set $X = \{x_1, x_2, \cdots, x_m\}$ of instances of $A^* = \{A_1, A_2, \cdots, A_n, C\}$, find a Bayesian network B that has the best classification accuracy.

The algorithm of Friedman, Geiger and Goldszmidt (1997) builds a tree structure over the training data using the concept of mutual information tests conditioned on class variable [5]. Cerquides' (1999) algorithm uses a variant of mutual information (*loglikelihood*) to evaluate a network [2]. Neither algorithm directly reflects classification accuracy. Keogh and Pazzani (1999) take a different approach to evaluating a tree-augmented Bayesian network [7]. They used leave-one-out cross validation to estimate the accuracy of the network with that arc added.

LBR [17] also uses leave-one-out cross validation and a significance test to manage the trade-off between the degree to which the attribute independence assumption of the naive Bayesian classifier is violated and the number of training instances available for training the local naive Bayesian classifier.

5. Lazy tree-augmented Bayesian network learning

LBR and TAN differ in two respects. The first is the types of inter-dependencies that they allow. TAN allows every node to have different parents, but each node may have at most one parent. LBR requires all nodes to have the same parents, but places no restriction on the number of parents. The primary reason for each of these restrictions is to restrict the space of alternative networks that is explored, both to reduce computation and also to reduce variance.

The second respect in which they differ is that LBR uses lazy learning while TAN uses eager learning. In consequence, LBR may create a different network for each case to be classified while TAN will apply the same network to all cases classified. To evaluate the effect of this latter difference, we implement a lazy version of TAN ($LazyTAN$). For a lazy learning method, the Bayesian classifier is only based on specific attribute values of the test instance to be classified.

Now, we further give some notations and functions to be used in the following lazy learning algorithm for building tree-augmented Bayesian classifiers, which is used mainly to explain the differences between the lazy Bayesian rule learning technique and the tree-augmented Bayesian network learning technique. We will then explain some special issues for our implementations. We also discuss criteria for attribute selection while building a Bayesian network to predict the class of an unseen instance.

5.1. Basic functions description

Assume e is a conjunction of some attributes values, then X_e is the subset of X that satisfies the conjunction of attributes values e. And $Parents(v)$ is the set of all parents of node v, and $Parents$ is the set of $Parents(v)$ for all nodes in a Bayesian network, i.e., $Parents = \{Parents(v) : \forall v \in V\}$. If v is an orphan, then $Parents(v) = \emptyset$. When an arc is to added into the current Bayesian network, we use the following functions to check whether the new Bayesian network is still a tree-augmented Bayesian network.

- $Ancestor(b, a)$ is a Boolean function, its value is true if $(b \in Parenst(a))$ or $(\exists v \in Parents(a) : Ancester(v, b))$.

Table 1. The Description of LazyTAN

ALGORITHM: LazyTAN (X, V, C, s)
INPUT: 1) X is the set of training instances,
2) V is the set of attribute values of the test,
3) C is the set of class values,
OUTPUT: Estimated values of the probabilities
$P(a_1 \wedge a_2 \wedge \cdots \wedge a_n | c_i)$.
FOR all $v \in V$, $Parents(v) = \emptyset$
/* Initialize the parents of each value */
$MiniErrors = Estimate(X, \mathbf{Parents})$
DO
$BestSuperParent = Argmin_{v \in V}$
$Estimate(X, InsertAll(\mathbf{Parents}, v))$
/* This is the value for which making that value a parent of all qualified values will have the greatest effect.*/
IF $Eestimate(X, InsertAll(Parents, a)) >$
$MiniErrors$ THEN BREAK
$b = Argmin_{\{v: Qualified(a,v)\}}$
$Estimate(X, Insert(Parents(v), a, v))$
/* This selects the value which is most desirable to add to a.*/
IF $Estimate(X, Insert(Parents(b), a, b)) >$
$MiniErrors$ THEN BREAK
$Parents = Insert(Parents(b), a, b)$
RETURN
the class probability estimates for V and $Parents$.

- $Qualified(b, a)$ is a Boolean function to check whether an arc, which is from node a to node b, can be added. Its value is true if $\neg ancestor(b, a)$ and $Parents(b) = \emptyset$.

The first condition $\neg ancestor(b, a)$ ensures the network is still acyclic if the arc is added. The second condition $Parents(b) = \emptyset$ guarantees node a is a unique parent for node b. The function $Insert(Parents(b), a, b)$ add nodes a into $Parents(b)$. That is, it adds an arc from node a to node b into the current Bayesian network. The function $InsertAll(\mathbf{Parents}, b)$ is the result of adding b to each node a of **Parents** such that $qualified(b, a)$. Given a tree-augmented network $\mathbf{B}$, if we extend an arc from node A_k to each orphan in turn, and test the effect on predictive accuracy, the node pointed to by the best arc is said to be the favourite child of node A_k.

The evaluation of classification accuracy is done by $Estimate(X, \mathbf{Parents})$, which is an integer function, its value is the number of instances which are classified incorrectly using leave-one-out cross validation estimate on the training data set X of a tree-augmented Bayesian network with **Parents**.

5.2. $LazyTAN$ algorithm

A lazy TAN algorithm is described in Table 1. The main loop is a straight-forward translation to lazy learning of Keogh and Pazzani's (1999) variant of the TAN algorithm [7]. First, it makes each node a super parent and the classification accuracy of the corresponding network is evaluated using leave-one-out cross validation. It then finds the best super parent. Next it tries to find the favourite child, by assessing the effect of adding a single arc from the best super parent to each orphan. For one test instance, the complexity of selecting the best super parent is $O(n)$, because there are at the most n Bayesian networks to be evaluated, and the complexity of selecting the favourite child is also $O(n)$, because there are at the most n arcs to be tested.

Each tree-augmented Bayesian network in our lazy algorithm is based on a single given test instance. Thus, an attribute only has two values to be used in the network. One is equal to the value of the test instance, and the other value is not equal to the value of the test instance on this attribute. We don't need to differentiate different values that are not equal to the value on the test instance. This kind of Bayesian network does not reflect the joint probability distributions for all attributes, but reflects the specific dependencies between the attribute values of the current test instance.

6. Experiments on Weka system

We compare the classification performance of four learning algorithms. We use the naive Bayes classifier implemented in the Weka system, simply called Naive. We implemented in Weka a lazy Bayesian rule (LBR) learning algorithm [17], a tree-augmented Bayesian network (TAN) learning algorithm [7], and a lazy tree-augmented Bayesian network learning algorithm (called $LazyTAN$). All the experiments were performed in the Weka system [15], which provides a workbench that includes full and working implementations of many popular learning schemes that can be used for practical data mining or for research.

Thirty-two natural domains are used in the experiments shown in Table 2. Twenty-eight of these are drawn from twenty-nine data sets used in previous research in the area [7, 17]. Splice-junction had to be discarded as computations could not be completed in reasonable time. The original corpus of data sets is dominated by small data sets. As LBR and TAN are directed at larger data sets, the remaining twenty-seven data sets were augmented by a selection of four larger data sets (vehicle, mfeat-mor, sign, segment).

In Table 2, 3, "$S\sharp$" means the number of instances in a data set. "$C\sharp$" means the number of values of a class attribute. "$A\sharp$" means the number of attributes, not including the class attribute. The classification accuracy of each classifier on each domain is obtained by running 10-fold cross validation, and the random seed for 10-fold cross validation takes on the default value in Weka system. We also use the default discretization method "weka.filters.DiscretizeFilter" as the discretization method for continuous values, which is offered by the Weka system. All experimental results for classification accuracies of the algorithms are shown in Table 3.

Table 2. Descriptions of Data

	Domain	S♯	C♯	A♯
1	Annealing Processes	898	6	38
2	Audiology	226	24	69
3	Breast Cancer(Wisconsin)	699	2	9
4	Chess (King-rook-vs-king-pawn)	3196	2	36
5	Credit Screening(Australia)	690	2	15
6	Echocardiogram	74	2	6
7	Glass Identification	214	7	10
8	Heart Disease(Cleveland)	303	2	13
9	Hepatitis Prognosis	155	2	19
10	Horse Colic	368	2	22
11	House Votes 84	435	2	16
12	Hypothyroid Diagnosis	3163	2	25
13	Iris Classification	150	3	4
14	Labor Negotiations	57	2	16
15	LED 24(noise level=10%)	1000	10	24
16	Liver Disorders(bupa)	345	2	6
17	Lung Cancer	32	3	56
18	Lymphography	148	4	18
19	Mfeat-mor	2000	10	6
20	Pima Indians Diabetes	768	2	8
21	Post-Operative Patient	90	3	8
22	Primary Tumor	339	22	17
23	Promoter Gene Sequences	106	2	57
24	Segment	2310	7	19
25	Sign	12546	3	8
26	Solar Flare	1389	2	9
27	Sonar Classification	208	2	60
28	Soybean Large	683	19	35
29	Tic-Tac-Toe End Game	958	2	9
30	Vehicle	846	4	18
31	Wine Recognition	178	3	13
32	Zoology	101	7	16

Tables 4, 5 and 6, present the WIN/LOSS/DRAW records for LBR, TAN and $LazyTAN$, respectively. This is a record of the number of data sets for which the nominated algorithm achieves lower, higher, and equal error to the comparison algorithm, measured to two decimal places. The tables also include the outcome of a two-tailed binomial sign test. This indicates the probability that the observed

outcome or more extreme should occur by chance if wins and losses were equiprobable. LBR and TAN demonstrate comparable levels of error rate.

LBR has a lower error rate than the naive Bayes classifier in fourteen out of the twenty-seven data sets, and a higher error rate in only three data sets. This outcome is statistically significant at the 0.05 level. LBR and TAN demonstrate comparable levels of error rate. LBR has a higher error rate than TAN classifier in thirteen data sets, and lower error rate in ten. $LazyTAN$ also has similar error performance to LBR and TAN, but the $LazyTAN$ algorithm is not so efficient as LBR classifier.

In the thirty-two databases, there are fifteen databases which TAN has higher classification accuracy than Naive Bayes, and eight for which it has lower classification accuracy. On these data sets TAN has failed to improve upon the error of naive Bayes with significant frequency. It is remarkable that the pattern of the databases for which each of TAN and LBR outperforms naive Bayes differs substantially. It is clear that they find different Bayesian networks topologies.

There are nineteen databases which $LazyTAN$ has higher classification accuracy than the naive Bayes classifier, and nine for which it has lower classification accuracy. This result approaches but does not achieve statistical significance at the 0.05 level (although if one-tailed tests were applied, as it could be argued is appropriate in this case as we expected $LazyTAN$ to outperform naive Bayes, then the result would be significant - 0.044). $LazyTAN$ has lower error than TAN for fourteen data sets and higher for eleven. This difference is not statistically significant. $LazyTAN$ has similar error performance to the LBR classifier. These results give some suggestion that directly using lazy learning techniques to build a tree-augmented Bayesian network for a given test instance benefit classification accuracy.

The mean accuracy over all data sets for each algorithm is given in Table 7. Due to the incommensurability of accuracy scores across different data sets this is a gross measure to which it is not wise to pay too much attention. Nonetheless, it might be indicative that LBR achieves the highest mean accuracy.

To further compare LBR and TAN we calculate the geometric mean accuracy ratio. For each data set the accuracy of TAN is divided by the accuracy of LBR. The geometric mean of these values is 0.994. A value above 1.0 favors TAN while a value below 1.0 favours LBR. A value of 0.994 indicates a very slight advantage toward LBR for the data sets studied.

Table 3. Average Error Rate for Each Dataset

	Domain	*Naive*	*LBR*	*TAN*	*LTAN*
1	Annealing	5.46	5.46	4.01	3.45
2	Audiology	29.20	29.20	29.20	28.32
3	B Cancer	2.58	2.58	2.58	2.58
4	Chess	12.36	3.57	5.07	5.07
5	Australia	15.07	14.64	14.35	14.64
6	Echocardiogram	27.48	27.48	28.24	28.24
7	Glass	11.68	9.81	6.07	9.81
8	Cleveland	16.50	16.50	16.50	18.48
9	Hepatitis	16.13	16.13	16.13	14.84
10	Horse Colic	20.11	19.29	18.48	18.48
11	House Votes	9.89	7.13	6.90	9.66
12	Hypothyroid	2.94	2.78	2.88	2.66
13	Iris	6.67	6.67	6.00	4.67
14	Labor	3.51	3.51	3.51	3.51
15	LED	24.50	24.70	24.50	24.60
16	Bupa	36.81	36.81	40.29	37.39
17	L Cancer	46.88	43.75	50.00	46.88
18	Lymphography	14.19	14.19	15.54	12.84
19	Mfeat-mor	30.65	29.95	30.10	29.95
20	PID	25.00	24.87	25.39	26.56
21	Post-Operative	28.89	28.89	30.00	34.44
22	PT	48.97	49.85	49.85	49.85
23	Promoter	8.49	8.49	8.49	9.43
24	Segment	11.08	6.41	6.28	6.54
25	Sign	38.58	20.93	26.85	28.71
26	Solar Flare	18.57	15.69	16.85	15.48
27	Sonar	25.48	25.96	23.56	23.08
28	Soybean	7.17	7.17	7.03	7.32
29	TTT	29.54	14.61	28.81	25.57
30	Vehicle	39.48	31.44	31.68	30.61
31	Wine	3.37	3.37	3.37	2.81
32	Zoology	5.94	5.94	5.94	5.94

Table 4. Comparison of LBR to others

	WIN	*LOSS*	*DRAW*	*P*
Naive	14	3	15	0.013
TAN	13	10	9	0.678
LazyTAN	13	13	6	1.000

Table 5. Comparison of TAN to others

	WIN	$LOSS$	$DRAW$	P
$Naive$	15	8	9	0.210
LBR	10	13	9	0.678
$LazyTAN$	11	14	7	0.690

Table 6. Comparison of $LazyTAN$ to others

	WIN	$LOSS$	$DRAW$	P
$Naive$	19	9	4	0.087
LBR	14	11	7	0.690
TAN	13	13	6	1.000

7. Conclusion

Lazy Bayesian Rules and tree-augmented Bayesian networks are two extensions to naive Bayes that have previously independently been shown to substantially reduce its error. We have presented the first comparative evaluation of these two approaches. Our evaluation suggests that the two algorithms are most effective on quite different data sets. Neither exhibits a general advantage over the other. To evaluate the impact of the lazy/eager differentiation between the two algorithms we also implemented a lazy version of TAN. The comparison indicated that $LazyTAN$ enjoyed at best a modest improvement upon TAN. These outcomes suggest that the most important difference between LBR and TAN is the difference in the topologies of the networks that they construct. Due to its lazy approach, LBR enjoys greater computational efficiency than TAN when few objects are to be classified from a single training set and TAN enjoys a computational advantage when many objects are to be classified.

References

[1] D. W. Aha. Lazy learning: special issue editorial. *Artificial Intelligence Review*, pages 7–10, 11 1997.

[2] J. Cerquides. Applying general Bayesian techniques to improve TAN induction. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining - KDD'1999*, pages 292–296, 1999.

[3] J. Cheng and R. Greiner. Comparing Bayesian network classifiers. In *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence (UAI-99)*, Sweden, 1999.

[4] P. Domingos and M. Pazzani. Beyond independence: conditions for the optimality of the simple Bayesian classifier. In *Proceedings of the Thirteenth International Conference on Machine Learning*, pages 105–112, San Francisco, CA, 1996. Morgan Kaufmann Publishers, Inc.

Table 7. The Mean Error Rate

	$Naive$	TAN	LBR	$LazyTAN$
Error Rate	19.47	18.26	17.43	18.13

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

[6] N. Friedman, R. Kohavi, and Y. Yun. Lazy decision tree. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, pages 717–724, Menlo Park, CA, 1996. The AAAI PressMorgan Kaufmann Publishers, Inc.

[7] E. J. Keogh and M. J. Pazzani. Learning augmented Bayesian classifiers: a comparison of distribution-based and classification-based approaches. In *Proceedings of the Seventh International Workshop on Artificial Intelligence and Statistics*, pages 225–230, 1999.

[8] R. Kohavi. Scaling up the accuracy of naive-Bayes classifiers: a decision-tree hybird. In E. Simoudis, J. W. Han, and U. M. Fayyad, editors, *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, pages 202–207, Menlo Park, CA, 1996. The AAAI Press.

[9] R. Kohavi and G. H. John. Wrappers for feature subset selection. *Artificial Intelligence*, pages 273–324, 1997.

[10] I. Kononenko. Semi-nave Bayesian classifier. In *Proceedings of European Conference on Artificial Intelligence*, pages 206–219, 1991.

[11] P. Langley and S. Sage. Induction of selective Bayesian classifiers. In *Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence*, pages 339–406, Seattle, WA, 1994. Morgan Kaufmann Publishers.

[12] T. M. Mitchell. *Machine Learning*. The McGraw-Hill Companies, Inc., New York, 1997.

[13] M. Pazzani. Constructive induction of Cartesian product attributes. In *Proceedings of Information, Statistics and Induction in Science*, Melbourne, Australia, 1996.

[14] M. Sahami. *Learning limited dependence Bayesian classifiers*, pages 335–338. AAAI Press, Portland, OR, 1996.

[15] I. H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufmann Publishers, Seattle, WA, 2000.

[16] H. Zhang and X. Ling. Learnability of augmented naive Bayes in nominal domains. In *Proceedings of the Eighteenth International Conference on Machine Learning (ICML-2001)*, Williams College, 2001.

[17] Z. Zheng and G. I. Webb. Lazy learning of Bayesian rules. *Machine Learning*, 41(1):53–84, 2000.

[18] Z. Zheng, G. I. Webb, and K. M. Ting. Lazy Bayesian Rules: A lazy semi-naive Bayesian learning technique competitive to boosting decision trees. In *Proceedings of the Sixteenth International Conference on Machine Learning (ICML-99)*, pages 493–502, Bled, Slovenia, 1999. Morgan Kaufmann.