Pre-publication draft of a paper which appeared in Simoff, S.J., G.J. Williams, J. Galloway, and I. Kolyshkina (Eds.), Proceedings of the Fourth Australasian Data Mining Workshop (AusDM05), pages 141-156. Sydney: University of Technology

A Comparative Study of Semi-naive Bayes Methods in Classification Learning

Fei Zheng and Geoffrey I. Webb

Clayton School of Information Technology
Monash University
Melbourne VIC 3800, Austrailia
{feizheng, Geoff.Webb}@infotech.monash.edu.au

Abstract. Numerous techniques have sought to improve the accuracy of Naive Bayes (NB) by alleviating the attribute interdependence problem. This paper summarizes these semi-naive Bayesian methods into two groups: those that apply conventional NB with a new attribute set, and those that alter NB by allowing inter-dependencies between attributes. We review eight typical semi-naive Bayesian learning algorithms and perform error analysis using the bias-variance decomposition on thirty-six natural domains from the UCI Machine Learning Repository. In analysing the results of these experiments we provide general recommendations for selection between methods.

Keywords

Naive Bayes, Semi-naive Bayes, attribute independence assumption, bias-variance decomposition

1 Introduction

Supervised classification is a basic task in data mining, predicting a discrete class label for a previously unseen instance $I = \langle a_1, \ldots, a_n \rangle$ from a labelled training sample, where a_i is the value of the i^{th} attribute A_i. There are numerous approaches to produce classifiers, functions that map an unlabelled instance to a class label, such as decision trees, neural networks and probabilistic methods. The Bayesian classifier [1] is a well known probabilistic induction method. It predicts the class label for I by selecting

$$\underset{c_i}{\operatorname{argmax}} \left(P(c_i \,|\, a_1, \ldots, a_n) \right) \propto \underset{c_i}{\operatorname{argmax}} \left(P(c_i) P(a_1, \ldots, a_n \,|\, c_i) \right), \qquad (1)$$

where $c_i \in \{c_1, \ldots, c_k\}$ is the i^{th} value of the class variable C.

However, accurate estimation of $P(a_1, \ldots, a_n \,|\, c_i)$ is non-trivial. Naive Bayes (NB) [2–4] gets round this problem by making the assumption that the attributes are independent given the class. Although the assumption is unrealistic in many practical scenarios, NB has exhibited competitive accuracy with other learning algorithms, especially in domains without highly related attributes. There are many attempts to explain NB's impressive performance, and to develop techniques that further improve its accuracy by alleviating the attribute interdependence problem [4–19]. Collectively, we call these methods *semi-naive Bayesian methods*. Domingos and Pazzani [20] argue that the interdependence between attributes will not affect NB's accuracy performance, so long as it can generate the correct ranks of conditional probabilities for the classes. However, the success of semi-naive Bayesian methods suggest that weakening the attribute independence assumption is effective.

Gaining a better understanding of the strengths and limitations of different semi-naive Bayesian algorithms motivates our comparative study. In this paper, we broadly classify semi-naive Bayesian algorithms into two groups, and examine eight representative semi-naive Bayesian algorithms, including a detailed time and space complexity analysis. We compare these algorithms on thirty-six natural domains from the UCI Machine Learning Repository [21] by using the bias-variance decomposition [22], a key tool for understanding machine learning algorithms. We also give some general recommendations for selecting appropriate semi-naive Bayesian methods.

2 Naive Bayes (NB)

Naive Bayes (NB) [2–4] simplifies probabilistic induction by making the assumptions that the attributes are independent given the class and all the probability estimations from the training sample are accurate. Hence, NB classifies I by selecting

$$\underset{c_i}{\operatorname{argmax}} \left(P(c_i) \prod_{j=1}^{n} P(a_j \mid c_i) \right) . \tag{2}$$

Due to the independence assumption, NB is simple, and computationally efficient. Although the attribute independence assumption is often violated, previous research [3, 12, 20] has shown that NB behaves well across many domains. As it uses a fixed formula to classify, there is no model selection.

At training time NB generates a one-dimensional table of class probability estimates, indexed by the classes, and a two-dimensional table of conditional attribute-value probability estimates, indexed by the classes and attribute-values. The time complexity of calculating the estimates is $O\big(tn\big)$, where t is the number of training examples. The resulting space complexity is $O\big(knv\big)$, where v is the mean number of values per attribute. At classification time, to classify a single example has time complexity $O\big(kn\big)$ using the tables formed at training time with space complexity $O\big(knv\big)$.

3 Semi-naive Bayes Methods

Previous semi-naive Bayesian methods can be roughly subdivided into two groups. The first group establishes NB with a new attribute set, which can be generated by deleting attributes [4,5,9] and joining attributes [6,9]. The second group adds explicit links between attributes, which represent attribute interdependencies. Sahami [10] introduces the notion of the x-dependence Bayesian classifier, which allows each attribute to depend on the class and at most x other attributes. NB is a 0-dependence classifier, and the methods that add explicit links between attributes can be classified into those that establish 1-dependence classifiers [12,14,19] and those that establish x-dependence classifiers ($x \geq 1$) [8,16]. In addition, these methods can be classified into eager learning methods [4,5,8,9,12,14,19], which perform learning at training time, and lazy learning methods [16], which defer learning until classification time. The following Sections present these methods in more details.

3.1 Backwards Sequential Elimination (BSE) and Forward Sequential Selection (FSS)

In Naive Bayes, all the attributes are utilised for classification. When two attributes are strongly related, NB may overweight the influence from these two attributes, and reduce the influence of the other attributes, which can result in prediction bias. Deleting one of these attributes may have the effect of alleviating the problem.

Backwards Sequential Elimination (BSE) [5] and Forward Sequential Selection (FSS) [4] select a subset of attributes using leave-one-out cross validation error as a selection criterion and establish a NB with these attributes. Starting from the full set of attributes, BSE successively eliminates the attribute whose elimination most improves accuracy, until there is no further accuracy improvement. FSS uses the reverse search direction, that is iteratively adding the attribute whose addition most improves accuracy, starting with the empty set of attributes. The subset of selected attributes is denoted as $Atts = \{A_{g_1}, \ldots, A_{g_h}\}$. Independence is assumed among the resulting attributes given the class. Hence, BSE and FSS classify I by selecting

$$\underset{c_i}{\operatorname{argmax}} \left(P(c_i) \prod_{j=g_1}^{g_h} P(a_j \mid c_i) \right). \tag{3}$$

At training time BSE and FSS generate a one-dimensional table of class probability estimates and a two-dimensional table of conditional attribute-value probability estimates, as NB does. As they perform leave-one-out cross validation to select the subset of attributes, they must also store the training data, with additional space complexity $O\big(tn\big)$. The resulting space complexity is $O\big(tn + knv\big)$. Deleting attributes for BSE and adding attributes for FSS have time complexity of $O\big(tkn^2\big)$, as leave-one-out cross validation will be performed at most $O\big(n^2\big)$ times. They have identical time and space complexity with NB at classification time.

3.2 Backward Sequential Elimination and Joining (BSEJ)

Creating new compound attributes when inter-dependencies between attributes are detected is another approach to relaxing the attribute independence assumption. Semi-naive Bayesian classifier [6] uses exhaustive search to join attribute values iteratively based on a statistical method. However, the experimental results are somewhat disappointing.

Backward Sequential Elimination and Joining (BSEJ) [9] uses predictive accuracy as a merging criterion to create new Cartesian product attributes. The value set of a new compound attribute is the Cartesian product of the value sets of the two original attributes. As well as joining attributes, BSEJ also considers deleting attributes. BSEJ repeatedly joins the pair of attributes or deletes the attribute that most improves predictive accuracy using leave-one-out cross validation. This process terminates if there is no accuracy improvement. The resulting Cartesian product attribute set is denoted as $JoinAtts = \{Join_{g_1}, \ldots, Join_{g_h}\}$. The remaining original attributes that have not been either deleted or joined are indicated as $\{A_{l_1}, \ldots, A_{l_q}\}$. Hence, BSEJ classifies I by selecting

$$\underset{c_i}{\text{argmax}} \left(P(c_i) \prod_{j=g_1}^{g_h} P(join_j \mid c_i) \prod_{r=l_1}^{l_q} P(a_r \mid c_i) \right), \tag{4}$$

where $join_j$ is the value of attribute $Join_j$.

At training time BSEJ generates a one-dimensional table of class probability estimates, a two-dimensional table of conditional attribute-value probability estimates, as NB does. It also generates two-dimensional tables of conditional joined attribute-value probability estimates, indexed by the classes and compound attribute-values. In the worst case, the new Cartesian product attribute has v^n values. Therefore, the space complexity is $O\big(tn + kv^n\big)$. BSEJ considers at most $O\big(n^3\big)$ Cartesian product attributes. The time complexity of joining and deleting attributes is $O\big(tkn^3\big)$. At classification time, to classify a single example has time complexity $O\big(kn\big)$ and space complexity $O\big(kv^n\big)$.

3.3 Tree Augment Naive Bayes (TAN) and SuperParent TAN (SP-TAN)

Friedman et al. [12] compared NB with unrestricted Bayesian networks. The observation that unrestricted Bayesian networks did not usually result in accuracy improvement and sometimes lead to reduction in accuracy motivated them to use an intermediate solution that allows each attribute to depend on at most one non-class attribute, called the parent of the attribute. Based on this representation, they utilised conditional mutual information to efficiently find a maximum spanning tree as a classifier. As each attribute depends on at most one other non-class attribute, TAN is a 1-dependence classifier. The parent of each attribute

A_i is indicated as $\pi(A_i)$. Hence, TAN classifies by selecting

$$\underset{c_i}{\operatorname{argmax}} \left(P(c_i) \prod_{j=1}^{n} P(a_j \mid c_i, \pi(a_j)) \right). \tag{5}$$

At training time TAN generates a one-dimensional table of class probability estimates, and a three-dimensional table of probability estimates for each attribute-value, conditioned by each other attribute-value and each class, with space complexity $O\big(k(nv)^2\big)$. The time complexity of forming the three dimensional probability table is $O\big(tn^2\big)$, as it requires each entry for every combination of the two attribute-values for every instance to be updated. Creating the conditional mutual information matrix requires each pair of attributes, every pairwise combination of their respective values in conjunction with each class to be considered, resulting in time complexity $O\big(kn^2v^2\big)$. The parent function is then generated by establishing a maximal spanning tree, with time complexity $O\big(n^2 \log n\big)$. At classification time, to classify a single example has time complexity $O\big(kn\big)$. The three dimensional conditional probability table formed at training time can be compressed by storing probability estimates for each attribute-value conditioned by the parent selected for that attribute and the class. Hence, the space complexity is $O\big(knv^2\big)$.

SuperParent TAN (SP-TAN) [14], a variant of TAN, uses a different approach to construct the parent function. It uses the same representation as TAN, but utilises leave-one-out cross validation error as a criterion to add a link. The SuperParent is the attribute that is the parent of all the other orphans, the attributes without a non-class parent. There are two steps to add a link: first selecting the best SuperParent that improves accuracy the most, and then selecting the best child of the SuperParent from orphans. SP-TAN stops adding links when there is no accuracy improvement. As TAN and SP-TAN use different criteria to establish the parent function, TAN tends to add $N-1$ links, while SP-TAN may have fewer links between attributes. Another difference is the direction of links. TAN chooses the direction randomly, while SP-TAN makes the direction from SuperParents to their favorite children. SP-TAN also uses Equation 5 to classify an unseen instance.

At training time SP-TAN needs additional space complexity $O\big(tn\big)$ for storing the training data compared with TAN. The time complexity of forming the parent function is $O\big(tkn^3\big)$, as the selection of a single SuperParent is order $O\big(tkn^2\big)$, the selection of the favorite child of the SuperParent is order $O\big(tkn\big)$, and parent selection is performed repeatedly at most $O\big(n\big)$ times. SP-TAN has identical classification time complexity and space complexity to TAN.

3.4 NBTree

NBTree [8] is a hybrid approach combining NB and decision tree learning. It partitions the training data using a tree structure and establishes a local NB in each leaf. It uses 5-fold cross validation accuracy estimate as the splitting criterion. A split is defined to be significant if the relative error reduction is greater than 5% and the splitting node has at least 30 instances. When there is no significant improvement, NBTree stops the growth of the tree. As the number of splitting attributes is greater than or equals one, NBTree is a x-dependence classifier. The classical decision tree predicts the same class for all the instances that reach a leaf. In NBTree, these instances are classified using a local NB in the leaf, which only considers those non-tested attributes. Let $S = \{S_1, \ldots, S_g\}$ be the set of the test attributes on the path leading to the leaf, and let $R = \{R_1, \ldots, R_{n-g}\}$ be the set of the remaining attributes, we have

$$P(C, \mathbf{I}) = P(S)P(C \mid S)P(R \mid C, S) \propto P(C \mid S)P(R \mid C, S). \tag{6}$$

Therefore, NBTree classifies I by selecting

$$\operatorname*{argmax}_{c_i} \left(P(c_i \mid s) \prod_{j=1}^{n-g} P(r_j \mid c_i, s) \right), \tag{7}$$

where s is a value of S and r_j is a value of R_j.

In NBTree, the number of leaves possible is $O\big(t\big)$, and the height of the tree is $O\big(log_v t\big)$. Therefore, there are $O\big(t/v\big)$ internal nodes in the tree. At the root, NBTree performs 5-fold cross validation on each attribute to select the best one to split, time complexity of $O\big(tkn^2\big)$. Less time is required for the other internal nodes. Hence, the time complexity of building the tree is $O\big(t^2kn^2/v\big)$. Each leaf has $O\big(n - log_v t\big)$ attributes and stores a two-dimensional table of conditional attribute-value probability estimates. The space complexity is $O\big(tk(n - log_v t)v\big)$. At classification time, to classify a single example has time complexity $O\big(kn\big)$, and space complexity $O\big(tk(n - log_v t)v\big)$.

3.5 Lazy Bayesian Rules (LBR)

Zheng and Webb [16] developed Lazy Bayesian Rules (LBR), which adopts a lazy approach, and generates a new Bayesian rule for each test example. The antecedent of a Bayesian rule is a conjunction of attribute-value pairs, and the consequent of the rule is a local NB, which uses those attributes that do not appear in the antecedent to classify. LBR stops adding attribute-value pairs into the antecedent if the outcome of a one-tailed pairwise sign test of error difference is not better than 0.05. As the number of the attribute-value pairs in the antecedent is greater than or equals one, LBR is a x-dependence classifier. Let $s = \{s_1, \ldots, s_g\}$ be the set of attribute values in the antecedent, and let $r = \{r_1, \ldots, r_{n-g}\}$ be the set of remaining attribute values, LBR classifies I by selecting

$$\operatorname*{argmax}_{c_i} \left(P(c_i \mid s) \prod_{j=1}^{n-g} P(r_j \mid c_i, s) \right). \tag{8}$$

The Bayesian rule generated by LBR can be described as a branch of a tree built by NBTree. LBR generates a rule for each unseen instance, while NBtree builds a single model according to all the examples in the training data. If examples are not evenly distributed among branches in NBTree, small disjuncts, which cover only few training samples, will result in poor prediction performance. As LBR uses lazy learning, it may avoid this problem. LBR is efficient when few examples are to be classified. However, the computational overhead of LBR may be excessive when large numbers of examples are to be classified.

At training time, the time and space complexity of LBR are $O\big(tn\big)$, as it only stores the training data. At classification time, LBR adds attribute-value pairs to the antecedent with time complexity of $O\big(tkn^3\big)$, as the selection of an attribute-value pair for the antecedent is order $O\big(tkn^2\big)$ and this selection is performed repeatedly until there is no significant improvement on accuracy. The space complexity is $O\big(tn + knv\big)$.

3.6 Averaged One-Dependence Estimators (AODE)

To avoid model selection and retain the efficiency of 1-dependence classifiers, Webb et al. [19] proposed AODE, which averages the predictions of all qualified 1-dependence classifiers. In each 1-dependence classifier, all attributes depend on the class and a single attribute. For any attribute value a_j,

$$P(c_i, I) = P(c_i, a_j)P(I \mid c_i, a_j). \tag{9}$$

This equality holds for every a_j. Therefore,

$$P(c_i, I) = \frac{\sum_{j:1\leq j\leq n \wedge F(a_j)\geq m} P(c_i, a_j)P(I \mid c_i, a_j)}{|\{j : 1 \leq j \leq n \wedge F(a_j) \geq m\}|}, \tag{10}$$

where $F(a_j)$ is the frequency of a_j in the training sample.

AODE classifies by selecting:

$$\underset{c_i}{\operatorname{argmax}} \left(\sum_{j:1\leq j\leq n \wedge F(a_j)\geq m} P(c_i, a_j) \prod_{h=1}^{n} P(a_h \mid c_i, a_j) \right). \tag{11}$$

If $P(a_j)$ is small, the estimate of $P(I|c_i, a_j)$ may be unreliable. Hence, AODE averages models where the frequency of the parent attribute is larger than $m = 30$, a widely used minimum sample size in statistics.

At training time AODE generates a three-dimensional table of probability estimates for each attribute-value, conditioned by each other attribute-value and each class. The resulting space complexity is $O(k(nv)^2)$. Forming this table is of time complexity $O(tn^2)$. Classification requires the tables of probability estimates formed at training time of space complexity $O(k(nv)^2)$. The time complexity of classifying a single example is $O(kn^2)$ as we need to consider each pair of qualified parent and child attribute within each class.

4 Algorithm Comparisons

In this study, we compare eight representative semi-naive algorithms and NB. These semi-naive Bayesian algorithms are BSE, FSS, BSEJ, TAN, SP-TAN, NBTree, LBR and AODE.

4.1 Experimental Domains and Methodology

The thirty-six data sets from the UCI Machine Learning Repository used in our experiments are shown in Table 1. The experiments were performed in the Weka workbench [23] on a dual-processor 1.7 GHz Pentium 4 Linux computer with 2 Gb RAM, and all data were discretized using MDL discretization [24].

Table 1. Data sets

No.	Domain	Case	Att	Class	No.	Domain	Case	Att	Class
1	Adult	48,842	14	2	19	Labor negotiations	57	16	2
2	Annealing	898	38	6	20	LED	1,000	7	10
3	Balance Scale	625	4	3	21	Letter Recognition	20,000	16	26
4	Breast Cancer (Wisconsin)	699	9	2	22	Liver Disorders (bupa)	345	6	2
5	Chess	551	39	2	23	Lung Cancer	32	56	3
6	Credit Screening	690	15	2	24	Mfeat-mor	2,000	6	10
7	Echocardiogram	131	6	2	25	New-Thyroid	215	5	3
8	German	1,000	20	2	26	Pen Digits	10,992	16	10
9	Glass Identification	214	9	3	27	Postoperative Patient	90	8	3
10	Heart	270	13	2	28	Primary Tumor	339	17	22
11	Heart Disease (cleveland)	303	13	2	29	Promoter Gene Sequences	106	57	2
12	Hepatitis	155	19	2	30	Segment	2,310	19	7
13	Horse Colic	368	21	2	31	Sign	12,546	8	3
14	House Votes 84	435	16	2	32	Sonar Classification	208	60	2
15	Hungarian	294	13	2	33	Syncon	600	60	6
16	Hypothyroid	3,163	25	2	34	Tic-Tac-Toe Endgame	958	9	2
17	Ionosphere	351	34	2	35	Vehicle	846	18	4
18	Iris Classification	150	4	3	36	Wine Recognition	178	13	3

As bias-variance decomposition provides a valuable insight into the aspects that affect the performance of a learning algorithm, we use Weka's bias-variance decomposition utility which utilised the experimental method proposed by Kohavi and Wolpert [22] to compare the performance of the nine algorithms. Bias denotes the systematic component of error, and variance describes the component of error that stems from sampling [22]. There is a bias-variance tradeoff such that bias typically increases when variance decreases and vice versa.

In Kohavi and Wolpert's method, the training data are divided into a training pool and a test pool randomly. Each pool contains 50% of the data. 50 local training sets, each containing half of the training pool, are sampled from the training pool. Classifiers are generated from each local training set, which is 25% of the full data set. Bias, variance and error are estimated from the performance of the classifiers on the test set.

4.2 Experimental Results

The mean error, bias and variance across all the thirty-six data sets for the nine algorithms are shown in Table 2, 3 and 4 respectively. The pairwise win/draw/loss

Table 2. Error

No.	Domain	NB	AODE	NBTree	LBR	TAN	SP-TAN	BSEJ	BSE	FSS
1	Adult	0.168	0.152	0.144	0.140	0.147	0.147	0.141	0.146	0.144
2	Annealing	0.082	0.065	0.085	0.064	0.067	0.067	0.070	0.076	0.123
3	Balance Scale	0.303	0.302	0.304	0.302	0.303	0.300	0.301	0.304	0.319
4	Breast Cancer (Wisconsin)	0.030	0.027	0.031	0.030	0.050	0.030	0.030	0.030	0.050
5	Chess	0.143	0.140	0.151	0.141	0.128	0.137	0.133	0.142	0.186
6	Credit Screening	0.171	0.163	0.174	0.172	0.177	0.172	0.172	0.172	0.167
7	Echocardiogram	0.389	0.382	0.388	0.392	0.388	0.388	0.382	0.386	0.389
8	German	0.268	0.262	0.283	0.269	0.277	0.268	0.270	0.269	0.288
9	Glass Identification	0.300	0.299	0.299	0.303	0.300	0.295	0.298	0.300	0.313
10	Heart	0.215	0.216	0.232	0.215	0.236	0.218	0.224	0.221	0.269
11	Heart Disease (cleveland)	0.174	0.176	0.191	0.174	0.176	0.178	0.185	0.188	0.246
12	Hepatitis	0.139	0.140	0.144	0.140	0.143	0.138	0.138	0.140	0.153
13	Horse Colic	0.221	0.219	0.228	0.210	0.213	0.219	0.222	0.218	0.218
14	House Votes 84	0.086	0.054	0.064	0.069	0.068	0.082	0.082	0.083	0.039
15	Hungarian	0.169	0.173	0.176	0.173	0.179	0.172	0.176	0.172	0.196
16	Hypothyroid	0.024	0.021	0.018	0.016	0.025	0.018	0.015	0.015	0.014
17	Ionosphere	0.119	0.102	0.121	0.119	0.099	0.118	0.114	0.113	0.137
18	Iris Claasification	0.058	0.058	0.061	0.058	0.056	0.058	0.057	0.058	0.060
19	Labor negotiations	0.150	0.150	0.151	0.196	0.168	0.154	0.154	0.150	0.249
20	LED	0.255	0.258	0.272	0.257	0.271	0.259	0.265	0.265	0.271
21	Letter Recognition	0.292	0.193	0.238	0.220	0.212	0.210	0.250	0.287	0.288
22	Liver Disorders (bupa)	0.424	0.424	0.424	0.424	0.424	0.424	0.424	0.424	0.424
23	Lung Cancer	0.556	0.556	0.556	0.557	0.562	0.555	0.556	0.550	0.619
24	mfeat-mor	0.317	0.311	0.320	0.313	0.312	0.314	0.322	0.317	0.322
25	New-Thyroid	0.074	0.074	0.077	0.074	0.077	0.075	0.075	0.075	0.108
26	Pen Digits	0.132	0.037	0.071	0.065	0.066	0.055	0.078	0.124	0.125
27	Postoperative Patient	0.366	0.366	0.366	0.364	0.383	0.386	0.380	0.354	0.319
28	Primary Tumor	0.559	0.572	0.603	0.571	0.593	0.571	0.573	0.567	0.649
29	Promoter Gene Sequences	0.130	0.130	0.130	0.132	0.315	0.134	0.134	0.133	0.248
30	Segment	0.112	0.071	0.081	0.092	0.082	0.090	0.090	0.092	0.084
31	Sign	0.362	0.302	0.279	0.280	0.292	0.297	0.287	0.362	0.364
32	Sonar Classification	0.274	0.275	0.286	0.274	0.293	0.279	0.280	0.282	0.301
33	Syncon	0.069	0.059	0.095	0.069	0.058	0.069	0.068	0.068	0.115
34	Tic-Tac-Toe Endgame	0.296	0.261	0.254	0.291	0.294	0.295	0.265	0.294	0.293
35	Vehicle	0.444	0.383	0.375	0.385	0.382	0.428	0.421	0.433	0.420
36	Wine Recognition	0.040	0.042	0.049	0.040	0.053	0.040	0.044	0.043	0.158
	Mean	0.220	0.206	0.214	0.211	0.219	0.212	0.213	0.218	0.241

summary of error, bias and variance for all the algorithms on thirty-six data sets are presented in Table 5, 6 and 7. The win/draw/loss record in each table entry compares the algorithm with which the row is labelled (L) against the algorithm with which the column is labelled (C). The number of wins is the number of data sets for which L achieved a lower mean value for the metric than C. Losses represent higher mean values and draws represent values that are identical for 3 decimal places. The algorithms are sorted in ascending order on the mean metric in each win/draw/loss table. As no specific prediction about relative performance has been made, the p value is the outcome of a two-tailed binomial sign test. We assess a difference as significant if $p \leq 0.05$.

Considering first the error outcomes, AODE achieves the lowest mean error, its mean error being substantially (0.010 or more) lower than that of BSE, TAN, NB and FSS. The mean error of FSS is substantially higher than that of all the other algorithms. The win/draw/loss record indicates that AODE has a significant advantage over all the other algorithms, except LBR and SP-TAN. The advantage of LBR, SP-TAN and BSE is significant compared to NBTree

Table 3. Bias

No.	Domain	NB	AODE	NBTree	LBR	TAN	SP-TAN	BSEJ	BSE	FSS
1	Adult	0.156	0.139	0.123	0.127	0.129	0.119	0.114	0.115	0.122
2	Annealing	0.053	0.045	0.048	0.041	0.043	0.043	0.042	0.046	0.092
3	Balance Scale	0.175	0.172	0.177	0.173	0.172	0.172	0.174	0.172	0.181
4	Breast Cancer (Wisconsin)	0.028	0.025	0.025	0.028	0.027	0.028	0.026	0.026	0.030
5	Chess	0.104	0.101	0.078	0.097	0.062	0.090	0.081	0.095	0.112
6	Credit Screening	0.147	0.138	0.117	0.147	0.130	0.143	0.138	0.172	0.124
7	Echocardiogram	0.249	0.247	0.248	0.253	0.246	0.250	0.248	0.245	0.246
8	German	0.203	0.195	0.183	0.202	0.174	0.196	0.185	0.197	0.217
9	Glass Identification	0.169	0.168	0.160	0.167	0.164	0.166	0.161	0.161	0.153
10	Heart	0.156	0.156	0.153	0.156	0.165	0.154	0.152	0.146	0.143
11	Heart Disease (cleveland)	0.127	0.127	0.119	0.127	0.117	0.126	0.124	0.123	0.124
12	Hepatitis	0.098	0.096	0.082	0.094	0.078	0.095	0.088	0.094	0.083
13	Horse Colic	0.188	0.179	0.158	0.177	0.170	0.183	0.177	0.183	0.174
14	House Votes 84	0.077	0.043	0.028	0.046	0.044	0.071	0.071	0.070	0.028
15	Hungarian	0.156	0.156	0.144	0.157	0.134	0.155	0.151	0.150	0.158
16	Hypothyroid	0.021	0.018	0.012	0.013	0.022	0.014	0.012	0.012	0.013
17	Ionosphere	0.077	0.068	0.070	0.077	0.063	0.076	0.075	0.073	0.075
18	Iris Claasification	0.037	0.037	0.038	0.037	0.034	0.038	0.039	0.039	0.039
19	Labor negotiations	0.046	0.046	0.047	0.068	0.057	0.048	0.045	0.044	0.088
20	LED	0.209	0.211	0.209	0.208	0.221	0.208	0.207	0.211	0.212
21	Letter Recognition	0.230	0.133	0.102	0.103	0.124	0.110	0.142	0.226	0.223
22	Liver Disorders (bupa)	0.292	0.292	0.292	0.292	0.292	0.292	0.292	0.292	0.292
23	Lung Cancer	0.311	0.311	0.312	0.312	0.375	0.309	0.310	0.306	0.311
24	mfeat-mor	0.246	0.240	0.212	0.231	0.235	0.234	0.218	0.227	0.217
25	New-Thyroid	0.039	0.040	0.037	0.039	0.028	0.039	0.039	0.039	0.039
26	Pen Digits	0.111	0.023	0.025	0.025	0.035	0.025	0.045	0.097	0.094
27	Postoperative Patient	0.299	0.299	0.300	0.300	0.315	0.306	0.307	0.298	0.305
28	Primary Tumor	0.346	0.348	0.330	0.352	0.370	0.342	0.330	0.331	0.354
29	Promoter Gene Sequences	0.043	0.043	0.044	0.044	0.134	0.044	0.045	0.048	0.080
30	Segment	0.075	0.044	0.034	0.047	0.039	0.056	0.053	0.055	0.043
31	Sign	0.324	0.260	0.206	0.218	0.245	0.235	0.214	0.310	0.311
32	Sonar Classification	0.181	0.180	0.172	0.181	0.169	0.182	0.172	0.175	0.178
33	Syncon	0.046	0.037	0.027	0.046	0.027	0.046	0.045	0.045	0.033
34	Tic-Tac-Toe Endgame	0.234	0.191	0.107	0.207	0.195	0.199	0.134	0.214	0.192
35	Vehicle	0.315	0.255	0.225	0.248	0.231	0.300	0.299	0.306	0.267
36	Wine Recognition	0.015	0.016	0.014	0.015	0.017	0.016	0.016	0.016	0.063
	Mean	0.155	0.141	0.129	0.140	0.141	0.142	0.138	0.149	0.150

and FSS. All the algorithms, except NB, have a significant advantage over FSS. It is notable that AODE is the only algorithm to have a significant advantage in error over NB.

With respect to bias, NBTree exhibits the lowest mean bias, its mean bias being substantially lower than that of all the remaining algorithms but BSEJ. The win/draw/loss record shows that NBTree has a significant advantage over the other algorithms, except TAN. The advantage of BSEJ is significant compared with SP-TAN and NB. All the algorithms except FSS have significant advantage over NB.

Turning to variance, the mean variance of NB and AODE is substantially lower than that of BSEJ, TAN, NBTree and FSS. The win/draw/loss record indicates that NB has a significant advantage over the other algorithms, but AODE. AODE shares similar levels of variance with NB and LBR, and has a significant advantage over the other algorithms. LBR and SP-TAN have a significant advantage over BSEJ, TAN, NBTree and FSS. The advantage of BSE

Table 4. Variance

No.	Domain	NB	AODE	NBTree	LBR	TAN	SP-TAN	BSEJ	BSE	FSS
1	Adult	0.011	0.012	0.020	0.013	0.018	0.027	0.027	0.031	0.021
2	Annealing	0.028	0.020	0.036	0.023	0.023	0.024	0.027	0.030	0.031
3	Balance Scale	0.125	0.128	0.125	0.126	0.128	0.126	0.125	0.129	0.135
4	Breast Cancer (Wisconsin)	0.002	0.002	0.006	0.002	0.023	0.002	0.004	0.004	0.020
5	Chess	0.038	0.038	0.072	0.043	0.065	0.046	0.051	0.046	0.074
6	Credit Screening	0.024	0.025	0.056	0.024	0.047	0.028	0.034	0.037	0.042
7	Echocardiogram	0.137	0.133	0.138	0.137	0.139	0.135	0.131	0.138	0.140
8	German	0.063	0.066	0.098	0.066	0.101	0.071	0.084	0.071	0.070
9	Glass Identification	0.129	0.129	0.136	0.134	0.134	0.127	0.134	0.136	0.156
10	Heart	0.058	0.058	0.078	0.058	0.070	0.063	0.070	0.074	0.123
11	Heart Disease (cleveland)	0.046	0.048	0.071	0.046	0.059	0.051	0.059	0.063	0.119
12	Hepatitis	0.040	0.043	0.061	0.044	0.064	0.043	0.049	0.045	0.069
13	Horse Colic	0.032	0.040	0.068	0.033	0.042	0.035	0.043	0.034	0.043
14	House Votes 84	0.009	0.010	0.035	0.022	0.024	0.011	0.011	0.013	0.011
15	Hungarian	0.013	0.017	0.032	0.016	0.044	0.016	0.025	0.021	0.038
16	Hypothyroid	0.003	0.003	0.006	0.002	0.003	0.004	0.003	0.003	0.002
17	Ionosphere	0.041	0.033	0.050	0.041	0.036	0.041	0.039	0.039	0.061
18	Iris Claasification	0.021	0.021	0.022	0.021	0.021	0.019	0.018	0.019	0.020
19	Labor negotiations	0.102	0.102	0.102	0.126	0.109	0.104	0.107	0.104	0.157
20	LED	0.045	0.046	0.062	0.048	0.049	0.050	0.057	0.052	0.058
21	Letter Recognition	0.061	0.058	0.133	0.114	0.086	0.098	0.106	0.060	0.064
22	Liver Disorders (bupa)	0.130	0.130	0.130	0.130	0.130	0.130	0.130	0.130	0.130
23	Lung Cancer	0.240	0.240	0.240	0.240	0.184	0.241	0.241	0.239	0.302
24	mfeat-mor	0.070	0.070	0.106	0.081	0.075	0.079	0.101	0.088	0.103
25	New-Thyroid	0.034	0.034	0.038	0.034	0.049	0.035	0.036	0.036	0.068
26	Pen Digits	0.020	0.014	0.045	0.039	0.030	0.029	0.033	0.026	0.031
27	Postoperative Patient	0.065	0.065	0.065	0.064	0.067	0.078	0.071	0.056	0.013
28	Primary Tumor	0.210	0.219	0.268	0.215	0.218	0.225	0.238	0.232	0.290
29	Promoter Gene Sequences	0.085	0.085	0.085	0.086	0.177	0.088	0.088	0.084	0.165
30	Segment	0.036	0.026	0.047	0.044	0.043	0.034	0.036	0.037	0.040
31	Sign	0.037	0.041	0.072	0.060	0.045	0.061	0.072	0.051	0.052
32	Sonar Classification	0.092	0.093	0.112	0.092	0.122	0.095	0.106	0.105	0.120
33	Syncon	0.022	0.022	0.067	0.022	0.030	0.022	0.023	0.023	0.081
34	Tic-Tac-Toe Endgame	0.061	0.068	0.145	0.083	0.097	0.094	0.129	0.079	0.099
35	Vehicle	0.126	0.126	0.147	0.134	0.148	0.125	0.120	0.124	0.149
36	Wine Recognition	0.024	0.025	0.034	0.024	0.036	0.024	0.027	0.026	0.093
	Mean	0.063	0.064	0.083	0.069	0.076	0.069	0.074	0.069	0.089

and BSEJ compared with NBTree and FSS is significant. TAN, NBTree and FSS share similar levels of variance.

4.3 Analysis

Bias describes how closely the learner is able to describe the decision surfaces for a domain, while variance reflects the sensitivity of the learner to variations in the training sample. In general, the better the learner is able to fit the training data, the lower the bias. However, closely fitting the training data may result in greater changes in the model formed from sample to sample, and hence higher variance. There is a tension between bias and variance. However, variance is expected to decrease with increasing training sample size, as the differences between the different samples decrease [25]. Therefore, bias may come to dominate error for problems with large training samples.

NB uses a fixed formula to classify, and hence there is no model selection, which results in relatively low variance. Weakening the attribute independence

Table 5. Win/Draw/Loss Records of Error on 36 Datasets

W/D/L *p* of W/D/L	AODE	LBR	SP-TAN	BSEJ	NBTree	BSE	TAN	NB	FSS
AODE									
LBR	12–6–18								
	0.3616								
SP-TAN	11–3–22	13–7–16							
	0.0802	0.7110							
BSEJ	8–3–25	13–3–20	11–9–16						
	0.0046	0.2962	0.4420						
NBTree	5–5–26	10–1–25	9–3–24	11–3–22					
	0.0002	**0.0166**	**0.0136**	0.0802					
BSE	7–4–25	10–7–19	12–6–18	12–7–17	24–2–10				
	0.0022	0.1360	0.3616	0.4582	**0.0244**				
TAN	8–2–26	12–1–23	13–4–19	13–1–22	15–3–18	15–3–18			
	0.0030	0.0896	0.3770	0.1754	0.7284	0.7284			
NB	8–7–21	11–10–15	11–6–19	15–3–18	21–4–11	13–7–16	17–3–16		
	0.0242	0.5572	0.2004	0.7284	0.1102	0.7110	1.0000		
FSS	5–1–30	6–1–29	9–1–26	7–2–27	7–2–27	8–2–26	7–3–26	11–2–23	
	<0.0001	**0.0002**	**0.0090**	**0.0008**	**0.0008**	**0.0030**	**0.0014**	0.0580	

assumption may make semi-naive Bayesian methods fit the training sample better. Consequently, they may have lower bias, but higher variance compared with NB. AODE reduces variance successfully by aggregating all the qualified 1-dependence classifiers. It delivers competitive variance with NB. NBTree has relatively low bias, but high variance. Brain and Webb [25] hypothesized that the low variance algorithms tend to enjoy lower relative error on small training sets, while low bias algorithms enjoy lower relative error on large training sets. Therefore, the Weka bias-variance estimation method used in this study, which produces small training sets, might put NBTree at a disadvantage. We believe that this also accounts for why AODE was the only algorithm to achieve a significant advantage over NB with respect to error in our experiments, given the low variance of these two algorithms.

As has been discussed above, bias tends to dominate error for large training samples. Therefore, for large training data we recommend use of the lowest bias semi-naive Bayesian method whose complexity satisfies the computational constraints of the application context. For small training data we recommend the lowest variance semi-naive Bayesian method that has suitable computational complexity. For intermediate size training samples, an appropriate trade-off between bias and variance should be sought within the prevailing computational complexity constraints. AODE has very low variance, relatively low bias, and low training time and space complexity. In consequence, it may prove competitive over a considerable range of classification tasks. For extremely small data

Table 6. Win/Draw/Loss Records of Bias on 36 Datasets

W/D/L *p* of W/D/L	NBTree	BSEJ	LBR	AODE	TAN	SP-TAN	BSE	FSS	NB
NBTree									
BSEJ	7–5–24								
	0.0034								
LBR	4–5–27	11–3–22							
	<0.0001	0.0814							
AODE	10–2–24	13–3–20	18–4–14						
	0.0244	0.2962	0.5966						
TAN	12–2–22	19–1–16	21–1–14	21–2–13					
	0.1214	0.7358	0.3106	0.2294					
SP-TAN	5–4–27	7–4–25	15–7–14	16–3–17	14–3–19				
	0.0001	**0.0022**	1.0000	1.0000	0.4868				
BSE	8–2–26	10–8–18	19–3–14	16–4–16	14–2–20	19–5–12			
	0.0030	0.1850	0.4868	1.2734	0.3916	0.2810			
FSS	5–2–29	12–5–19	16–3–17	15–2–19	13–2–21	17–2–17	13–3–20		
	<0.0001	0.2810	1.0000	0.6076	0.1754	1.2734	0.2962		
NB	6–2–28	4–2–30	7–11–18	4–9–23	9–1–26	7–4–25	5–2–29	13–3–20	
	0.0002	**<0.0001**	**0.0432**	**0.0004**	**0.0060**	**0.0022**	**<0.0001**	0.2962	

NB may prove better and for large data NBTree, BSEJ and LBR may have an advantage if their computational profiles are appropriate to the task.

Admittedly these guidelines are imprecise, as the relevant data size is relative to the complexity of the decision surfaces that must be approximated, and in most applications this is unknown. Nonetheless, we believe that they provide a useful framework within which to operate when choosing between semi-naive Bayesian methods.

5 Conclusion

A number of techniques have developed to improve Naive Bayes's accuracy performance by relaxing the attribute independence assumption. We study eight typical semi-naive Bayesian algorithms, and give details of the time and space complexity of these methods. BSEJ, NBTree and SP-TAN have relatively high training time complexity, while LBR has high classification time complexity. BSEJ has very high space complexity. We performed extensive experimental evaluation of the relative error, bias and variance of these algorithms. For the experimental data sets investigated, AODE shares similar levels of error with LBR and SP-TAN, and has a significant advantage over the other algorithms. NBTree has a significant advantage over all the other algorithms, except TAN. All the other algorithms, except TAN and FSS have a significant advantage over NBTree. As bias tends to be a larger portion of error when training set size

Table 7. Win/Draw/Loss Records of Variance on 36 Datasets

W/D/L *p* of W/D/L	NB	AODE	LBR	SP-TAN	BSE	BSEJ	TAN	NBTree	FSS
NB									
AODE	6–15–15								
	0.0784								
LBR	3–13–20	10–8–18							
	0.0004	0.1850							
SP-TAN	6–5–25	7–4–25	11–7–18						
	0.0008	**0.0022**	0.2650						
BSE	7–2–27	6–2–28	13–1–22	11–5–20					
	0.0008	**0.0002**	0.1754	0.1496					
BSEJ	5–4–27	4–2–30	10–2–24	7–5–24	12–6–18				
	0.0001	**<0.0001**	**0.0244**	**0.0034**	0.3636				
TAN	3–3–30	2–4–30	8–4–24	11–1–24	12–2–22	13–5–18			
	<0.0001	**<0.0001**	**0.0070**	**0.0410**	0.1214	0.4732			
NBTree	0–6–30	1–5–30	3–2–31	6–1–29	3–3–30	5–3–28	13–1–22		
	<0.0001	**<0.0001**	**<0.0001**	**0.0001**	**<0.0001**	**<0.0001**	0.1754		
FSS	3–1–32	3–1–32	7–2–27	6–2–28	5–1–30	8–3–25	12–1–23	15–1–20	
	<0.0001	**<0.0001**	**0.0008**	**0.0002**	**<0.0001**	**0.0046**	0.0896	0.4996	

increases, we suggest using low bias methods for large data sets, and low variance methods for small data sets, within the further constraints on applicable algorithms implied by the computational constraints of the given application. Computation cost and the trade-off between bias and variance should be considered for intermediate size data.

References

1. Duda, R.O., Hart, P.E.: Pattern classification and scene analysis. John Wiley and Sons, New York (1973)
2. Kononenko, I.: Comparison of inductive and naive Bayesian learning approaches to autpmatic knowledge acquisition. In Wielinga, B., Boose, J., B.Gaines, Schreiber, G., van Someren, M., eds.: Current Trends in Knowledge Acquisition. Amsterdam: IOS Press (1990)
3. Langley, P., Iba, W., Thompson, K.: An analysis of Bayesian classifiers. In: Proc. 10th Nat. Conf. Artificial Intelligence, AAAI Press and MIT Press (1992) 223–228
4. Langley, P., Sage, S.: Induction of selective Bayesian classifiers. In: Proc. Tenth Conf. Uncertainty in Artificial Intelligence, Morgan Kaufmann (1994) 399–406
5. Kittler, J.: Feature selection and extraction. In Young, T.Y., Fu, K.S., eds.: Handbook of Pattern Recognition and Image Processing. Academic Press, New York (1986)
6. Kononenko, I.: Semi-naive Bayesian classifier. In: Proc. 6th European Working Session on Machine Learning, Berlin: Springer-Verlag (1991) 206–219

7. Langley, P.: Induction of recursive Bayesian classifiers. In: Proc. 1993 European Conf. Machine Learning, Berlin: Springer-Verlag (1993) 153–164
8. Kohavi, R.: Scaling up the accuracy of naive-Bayes classifiers: a decision-tree hybrid. In: Proc. 2nd ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining. (1996) 202–207
9. Pazzani, M.J.: Constructive induction of Cartesian product attributes. ISIS: Information, Statistics and Induction in Science (1996) 66–77
10. Sahami, M.: Learning limited dependence Bayesian classifiers. In: Proc. 2nd Int. Conf. Knowledge Discovery in Databases, Menlo Park, CA: AAAI Press (1996) 334–338
11. Singh, M., Provan, G.M.: Efficient learning of selective Bayesian network classifiers. In: Proc. 13th Int. Conf. Machine Learning, Morgan Kaufmann (1996) 453–461
12. Friedman, N., Geiger, D., Goldszmidt, M.: Bayesian network classifiers. Machine Learning **29** (1997) 131–163
13. Webb, G.I., Pazzani, M.J.: Adjusted probability naive Bayesian induction. In: Proc. 11th Australian Joint Conf. Artificial Intelligence, Berlin:Springer (1998) 285–295
14. Keogh, E.J., Pazzani, M.J.: Learning augmented Bayesian classifers: A comparison of distribution-based and classification-based approaches. In: Proc. Int. Workshop on Artificial Intelligence and Statistics. (1999) 225–230
15. Zheng, Z., Webb, G.I., Ting, K.M.: Lazy Bayesian rules: A lazy semi-naive Bayesian learning technique competitive to boosting decision trees. In: Proc. Sixteenth Int. Conf. Machine Learning (ICML 1999), Morgan Kaufmann (1999) 493–502
16. Zheng, Z., Webb, G.I.: Lazy learning of Bayesian rules. Machine Learning **41** (2000) 53–84
17. Webb, G.I.: Candidate elimination criteria for lazy Bayesian rules. In: Proc. Fourteenth Australian Joint Conf. Artificial Intelligence. Volume 2256., Berlin:Springer (2001) 545–556
18. Xie, Z., Hsu, W., Liu, Z., Lee, M.L.: Snnb: A selective neighborhood based naive Bayes for lazy learning. In: Advances in Knowledge Discovery and Data Mining, Proc. Pacific-Asia Conference (PAKDD 2002), Berlin:Springer (2002) 104–114
19. Webb, G.I., Boughton, J., Wang, Z.: Not so naive Bayes: Aggregating one-dependence estimators. Machine Learning **58** (2005) 5–24
20. Domingos, P., Pazzani, M.J.: Beyond independence: Conditions for the optimality of the simple Bayesian classifier. In: Proc. 13th Int. Conf. Machine Learning, Morgan Kaufmann (1996) 105–112
21. Newman, D.J., Hettich, S., Blake, C.L., Merz, C.J.: UCI repository of machine learning databases (1998) [http://www.ics.uci.edu/∼mlearn/MLRepository.html]. Irvine, CA: University of California, Dept. Information and Computer Science.
22. Kohavi, R., Wolpert, D.: Bias plus variance decomposition for zero-one loss functions. In: Proc. 13th Int. Conf. Machine Learning, San Francisco: Morgan Kaufmann (1996) 275–283
23. Witten, I.H., Frank, E.: Data mining : practical machine learning tools and techniques with Java implementations. San Francisco, CA: Morgan Kaufmann (2000)
24. Fayyad, U.M., Irani, K.B.: Multi-interval discretization of continuous-valued attributes for classification learning. In: Proc. 13th Int. Joint Conf. Artificial Intelligence (IJCAI-93), Morgan Kaufmann (1993) 1022–1029
25. Brain, D., Webb, G.I.: The need for low bias algorithms in classification learning from large data sets. In: Proc. 16th European Conf. Principles of Data Mining and Knowledge Discovery (PKDD2002), Berlin:Springer-Verlag (2002) 62–73