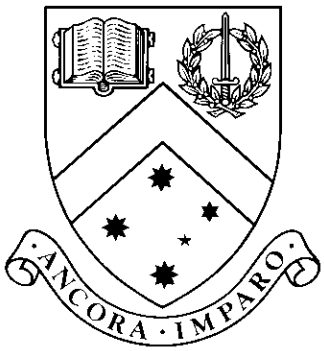
Semi-naive Bayesian Classification

by

Fei Zheng, MCompSc



Thesis

Submitted for fulfillment of the Requirements for the Degree of

Doctor of Philosophy

Supervisor: Professor Geoffrey I. Webb

Clayton School of Information Technology
Monash University

July, 2008

Semi-naive Bayesian Classification

Declaration

I declare that this thesis is my own work and has not been submitted in any form for another degree or diploma at any university or other institute of tertiary education. Information derived from the published and unpublished work of others has been acknowledged in the text and a list of references is given.

Fei Zheng
July 6, 2008

Acknowledgments

Foremost, I would like to thank my supervisor, Professor Geoffrey I. Webb, whose guidance, encouragement and prompt help have accompanied me throughout this very important stage of my life. While Geoff has always allowed me freedom in pursuing my work, this thesis owes enormously to his inspirations and supervisions. I am always envious of his expertise, research insight and enthusiasm and have him as a role model of a successful researcher.

I am grateful to the Clayton School of Information Technology, Monash University. It has provided me not only with much appreciated financial support but also an intellectually stimulating environment.

I would like to thank members of the Knowledge Systems Laboratory and my fellow graduate students: Michelle Kinsman, Ying Yang, Janice Boughton, Khalid Mahmood, Bin Liu, Shane Butler, Alex Sim, Haorianto Tjioe, Jingli Lu, Sattar Hashemi, Shiying Huang, Xiaoya Lin, Zhihai Wang and Yingying Wen. I have enjoyed the collaboration and discussion with them. Janice Boughton also deserves thanks for proofreading parts of this thesis.

I am also grateful to João Gama for kindly providing the source code of Iterative Bayes and to Jesús Cerquides for the executable file of MAPLMG. Thanks to Janez Demšar for his help with the Nemenyi test.

Special thanks must also go to my parents, brother, sisters and friends. They have provided unconditional support and encouragement over my years of study. Finally, I would especially like to thank my partner Ping, who spoiled me, supported me, encouraged me and loves me. This thesis is dedicated to them.

Fei Zheng

Monash University
July 2008

Publications

Refereed Journal Papers

Zheng, F and Webb, G. I.. Subsumption Resolution for Probability Estimates, *Machine Learning*, submitted (Chapters 5 and 6).

Zheng, F and Webb, G. I.. Semi-naive Bayesian Classification: A Survey, *Journal of Machine Learning Research*, resubmission encouraged (Chapter 3).

Refereed Conference Papers

Zheng, F and Webb, G. I. (2007). Finding the right family: Parent and child selection for averaged one-dependence estimators, *Proceedings of the Eighteenth European Conference on Machine Learning (ECML 2007)*, Springer Berlin/Heidelberg, pp. 490-501 (Chapter 4).

Zheng, F and Webb, G. I. (2006). Efficient lazy elimination for averaged-one dependence estimators, *Proceedings of the Twenty-third International Conference on Machine Learning (ICML 2006)*, ACM Press, pp. 1113-1120 (Chapter 5).

Zheng, F and Webb, G. I. (2005). A comparative study of semi-naive Bayes methods in classification learning, *Proceedings of the Fourth Australasian Data Mining Conference (AusDM 2005)*, pp. 141-156 (Chapter 3).

Abstract

The success and popularity of naive Bayes has led to a field of research exploring algorithms that seek to retain its numerous strengths while reducing error by alleviating the attribute interdependence problem. This thesis builds upon this promising field of research, contributing a systematic survey and several novel and effective techniques.

It starts with a study of the strengths and weaknesses of previous *semi-naive Bayesian methods*, providing a taxonomy of them and comparative analysis of their features. Twelve key semi-naive Bayesian methods are benchmarked using error analysis based on the bias-variance decomposition, probabilistic prediction analysis based on the quadratic loss function and training and classification time analysis on sixty natural domains from the UCI Machine Learning Repository. Results for logistic regression and LibSVM, a popular SVM implementation, are also presented to provide a baseline for comparison. In analyzing results of these experiments, we offer general recommendations for selection between semi-naive Bayesian methods based on the characteristics of the application to which they are applied.

This comparative study supports previous findings of strong performance from Averaged One-Dependence Estimators (AODE), which significantly reduces naive Bayes' error with modest training and classification time overheads. Backward Sequential Elimination is an effective wrapper technique to identify and repair harmful interdependencies, and has been profitably applied to naive Bayes. It is therefore surprising that its straightforward application to AODE has previously proved ineffective. In response to this observation, this thesis explores novel variants of this strategy leading to effective techniques. These eliminate child attributes from within the constituent One-Dependence Estimators, thereby significantly improving AODE's prediction accuracy. However, due to repeated accuracy evaluation of attribute subsets on AODE, these elimination techniques have very high training time complexity. In response to this drawback, a new type of semi-naive Bayesian operation, Subsumption Resolution (SR), is proposed. It efficiently identifies pairs of attribute values such that one is a generalization of the other and deletes the generalization at classification time. This adjustment is proved to be theoretically correct for such an interdependence relationship. The thesis demonstrates experimentally that SR can in practice significantly

improve both classification accuracy and the precision of conditional probability estimates. When applied to AODE, SR achieves classification and probability estimation accuracy competitive to state-of-the-art semi-naive Bayesian methods without undue time complexity and may prove desirable over a considerable range of classification tasks. In addition, SR is suited to incremental and semi-supervised learning. This thesis also explores circumstances under which elimination of near-generalizations proves beneficial.

Contents

List of Tables

List of Figures

List of Abbreviations

Abbreviation	Term
AODE	Averaged One-Dependence Estimators
APNB	Adjusted Probability Naive Bayesian Classification
BFGS	Broyden-Fletcher-Goldfarb-Shanno
BSE	Backward Sequential Elimination
BSEJ	Backward Sequential Elimination and Joining
CA	Child Addition
CD	Critical Difference
CE	Child Elimination
CSSB	Continue Search and Select Best
DAG	Directed Acyclic Graph
FSS	Forward Sequential Selection
IB	Iterative Bayes
LBR	Lazy Bayesian Rules
LibSVM	Library for Support Vector Machines
LMG	Linear Mixture of Generative Distributions
LWNB	Locally Weighted Naive Bayes
MAPLMG	Maximum a Posteriori Linear Mixture of Generative Distributions
MDL	Minimum Description Length
NB	Naive Bayes

Abbreviation	Term
NSR	Near-Subsumption Resolution
ODE	One-Dependence Estimator
PA	Parent Addition
PE	Parent Elimination
P∧CA	Parent and Child Addition
P∧CE	Parent and Child Elimination
P∨CA	Parent or Child Addition
P∨CE	Parent or Child Elimination
RBC	Recursive Bayesian Classifiers
RMSE	Root Mean Squared Error
SFN	Stop on First Nonimprovement
SFR	Stop on First Reduction
SPODE	SuperParent One-Dependence Estimator
SP-TAN	SuperParent Tree Augmented Naive Bayes
SR	Subsumption Resolution
SSSR	Semi-Supervised Subsumption Resolution
SVM	Support Vector Machines
TAN	Tree Augmented Naive Bayes
AODE^{BSE}	AODE with BSE
AODE^{SR}	AODE with SR
AODE_r^{SR}	AODE with NSR
AODE^{SSSR}	AODE with SSSR
NB^{BSE}	NB with BSE
NB^{SR}	NB with SR
NB_r^{SR}	NB with NSR
NB^{SSSR}	NB with SSSR

Chapter 1

Introduction

In this information age, data is accumulated and stored at an impressive pace. One tremendous challenge we are facing is to convert large quantities of data into useful information and knowledge, a process called *data mining* [Witten and Frank, 2005]. *Machine learning* is a key for data mining. It provides powerful tools for identifying patterns and trends in data and generalizing them into compact forms, usually referred to as *models*. These models help people to interpret existing data and make predictions for future data.

One important task of machine learning and data mining is *supervised classification learning*, in which a *class label* is associated with each observed example and the goal of the learning system is to predict class labels for unlabeled examples. There are numerous approaches to this task, among which naive Bayes (NB) [Kononenko, 1990; Langley, Iba and Thompson, 1992; Langley and Sage, 1994] may be the simplest and most computationally efficient. This approach is referred to as "naive" in that it makes the assumption of independence between attributes given the class. Due to its simplicity, efficiency and remarkably high classification accuracy in many application domains, for example, medical diagnosis and text classification [Domingos and Pazzani, 1996; Mitchell, 1997; Lewis, 1998], NB has attracted considerable interest.

Improvements to the accuracy of a computationally efficient method is of substantial practical value. Over the years, many refinements to NB have been proposed. This thesis builds upon this established body of work, analyzing the strengths and

weaknesses of previous techniques and utilizing that analysis to propose new algorithms that enhance NB's accuracy while retaining its attractive strengths. The following three sections present the motivations, contributions and structure of this thesis.

1.1 Motivations

One factor that contributes to the success and popularity of NB is its simplicity. It makes a "naive" assumption that attributes are independent given the class and hence dramatically reduces the computational overheads for estimating the posterior probability. A fixed structure in which all attributes only depend on the class is used to perform classification, consequently, there is no model selection. To learn from new examples, it only needs to update a small number of frequency counts. This characteristic, referred to as *incrementality*, is especially advantageous for data mining applications where databases grow at a rapid pace. NB performs optimal classification save only in so far as there are

- violations of its attribute conditional independence assumption; and
- inaccuracies in the estimation of the base probabilities from the training data.

However, this attribute independence assumption is frequently unrealistic in the real world. Numerous techniques have sought to enhance the accuracy of NB by relaxing the assumption. We call these methods *semi-naive Bayesian* methods.

Most semi-naive Bayesian methods are well-founded and tested, but they have previously compared only to the base learner, NB, and a small number of other semi-naive Bayesian methods. In consequence, the relative performance of those methods is not clear, which illuminates the need for a comparative study of semi-naive Bayesian methods. This leads to **the first objective of this thesis**, which is to study the strengths and weaknesses of previous semi-naive Bayesian methods and offer general suggestions for selection between these methods.

To achieve this objective, we examine twelve key semi-naive Bayesian methods and empirically evaluate them with respect to accuracy and computational overheads. We noticed that most algorithms that relax the attribute independence assumption without modifying NB's structure use a wrapper approach (refer to Section 2.5.2.2) to

identify and remove harmful interdependencies, a process with high execution time. Kononenko (1991) made an attempt to use a statistical method to identify correlations between attributes and join strongly related attribute values. However, the reported experimental results were not compelling. The question therefore arises as to whether there are approaches, other than wrapper approach, that can efficiently detect and repair harmful interdependencies while maintaining NB's structure. Due to a variety of relationships between attributes, we are especially interested in exploring methods for detection and removal of some common interdependencies, which leads to **the second objective of this thesis**.

Our extensive experimental results show that five semi-naive Bayesian methods achieve significant accuracy improvement and there is no statistically significant accuracy difference between them. In our experiments, these methods can be largely classified into three groups (Figure 1.1). The first group significantly improves NB's

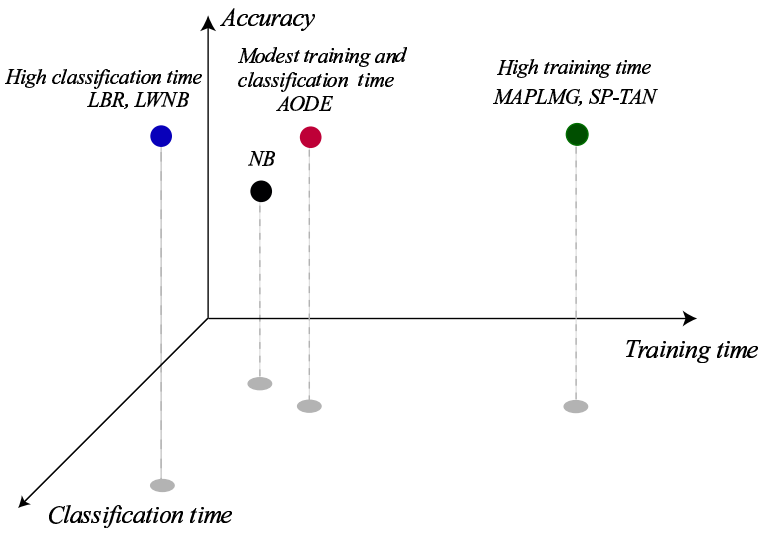


Figure 1.1: Significant accuracy improvement without undue computational overheads is desirable.

accuracy at the cost of considerable increase in *training time*, which is the time to build a model. The second group significantly enhances NB's accuracy with substantially increased *classification time*, which is the time to classify a new example. The third group significantly improves NB's accuracy without undue training and classification time. Methods with high computational cost usually do not scale well to large

data sets, which are very common in data mining applications. The combination of high accuracy and modest computational cost makes the third group a desirable option over a wide range of applications.

The third objective of this thesis is to develop efficient algorithms that further improve classification and conditional probability estimation accuracy of NB and compete favorably with state-of-the-art semi-naive Bayesian methods. More specifically, we are devoted to improving the accuracy and probabilistic prediction of Averaged One-Dependence Estimators (AODE) (refer to Section 3.2.2.4), the method in the third group, without increasing its computational complexity and interfering with its capacity for incremental learning.

1.2 Thesis Contributions

The original contributions to machine learning research are summarized as follows.

- Analysis of previous research

    1. Detailed time and space complexity analysis for twelve existing semi-naive Bayesian methods (Chapter 3).
    2. A comprehensive comparison of NB and twelve semi-naive Bayesian methods using error analysis based on the bias-variance decomposition, probabilistic prediction analysis based on the quadratic loss function and training and classification time analysis on sixty natural domains from the UCI Machine Learning Repository (Chapter 3).
    3. A comprehensive comparison of NB, five semi-naive Bayesian methods that in our experiments significantly improve NB's accuracy, logistic regression and LibSVM using error analysis based on the bias-variance decomposition on fifty-eight natural domains from the UCI Machine Learning Repository (Chapter 3).
    4. A taxonomy of semi-naive Bayesian techniques (Chapter 3).
    5. Recommendations for selection between semi-naive Bayesian methods based on the characteristics of the application to which they are applied (Chapter 3).

- Theoretical analysis

    1. Definitions for three extreme types of interdependencies between attributes: the generalization, substitution and duplication relationships, analysis for relationships between them and between surjection and generalization, bijection and substitution (Chapter 5), and a definition for near-generalization relationship (Chapter 6).
    2. The theorem that deletion of generalizations is a theoretically correct adjustment for the generalization relationship (Chapter 5).

- New semi-naive Bayesian learning algorithms and techniques

    1. A new technique, Subsumption Resolution (SR), for efficiently identifying occurrences of the generalization relationship and removing generalizations at classification time (Chapter 5).
    2. Empirical evidence that the application of SR to NB significantly improves both classification and conditional probability estimation accuracy at the cost of considerable increase in computing time (Chapter 5).
    3. Empirical evidence that the application of SR to AODE significantly improves both classification and conditional probability estimation accuracy without undue computation (Chapter 5).
    4. Empirical evidence that AODE with SR competes favorably with state-of-the-art semi-naive Bayesian methods (Chapter 5).
    5. A new technique, Semi-Supervised Subsumption Resolution (SSSR), for using both labeled and unlabeled data to efficiently identify occurrences of the generalization relationship and remove generalizations at classification time (Chapter 5).
    6. Empirical evidence that SSSR can substantially reduce the variance of SR (Chapter 5).
    7. An investigation into the circumstances in which elimination of near-generalizations often proves profitable (Chapter 6).
    8. The finding that child selection might have greater effect than parent selection in the context of AODE (Chapter 4).

9. An analysis of the reason why previous approaches to attribute selection in AODE have failed to significantly reduce error (Chapter 4).
10. Empirical evidence that AODE's accuracy can be significantly improved by the addition of child elimination with a statistical test (Chapter 4).

1.3 Thesis Organization

The rest of this thesis is organized as follows.

- Chapter 2 lays a foundation by introducing the fundamental terms and concepts of supervised classification learning and by reviewing three classification algorithms which will be compared in the following chapter.
- Chapter 3 reviews the literature on semi-naive Bayesian classification learning and conducts comparative analysis of twelve semi-naive Bayesian methods in terms of bias, variance, error, root mean squared error, training and classification time on sixty natural domains from the UCI Machine Learning Repository. In order to provide a baseline for comparison, it also presents results for logistic regression and LibSVM. Finally, it provides general recommendations for selection between semi-naive Bayesian methods.
- Chapter 4 investigates why the straightforward application of attribute selection to AODE has proved ineffective and develops novel attribute selection algorithms that do prove effective when applied to AODE.
- Chapter 5 formally defines three extreme types of interdependencies between attributes: the generalization, substitution and duplication relationships and discusses relationships between them. A new technique, Subsumption Resolution (SR), is proposed to efficiently identify and remove generalizations at classification time. The effect of SR on NB and AODE is evaluated and the resulting classifiers are compared to other state-of-the-art semi-naive Bayesian methods. Semi-Supervised Subsumption Resolution (SSSR), a variant of SR, is proposed to reduce the variance of SR.

- Chapter 6 extends Subsumption Resolution to Near-Subsumption Resolution and studies the circumstances in which deletion of near-generalizations often proves advantageous.
- Chapter 7 summarizes the major conclusions presented in this thesis and provides suggestions for future work.

Chapter 2

Supervised Classification Learning

This chapter provides background knowledge that is required by the thesis. First, it formally defines the supervised classification problem and the notation used throughout the thesis. Evaluation metrics for classification learning algorithms are briefly introduced in Section 2.2 and special attention is given to accuracy, including the bias and variance tradeoff, and computational complexity. Section 2.3 describes two frequently used methods to evaluate probabilistic prediction, which provide a useful adjunct to classification accuracy for algorithms that can also produce probability estimates. Discretization is briefly introduced in Section 2.4 and attribute selection in Section 2.5. Section 2.6 reviews three classification algorithms that will be compared in the next chapter. Finally, Section 2.7 provides a summary of this chapter.

2.1 Terminology and Notation

We begin by introducing the terminology and notation to be used in the rest of the thesis. The notation follows that of Webb, Boughton and Wang (2005). Let X_i be an *attribute* (also called *a feature*) which takes values from a set $V(X_i)$. There are two types of attributes. An attribute is *qualitative* (or *categorical*) if it can be classified into distinct categories. An attribute is *quantitative* (or *numeric*) if it has a numerical form and to which arithmetic operations can be applied [Bluman, 1997; Yang, 2003]. Let $\mathbf{X} = \langle X_1, \ldots, X_n \rangle$ be the instance variable. An *unlabeled instance* (also called *example*), denoted by $\mathbf{x} = \langle x_1, \ldots, x_n \rangle$, is a point in the instance space $V(X_1) \times \cdots \times V(X_n)$. Let $V(Y) = \{c_1, \ldots, c_k\}$ be the domain of the class variable Y. A class label

$y \in V(Y)$ is a value of Y. A *labeled instance* $\mathbf{x}' = \langle x_1, \ldots, x_n, y \rangle$ is a point in the instance space $V(X_1) \times \cdots \times V(X_n) \times V(Y)$.

A *training set* $T = \{\mathbf{x}'_1, \ldots, \mathbf{x}'_t\}$ is the set of labeled instances (also called *training instances*). A *classifier*, denoted as $\mathcal{C}$, is a function that maps an unlabeled instance $\mathbf{x}$ to a class label y. *Supervised classification learning* is the process to build a classifier from a training set such that the classifier can predict a class label for a new instance.

Given a set U, we denote the cardinality of U by $|U|$. We use v_i to indicate $|V(X_i)|$ and $v = \frac{1}{n}\sum_{i=1}^{n} |V(X_i)|$ to indicate the mean number of values per attribute. Table 2.1 summarizes the notation introduced in this section.

Table 2.1: Notation

Notation	Description
X_i	The ith attribute
x_i	The value of X_i
$\mathbf{X}$	The instance variable
$V(Z)$	The domain of variable Z
$\mathbf{x}$	An unlabeled instance
$\mathbf{x}'$	A labeled instance
Y	The class variable
y	A class label
$\|U\|$	The number of elements in U
T	The training set
$\mathcal{C}$	A classifier
n	The number of attributes
k	The number of classes
t	The number of training instances
v_i	The number of values of attribute X_i
v	The mean number of values per attribute

2.2 Evaluation Metrics for Classification Learning Algorithms

Learning algorithms can be examined from different perspectives: accuracy, computational complexity, incrementality, interpretability, scalability and robustness [Hilario and Kalousis, 1999; Hastie, Tibshirani and Friedman, 2001]. Informally, *accuracy* describes how well the classifier generated from a learning algorithm performs the classification task. *Computational complexity* indicates the efficiency of an algorithm. We say that an algorithm is *incremental* if it does not need to reprocess all earlier training instances when new instances become available. *Interpretability* refers to whether an user can easily understand the output of an algorithm. *Scalability* describes the behavior of an algorithm relative to the training sample size and hence is closely related to computational complexity. *Robustness* reflects the sensitivity of an algorithm to variations in data characteristics, including noise, missing, irrelevant and redundant values. Among these metrics, accuracy and computational complexity are most frequently used in evaluating learning algorithms.

2.2.1 Accuracy

We say that an instance is correctly classified if the prediction agrees with the actual class for that instance and incorrectly if it does not. The accuracy of an algorithm is the expected percentage of correctly classified examples that are randomly drawn from the population [Kohavi, 1995; Mitchell, 1997; Witten and Frank, 2005]. In other words, the accuracy of an algorithm is the probability that an instance randomly drawn from the population ($\langle \mathbf{x}, y \rangle$) can be correctly classified by the classifier $\mathcal{C}$ established by the algorithm:

$$accuracy = P(\mathcal{C}(\mathbf{x}) = y).$$

The *error* (or *zero-one loss*) of an algorithm is the probability that the classifier $\mathcal{C}$ misclassifies a randomly drawn instance. That is, the error is one minus the accuracy:

$$\begin{aligned} error &= P(\mathcal{C}(\mathbf{x}) \neq y) \\ &= 1 - accuracy. \end{aligned}$$

2.2.1.1 Methods for Accuracy Estimation

The accuracy on the training set, called the *resubstitution accuracy*, is an overoptimistic estimate of accuracy as the classifier has been learned from the same training set, and not a good indication for the future performance of an algorithm. One natural approach to overcoming this problem is to estimate accuracy on an independent set, called *test set* or *holdout set*, which is unavailable to the learning algorithm at training time.

Holdout method. The training set is randomly divided into two disjoint subsets. One is used as the training set and another the test set. The drawback of this method is that the accuracy estimate of a single random split may heavily depend on the specific partitions formed, as different partitions may lead to considerably different results. This problem can be alleviated by using resampling.

Random subsampling. This approach performs multiple data splits of the training data and averages estimated accuracies on holdout sets. Random subsampling produces more stable accuracy estimates than the holdout method.

Cross-validation. In a u-fold cross-validation, the training data is randomly divided into u mutually exclusive subsets of approximately equal size. Sequentially, one subset is used as the test set for the classifier generated from the remaining $u - 1$ subsets. Accuracy is obtained as the average of the estimates on u subsets. The advantage of this technique is that every instance in the training data is used once in the role of test set, and hence the cross-validation accuracy is the percentage of instances that are correctly classified.

Leave-One-Out cross-validation. This is a special case of u-fold cross-validation, in which u is the total number of instances (t). That is, each instance in the training set is in turn left out to test the classifier generated from all the remaining instances. Leave-One-Out cross-validation uses $t - 1$ instances for training and thus may be expected to produce a classifier with similar accuracy to that which might be expected from a classifier learnt from all the data. However, as the sample size grows, the

computational overheads of Leave-One-Out cross-validation may be quite high, as t separate classifiers must be learnt.

Bootstrap. This is a resampling technique with replacement. A training data of t instances is randomly sampled with replacement to form another data set of t instances. The new data set is used for training and those instances in the original data that have not been selected are used for testing. This process is repeated several times and the results are averaged to obtain the accuracy.

2.2.1.2 Bias and Variance

Error can be decomposed into bias, variance and irreducible error [Kong and Dietterich, 1995; Breiman, 1996; Kohavi and Wolpert, 1996; Friedman, 1997; Heskes, 1998; Webb, 2000; Domingos, 2000; James, 2003]. This decomposition provides valuable insights into the components of the error. *Bias* denotes the systematic component of error, which relates to how closely the learner is able to describe the decision surfaces for a domain. *Variance* describes the component of error that stems from sampling, which reflects the sensitivity of the learner to variations in the training sample. *Irreducible error* is the lower error bound of any classifier and usually aggregated into the bias or the variance in supervised classification learning.

Unfortunately, we cannot in general minimize bias and variance simultaneously. There is a bias-variance tradeoff such that bias typically increases when variance decreases and vice versa. Algorithms that form models with few parameters usually have low variance in that they are insensitive to data variations, and high bias because the simple models generated generally underfit the data. In contrast, algorithms that learn models that are highly parameterized usually have low bias because they can generate complex models to fit the training data closely, and high variance for the reason that the models they create differ substantially between different training samples. In such a case, the model built from the training data may not generalize well to other data. This is called *overfitting*. In general, the better the learner is able to fit the training data, the lower the bias will be. However, closely fitting the training data may result in greater changes in the models formed from sample to sample, and hence higher variance.

One frequently used bias-variance decomposition method is Kohavi and Wolpert's method (1996), which defines bias and variance as follows.

$$bias^2 = \frac{1}{2}\sum_{y \in Y} \left(P(Y = y \mid \mathbf{X} = \mathbf{x}) - P(\mathcal{C}(\mathbf{x}) = y)\right)^2$$

$$variance = \frac{1}{2}\left(1 - \sum_{y \in Y} P(\mathcal{C}(\mathbf{x}) = y)^2\right)$$

$$\sigma^2 = \frac{1}{2}\left(1 - \sum_{y \in Y} P(Y = y \mid \mathbf{X} = \mathbf{x})^2\right)$$

When estimating these metrics from experimental data, the irreducible error, σ^2, is usually aggregated into $bias^2$.

2.2.2 Computational Complexity

The computational complexity theoretically measures the time and space that an algorithm requires given the size of input data. Big-O notation describes how the size of input data affects the usage of computational resources and is always used to specify the worst-case performance of an algorithm given its inputs [Bachmann, 1894; Sipser, 1997; Cormen, Leiserson, Rivest and Stein, 2001].

Given the input size of s and two positive non-decreasing functions f and g, we say that $f(s)$ has order $O\big(g(s)\big)$ if for some positive constants c and s_0 the following condition holds:

$$\forall s \geq s_0, f(s) \leq cg(s).$$

In this thesis, we use Big-O notation to bound the worst-case running time or space of an algorithm. For example, $O\big(s^2\big)$ is used to upper bound the running time of the insertion sort algorithm, where s is the number of elements to be sorted.

2.2.2.1 Time Complexity

The time complexity of an algorithm is the amount of time needed to fulfil its task. In supervised classification learning, classification of an instance consists two steps:

- Generate a classifier from the training data (training time phase).
- Classify a new instance (classification time phase).

We use *training time complexity* to measure the time taken to establish a classifier and *classification time complexity* to measure the time taken to classify a new instance.

2.2.2.2 Space Complexity

The space complexity measures the amount of space (or memory) required by an algorithm. *Training space complexity* and *classification space complexity* are used to measure the space needed respectively to form a classifier and classify a new instance.

2.3 Evaluation of Probabilistic Prediction

Error rate (or zero-one-loss) is a standard measurement used to evaluate learning algorithms. In many applications, for algorithms that can also produce probability estimates, it is desirable to obtain accurate probability estimates rather than a simple classification. For example, a correct prediction of an edible mushroom with a probability of 1.0 is obviously more accurate than a prediction with a probability of 0.51. However, under zero-one loss, these two predictions have the same loss (zero) [Hope and Korb, 2004]. Two frequently used methods to evaluate probabilistic prediction are the quadratic loss function and the information loss function [Witten and Frank, 2005]. They reward predictors that can accurately predict the true probabilities.

2.3.1 Quadratic Loss function

For a single test instance $\mathbf{x}$, the quadratic loss function is:

$$\begin{aligned}&\sum_{i=1}^{k}(\hat{P}(y_i \mid \mathbf{x}) - b_i)^2 \\ &= \quad 1 - 2\hat{P}(y_j \mid \mathbf{x}) + \sum_{i=1}^{k} \hat{P}(y_i \mid \mathbf{x})^2,\end{aligned}$$

where $\hat{P}(y_i \mid \mathbf{x})$ is the probability estimate for the ith class, $\hat{P}(y_j \mid \mathbf{x})$ is the probability estimate for the actual class and b_i is 1 if i is the actual class and 0 otherwise. When there are several instances in the test set, the loss function is summed over them all. The lower the quadratic loss, the more accurate probability estimates we obtain. For the mushroom example, the quadratic loss of the first prediction is 0, while that of the second prediction is 0.4802.

Root mean squared error (RMSE) is a commonly used metric to evaluate the accuracy of probability estimates. It is the square root of the mean squared error given by the quadratic loss function:

$$\sqrt{\frac{\sum_i(\hat{P}(y_i \mid \mathbf{x}) - b_i)^2}{k}}.$$

2.3.2 Information Loss Function

The information loss function for a single test instance $\mathbf{x}$ is:

$$-\log_2 \hat{P}(y_i \mid \mathbf{x}),$$

where i is the index of actual class of $\mathbf{x}$, and $\hat{P}(y_i \mid \mathbf{x})$ is the probability estimate for the ith class. The larger the probability assigned to the actual class, the lower the information loss will be, and hence the better the probabilistic estimation. For the mushroom example, the information loss of the first prediction is 0 and that of the second prediction is 0.9714. If a classifier assigns a zero probability to the actual class, the information loss is infinite. In contrast, the upper bound of the quadratic loss function is $1 + \sum_i \hat{P}(y_i \mid \mathbf{x})^2 \leq 2$. We use the quadratic loss function to evaluate

probabilistic prediction due to the complexity of handling infinite values that can arise with the information loss function.

2.4 Discretization

Some learning algorithms require that data comprises only qualitative attributes. Therefore, before they can be applied to data sets with quantitative attributes, these algorithms need a preprocessing step, called *discretization*, to transform quantitative attributes to qualitative ones. Discretization can also improve accuracy and efficiency for other learning algorithms which can deal with quantitative attributes [Catlett, 1991; Kerber, 1992; Dougherty, Kohavi and Sahami, 1995; Cerquides and de Màntaras, 1997; Frank and Witten, 1999; Liu, Hussain, Tan and Dash, 2002; Yang, 2003].

Most discretization methods are supervised methods. They take into account the class information to select discretization *cut points*, which are real values that divide the range of quantitative values into intervals. One commonly used discretization approach is *MDL discretization* [Fayyad and Irani, 1993]. It uses class information entropy to evaluate candidate cut points, selecting the cut point with the lowest entropy to binarize the range. It recursively binarizes the two intervals of the previous partition until a stopping criterion is met. The *minimum description length* (MDL) principle [Rissanen, 1978], which recommends to select the hypothesis for the data that achieves the best data compression [Grünwald, Pitt and Myung, 2005], is used as a stopping criterion. The discretization is stopped when the cost of encoding the current cut point and the classes of instances below and above the cut point is greater than the cost of encoding the classes of instances before splitting.

For detailed studies of various discretization methods, the reader can refer to [Dougherty et al., 1995; Liu et al., 2002; Yang, 2003].

2.5 Attribute Selection

Attribute selection is the process of selecting an attribute subset [Langley, 1994; Kohavi and John, 1997; Blum and Langley, 1997; Guyon and Elisseeff, 2003; Liu and Yu, 2005]. It has many potential beneficial effects, reducing the dimensionality of the

data, removing irrelevant or redundant attributes, improving the accuracy of learning algorithms and providing faster prediction.

2.5.1 Irrelevant and Redundant Attributes

The accuracy of classification learners is often reduced by the presence of two types of attributes, irrelevant and redundant attributes.

Irrelevant attributes. An attribute that is not necessary for predicting the class is called an *irrelevant* attribute. In other words, an attribute is irrelevant if its removal from the attribute set does not reduce the prediction power. There are many different definitions of relevance in the literature. We refer the reader to Kohavi and John (1997) and Blum and Langley (1997).

Redundant attributes. An attribute whose values are dependent on or can be derived from the values of other attributes is called a *redundant* attribute. *Redundancy* implies that attributes share mutual information. In real world problems, redundancy is often inobvious. For instance, one attribute can be a function of another or of several other attributes.

2.5.2 Heuristic Search

Since there are 2^n candidate subsets of n attributes, an exhaustive search of the space, even with a moderate n, is prohibitive [Amaldi and Kann, 1998]. This necessitates the use of *heuristic search*, which guarantees to find a locally (not necessarily globally) optimal attribute subset in reasonable time. *Greedy hill climbing* is a simple and widely used heuristic search. It adds or removes an attribute irrevocably at each step. That is, once an attribute is added or removed, that action cannot be undone. In consequence, the time complexity of this search process is $O\big(n^2\big)$.

2.5.2.1 Search Direction

Forward selection begins with the empty attribute set and successively adds attributes, while *backward elimination* starts with the complete attribute set and successively removes attributes. *Bidirectional search* begins with both empty and entire attribute sets, and simultaneously adds and removes attributes.

2.5.2.2 Subset Evaluation

To measure the effectiveness of alternative attribute subsets, we need an *evaluation function*, which measures the discriminating ability of an attribute or an attribute set among classes. Following the classification of Kohavi and John (1997), algorithms for attribute selection broadly fall into two categories.

Wrapper model. The wrapper approach uses accuracy estimates of a target learning algorithm as an evaluation function [Kittler, 1986; Doak, 1992; Caruana and Freitag, 1994; Langley and Sage, 1994; Pazzani, 1996; Kohavi, 1996; Kohavi and John, 1997; Keogh and Pazzani, 1999; Zheng and Webb, 2000]. Given an attribute subset, the accuracy of the target algorithm is estimated using cross validation. The wrapper approach appears to deliver high accuracy as it takes the bias of the target algorithm into account and thus the selected attribute subset may be better suited to the algorithm [Langley, 1994]. However, even for an algorithm with a moderate complexity, repeatedly applying the algorithm on attribute subsets results in high computational overheads [Hall, 2000; Forman, 2003].

Filter model. The filter approach assesses attribute subsets based on general characteristics of data, independently of any learning algorithm [Mucciardi and Gose, 1971; Narendra and Fukunaga, 1977; Sheinvald, Dom and Niblack, 1990; Almuallim and Dietterich, 1991; Oliveira and Sangiovanni-Vincentelli, 1992; Kira and Rendell, 1992; Cardie, 1993; Modrzejewski, 1993; Schlimmer, 1993; Koller and Sahami, 1996; Hall, 2000] Generally, filter approaches are much more efficient than wrapper approaches. As a consequence, for data with high dimensionality, the filter approach might be a more practical choice.

Recently, there have been attempts to combine these two models. We call them the *hybrid* models [Das, 2001; Xing, Jordan and Karp, 2001].

2.5.2.3 Stopping Criteria

There are three commonly used options for halting the search.

Continue Search and Select Best (CSSB). This strategy continues the search until all attributes have been added or removed and then selects the attribute subset with the highest evaluation.

Stop on First Nonimprovement (SFN). This option terminates the search when subsequent attribute addition (or elimination) does not improve the evaluation of the current attribute set.

Stop on First Reduction (SFR). This method performs attribute selection continually so long as the evaluation of the current attribute set is not reduced.

2.6 Classification Algorithms

In this section, we first briefly describe Bayesian network classifiers [Friedman, Geiger and Goldszmidt, 1997]. The simplest Bayesian network classifier, naive Bayes (NB) [Kononenko, 1990; Langley et al., 1992; Langley and Sage, 1994], will be discussed in Chapter 3. Next, we describe logistic regression [Mclachlan, 1992], which is the discriminative counterpart of NB [Mclachlan, 1992; Rubinstein and Hastie, 1997]. *Discriminative* classifiers directly estimate the posterior probability $P(y \mid \mathbf{x})$, while *generative* classifiers estimate the joint probability $P(y, \mathbf{x})$ and calculate $P(y \mid \mathbf{x})$ by using Bayes rule. Finally, we introduce support vector machines (SVM) [Boser, Guyon and Vapnik, 1992; Cortes and Vapnik, 1995], which have become quite popular due to its strong theoretical foundation and desirable performance. These methods will be included in a comprehensive comparison performed in the next chapter.

2.6.1 Bayesian Network Classifiers

A *Bayesian network* [Pearl, 1988] represents a probability distribution by using a *directed acyclic graph* (DAG) $G = \langle V_G, E_G \rangle$, where $V_G = \{Z_1, \ldots, Z_n\}$ is the set of nodes which represent variables and E_G is the set of arcs which represent conditional

dependencies between variables. These dependencies are quantified by a set of local conditional probabilities. In the DAG G, each variable Z_i is independent of its non-descendants given its immediate predecessors, called *parents* and indicated as $\pi(Z_i)$. Figure 2.1 illustrates a simple Bayesian network taken from [Pearl and Russell, 2000].

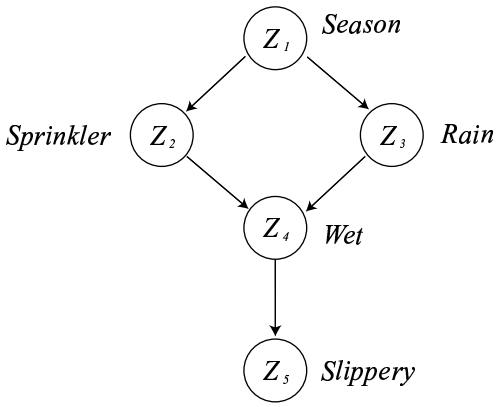


Figure 2.1: A Bayesian network [Pearl and Russell, 2000].

It describes the relationships among five variables. For example, Z_5 (whether the pavement is slippery) is independent of Z_1 (season of the year), Z_2 (whether the sprinkler is on) and Z_3 (whether it is raining) given Z_4 (the pavement is wet).

The joint probability over V_G can be defined by

$$P(z_1, \ldots, z_n) = \prod_{i=1}^{n} P(z_i \mid \pi(z_i)),$$

where z_i is a value of Z_i and $\pi(z_i)$ is a value of $\pi(Z_i)$. Hence, in the example illustrated in Figure 2.1, we have

$$P(z_1, z_2, z_3, z_4, z_5) = P(z_1)P(z_2|z_1)P(z_3|z_1)P(z_4|z_2, z_3)P(z_5|z_4).$$

Learning a Bayesian network is the process of finding a network that can best match the training set [Friedman et al., 1997; Acid, Campos and Castellano, 2005]. The application of a learned Bayesian network to classification is simple, calculating the posterior probability of the class given an instance and classifying the instance into the class with the highest posterior probability [Duda and Hart, 1973]. We call

the classifier that uses a Bayesian network to classify an instance a *Bayesian network classifier* [Friedman et al., 1997].

The simplest Bayesian network classifier is NB [Kononenko, 1990; Langley et al., 1992; Langley and Sage, 1994], in which Y has no parents and X_i only has Y as its parent ($1 \leq i \leq n$). We will formally describe NB and its extensions in the next chapter.

2.6.2 Logistic Regression

Logistic regression is a linear method for classification [Mclachlan, 1992; Mitchell, 2005; Witten and Frank, 2005]. It directly estimates the posterior class probability from the training set and assigns the class with the highest posterior probability to the test instance. In the case where Y is a boolean variable, the logistic regression model is defined as

$$P(Y = 1 \mid X) = \frac{1}{1 + \exp(\mathrm{w}_0 + \sum_{i=1}^{n} \mathrm{w}_i X_i)}$$

and

$$P(Y = 0 \mid X) = 1 - P(Y = 1 \mid X) = \frac{\exp(\mathrm{w}_0 + \sum_{i=1}^{n} \mathrm{w}_i X_i)}{1 + \exp(\mathrm{w}_0 + \sum_{i=1}^{n} \mathrm{w}_i X_i)},$$

where w_i is the parameter to be estimated ($0 \leq i \leq n$). The vector of parameters $W = \langle \mathrm{w}_0, \ldots, \mathrm{w}_n \rangle$ is usually fit to maximize the conditional log-likelihood:

$$W \leftarrow \underset{W}{\operatorname{argmax}} \sum_{o=1}^{t} \ln P(Y^o \mid X^o, W),$$

where Y^o and X^o are respectively the observed Y value and X value in the oth training instance. A commonly used method for this maximization problem is *gradient ascent* [Avriel, 2003; Pedregal, 2004]. The parameters are initialized to zero and continually updated in the direction of the gradient until a global maximum is reached:

$$W \leftarrow W + \eta \frac{\partial \sum_{o=1}^{t} \ln P(Y^o \mid X^o, W)}{\partial W}, \tag{2.1}$$

where η is the learning rate.

For a multi-class problem, the logistic regression model is defined as

$$P(Y = c_l \mid X) = \frac{\exp(\mathrm{w}_{l0} + \sum_{i=1}^{n} \mathrm{w}_{li} X_i)}{1 + \sum_{j=1}^{k-1} \exp(\mathrm{w}_{j0} + \sum_{i=1}^{n} \mathrm{w}_{ji} X_i)}.$$

and

$$P(Y = c_k \mid X) = \frac{1}{1 + \sum_{j=1}^{k-1} \exp(\mathrm{w}_{j0} + \sum_{i=1}^{n} \mathrm{w}_{ji} X_i)},$$

where $1 \leq l \leq k-1$ and w_{ji} is the parameter for $Y = c_j$ and X_i. It also uses (2.1) to optimize W.

2.6.3 Support Vector Machine

A *Support vector machine* (SVM) [Boser et al., 1992; Cortes and Vapnik, 1995] is defined for two-class problems, where $y_i \in \{-1, 1\}, 1 \leq i \leq t$. It performs classification by mapping the training data into a higher dimensional space and constructing a separating hyperplane to linearly separate the data into two classes with the maximal margin.

In general, a separating hyperplane takes the form

$$\mathrm{w} \cdot \mathbf{x} + b = 0,$$

where $\mathrm{w} = \langle \mathrm{w}_1, \ldots, \mathrm{w}_n \rangle$ is a weight vector, b is a threshold and $\cdot$ is the dot product. The *optimal separating hyperplane* is the one that provides maximum separation between the classes. The instances that have minimum distance to the optimal separating hyperplane are called *support vectors*. It is quite likely that the test instances are close to the training instances, and hence the greater the margin between support vectors and the optimal separating hyperplane, the better the classification accuracy a SVM may obtain.

Figure 2.2 illustrates the optimal separating hyperplane and support vectors in a *linearly separable* problem taken from [Hearst, Schölkopf, Dumais, Osuna and Platt, 1998]. The training set is linearly separable if it can be separated by an $n-1$ dimensional hyperplane. By scaling w and b, the hyperplanes, which are parallel to the optimal separating hyperplane and closest to support vectors can be described

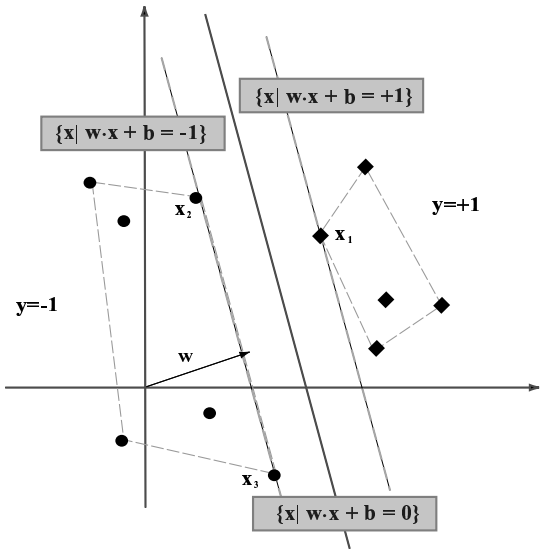


Figure 2.2: Separating hyperplanes for a linearly separable problem. $\mathbf{x}_1$, $\mathbf{x}_2$ and $\mathbf{x}_3$ are support vectors [Hearst, Schölkopf, Dumais, Osuna and Platt, 1998].

as:

$$\mathrm{w} \cdot \mathbf{x} + b = 1$$

and

$$\mathrm{w} \cdot \mathbf{x} + b = -1.$$

All instances can be separated into two classes if for $1 \leq i \leq t$, either $\mathrm{w} \cdot \mathbf{x}_i + b \geq 1$ or $\mathrm{w} \cdot \mathbf{x}_i + b \leq -1$. This is equivalent to

$$y_i(\mathrm{w} \cdot \mathbf{x}_i + b) \geq 1, \qquad 1 \leq i \leq t. \tag{2.2}$$

The distance between the hyperplanes is $2/ \parallel \mathrm{w} \parallel$. Therefore, to maximize the distance between the hyperplanes requires the solution of the following optimization problem:

$$\text{minimize } \frac{1}{2} \parallel \mathrm{w} \parallel^2, \quad \text{subject to} \quad y_i(\mathrm{w} \cdot \mathbf{x}_i + b) \geq 1, \quad 1 \leq i \leq t \tag{2.3}$$

Ideally, a SVM can establish an optimal separating hyperplane that completely separates the training data. However, due to outliers, perfect separation may not exist. In the case when perfect separation does exist, it may result in a model with large number of dimensions and hence overfitting. To allow a small number of misclassified instances, Corinna Cortes and Vladimir Vapnik (1995) introduce slack variables $\xi_i \geq 0$ $(1 \leq i \leq t)$, which describe the degree of misclassification, to relax (2.2) to

$$y_i(\mathrm{w} \cdot \mathbf{x}_i + b) \geq 1 - \xi_i \qquad 1 \leq i \leq t. \tag{2.4}$$

It is clear that the larger the ξ_i, the easier the (2.4) to be met. Thus, we need to penalize large ξ_i in the objective function. The equation (2.5) now transforms to

$$\text{minimize } \frac{1}{2} \parallel \mathrm{w} \parallel^2 + C \sum_i \xi_i, \quad \text{subject to} \quad y_i(\mathrm{w} \cdot \mathbf{x}_i + b) \geq 1 - \xi_i, \quad 1 \leq i \leq t, \tag{2.5}$$

where $C > 0$ is the penalty parameter of the error term.

For a non-linearly separable problem, SVM uses a *kernel function* to map the training data into a higher dimensional space and then construct the optimal separating hyperplane in the new space to perform linear classification. A kernel function has the form

$$K(\mathbf{x}_i, \mathbf{x}_j) = \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j),$$

where Φ is a nonlinear map. The most commonly used kernel function in SVM is *radial basis function* (RBF) kernel

$$K(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\gamma \parallel \mathbf{x}_i - \mathbf{x}_j \parallel^2), \quad for\ \gamma > 0.$$

Figure 2.3 [Hearst et al., 1998] shows that mapping a non-linearly separable training data into a higher dimensional space via Φ can make it possible to perform linear separation.

SVM is a binary classifier algorithm. One frequently used approach to generalizing from two-class classification to multi-class classification is the *one-against-one* method [Hsu and Lin, 2002; Wu, Lin and Weng, 2004], where a classifier is trained between

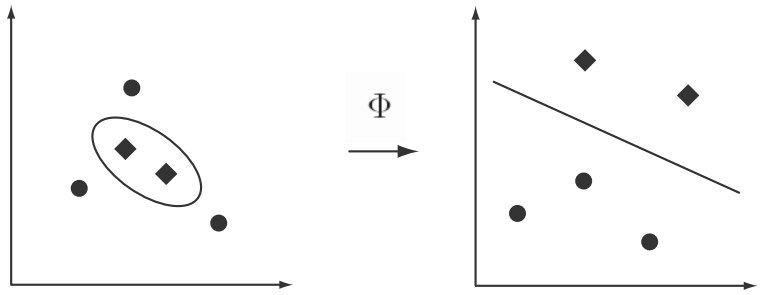


Figure 2.3: Linear separation might be possible in a higher dimensional space [Hearst, Schölkopf, Dumais, Osuna and Platt, 1998].

each pair of classes (hence, there are $k(k-1)/2$ binary classifiers), and the predicting class is the one with the largest vote.

2.7 Summary

The aim of this chapter is to provide some explanations of the concepts and terms of supervised classification learning. These concepts and terms assist the reader to appreciate NB and its extensions presented in the following chapters. Bias, variance, RMSE, time and space complexity introduced in this chapter are employed to evaluate these algorithms.

Chapter 3

Naive Bayes and Its Extensions

Naive Bayes (NB) is a simple, computationally efficient and effective probabilistic approach to classification learning built on the assumption of conditional independence between the attributes given the class. Its success and popularity has led to the development of semi-naive Bayesian methods that seek to retain its numerous strengths while reducing error by alleviating the attribute interdependence problem. This chapter describes NB and analyzes its strengths and weaknesses. It categorizes previous semi-naive Bayesian methods into four groups: those that apply conventional NB to a new attribute set, those that alter NB by allowing interdependencies between attributes, those that apply NB to a subset of the training sample, and those that perform corrections to NB's probability estimates. Twelve key semi-naive Bayesian algorithms are analyzed in detail. We perform comparative analysis of their features and benchmark them using error analysis based on the bias-variance decomposition, probabilistic prediction analysis based on the quadratic loss function and training and classification time analysis on sixty natural domains from the UCI Machine Learning Repository. To provide a baseline for comparison, we also present comprehensive experimental results for Logistic Regression and LibSVM, a popular SVM implementation. In analyzing the results of these experiments we provide general guidelines for selection between semi-naive Bayesian methods based on the characteristics of the application to which they are applied.

3.1 Naive Bayes (NB)

Despite its unrealistic attribute conditional independence assumption, NB has demonstrated competitive classification accuracy to many other sophisticated methods over a considerable range of classification tasks. This section formally describes classification with NB, analyzes NB's time and space complexity and examines its merits and limitations.

3.1.1 Classification with NB

The Bayesian classifier [Duda and Hart, 1973] labels an unseen instance with the class that has the highest estimated posterior probability. Formally, it predicts the class for an instance $\mathbf{x} = \langle x_1, \ldots, x_n \rangle$ by using

$$\begin{aligned} \operatorname*{argmax}_y P(y \,|\, \mathbf{x}) &= \operatorname*{argmax}_y P(y, \mathbf{x}) / P(\mathbf{x}) && (3.1) \\ &= \operatorname*{argmax}_y P(y, \mathbf{x}). && (3.2) \end{aligned}$$

The equality holds between (3.1) and (3.2) due to $P(\mathbf{x})$ being invariant across values of y.

Where estimates of $P(y \,|\, \mathbf{x})$ are required rather than a simple classification, these can be obtained by normalization,

$$\hat{P}(y \,|\, \mathbf{x}) = \frac{\hat{P}(y, \mathbf{x})}{\sum_{i=1}^{k} \hat{P}(c_i, \mathbf{x})}, \tag{3.3}$$

where $\hat{P}(\cdot)$ represents an estimate of $P(\cdot)$.

For ease of explication, we describe NB and its variants by the manner in which each calculates the estimate $\hat{P}(y, \mathbf{x})$. This estimate is then utilized with (3.2) or (3.3) to perform respectively classification or conditional probability estimation.

Since the number of combinations of attribute values is v^n, to accurately estimate $P(Y, \mathbf{X})$, we need to estimate $O(kv^n)$ parameters, each requiring sufficient examples to support reliable estimation [Mitchell, 2005]. It is clearly unrealistic to do this directly in most real world domains as, if the number of attributes is large, some

instances are unlikely to occur in the given training data, and hence it is impossible to obtain the estimate of $P(Y, \mathbf{X})$ directly from the training sample.

NB [Kononenko, 1990; Langley et al., 1992; Langley and Sage, 1994] circumvents this problem by making the assumption that the attributes are independent given the class. Consequently, the number of parameters to estimate $P(Y, \mathbf{X})$ is dramatically reduced to $O(kvn)$. Under this conditional independence assumption NB estimates $P(y, \mathbf{x})$ by

$$\hat{P}(y, \mathbf{x}) = \hat{P}(y) \prod_{i=1}^{n} \hat{P}(x_i \mid y). \tag{3.4}$$

In NB, the class Y is qualitative and an attribute X_i can be either qualitative or quantitative. For qualitative attributes, $P(y)$ is estimated by the frequency of instances with value y, and $P(x_i \mid y)$ is estimated by the frequency of instances with y and x_i divided by that of instances with y. To avoid the problems that result from zero frequencies and zero probabilities, smoothing methods, such as *Laplace estimation* and *m-estimation* [Cestnik, 1990], are employed. Using Laplace estimation, we have

$$\hat{P}(y) = \frac{F(y) + 1}{t + k}$$

and

$$\hat{P}(x_i \mid y) = \frac{F(y, x_i) + 1}{F(y) + v_i},$$

where $F(y)$ is the frequency of y and $F(y, x_i)$ is the frequency of y and x_i in the training set. Using m-estimation, we have

$$\hat{P}(y) = \frac{F(y) + \frac{m}{k}}{t + m}$$

and

$$\hat{P}(x_i \mid y) = \frac{F(y, x_i) + \frac{m}{v_i}}{F(y) + m},$$

where m is a constant.

For quantitative attributes, one common approach to representing the distributions $P(X_i \mid Y)$ is to assume that X_i has a Gaussian distribution whose mean and variance depends on Y. Previous research [Dougherty et al., 1995] shows that the

classification errors of NB with discretization methods employed in their study are lower than that of NB with the assumption that quantitative attributes have a Gaussian distribution. Theoretical analysis on why discretization is effective on NB can be found in [Hsu, Huang and Wong, 2000; Hsu, Huang and Wong, 2003; Yang, 2003]. For this reason, in this thesis, quantitative attributes are discretized prior to NB's classification.

3.1.2 Complexity

At training time, NB generates a one-dimensional table of class probability estimates, indexed by class, and a two-dimensional table of conditional attribute-value probability estimates, indexed by class and attribute-value. The resulting space complexity is $O(knv)$. To generate the estimates, NB needs to scan the training data, hence the time complexity is $O(tn)$.

At classification time, to classify a single example has time complexity $O(kn)$ using the tables formed at training time with space complexity $O(knv)$.

3.1.3 Merits of NB

Simplicity and Efficiency. Due to the independence assumption, NB is simple and computationally efficient. It produces frequency tables at training time, obtained by scanning the training data once if all attributes are qualitative. Therefore, as has been discussed in Section 3.1.2, NB's training time complexity is only linear in the number of instances and attributes. Since it does not store the training data after the tables of probability estimates are generated, it is space efficient as well.

Effectiveness. Even though the attribute independence assumption is frequently unrealistic, NB has exhibited accuracy competitive with other learning algorithms for many tasks, including medical diagnosis and text classification [Domingos and Pazzani, 1996; Mitchell, 1997]. It performs optimal classification if the attribute conditional independence assumption holds and the estimation of the base probabilities from the training data is accurate. In the presence of interdependencies between attributes, NB's classification may still be optimal so long as it can generate the highest conditional probability for the most probable class [Domingos and Pazzani, 1996].

Robustness. Since missing values are simply ignored in frequency tables, if data are missing at random they may have little impact on NB's classification. NB takes into account the evidence from all attributes and uses only class and conditional attribute probabilities to perform classification. Therefore, irrelevant attributes and noise may have little influence on NB's classification as well.

Low Variance Profile. NB has low sensitivity to data variations in that it uses a fixed formula to classify instances and hence does not perform model selection. This may have the effect of decreasing the variance component of NB's error [Hastie et al., 2001].

Incrementality. NB performs classification by using class and conditional attribute probability estimates. In consequence, when new training instances are available, it only needs to update the probability estimates, which process has time complexity $O(n)$. This desirable characteristic makes NB an attractive option for data mining applications where databases grow at a rapid pace.

3.1.4 Limitations of NB

Limited Representation Ability. NB can master linearly separable concepts (refer to Section 2.6.3) in binary domains [Duda and Hart, 1973], but cannot represent some linearly separable concepts (for example, some *m-of-n concepts* [1]) and many non-linearly separable concepts (for example, the *XOR function* [2]) [Kohavi, 1995; Domingos and Pazzani, 1997; Rish, 2001].

Sub-optimal Classification. NB excels when the attribute conditional independence assumption holds. Although some violations of the assumption do not affect its optimality [Domingos and Pazzani, 1996], many do render its classification sub-optimal. For example, in an extreme case where an attribute is perfectly correlated with another, NB double counts the redundant information provided by the attribute. Table 3.1.4 illustrates a linearly separable function which classifies instances to 1 if the values of any two of X_1, X_2 and X_3 are 1 and to 0 otherwise [Duda and

[1] An m-of-n concept is true if m or more out of n binary attributes are true.
[2] The XOR function, indicated as $x_i \oplus x_j$, is true if and only if $x_i \neq x_j$.

Hart, 1973; Langley, 1993]. NB can successfully represent this function. However, as illustrated in Table 3.2, when a redundant attribute X_0 (the negation of X_1) is included, NB misclassifies the 4th instance to 0.

Table 3.1: NB can successfully represent this linearly separable function.

X_1	X_2	X_3	Y
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Table 3.2: Given a redundant attribute X_0, NB misclassifies the 4th instance.

X_0	X_1	X_2	X_3	Y	
1	0	0	0	0	
1	0	0	1	0	
1	0	1	0	0	
1	0	1	1	0	×
0	1	0	0	0	
0	1	0	1	1	
0	1	1	0	1	
0	1	1	1	1	

High Bias Profile. As discussed in Section 2.2.1.2, there is a bias-variance trade-off such that bias typically increases when variance decreases and vice versa. NB has low variance and at the same time high bias. Variance is expected to decrease with increasing training sample size, as the differences between the different samples decrease and hence the differences between the models generated from the samples decrease [Brain and Webb, 2002]. It follows that bias is likely to be the most important factor that governs the prediction error for problems with large training samples. Therefore, the accuracy of NB may not scale up as well as other algorithms that have lower bias.

3.2 Previous Semi-naive Bayesian Methods

There are many attempts to further improve NB's accuracy by alleviating the attribute interdependence problem while at the same time retaining its simplicity and efficiency [Kittler, 1986; Kononenko, 1991; Langley, 1993; Langley and

Sage, 1994; Kohavi, 1996; Pazzani, 1996; Sahami, 1996; Singh and Provan, 1996; Friedman et al., 1997; Webb and Pazzani, 1998; Keogh and Pazzani, 1999; Zheng, Webb and Ting, 1999; Zheng and Webb, 2000; Webb, 2001; Xie, Hsu, Liu and Lee, 2002; Frank, Hall and Pfahringer, 2003; Gama, 2003; Webb, Boughton and Wang, 2005; Acid et al., 2005; Cerquides and Mántaras, 2005b; Cerquides and Mántaras, 2005a; Greiner, Su, Shen and Zhou, 2005; Roos, Wettig, Grünwald, Myllymäki and Tirri, 2005; Zhang, Jiang and Su, 2005; Langseth and Nielsen, 2006]. As already discussed in Section 3.1.3, Domingos and Pazzani (1996) point out that interdependence between attributes will not affect NB's classification accuracy, so long as it can generate the correct ranks of conditional probabilities for the classes. However, the success of semi-naive Bayesian methods show that appropriate relaxation of the attribute independence assumption is effective at improving its accuracy. Further, in many applications it is desirable to obtain accurate estimates of the conditional class probability rather than a simple classification, and hence mere correct ranking will not suffice.

Previous semi-naive Bayesian methods can be roughly subdivided into four groups. The first group applies NB to a new attribute set, which can be generated by deleting attributes [Kittler, 1986; Langley and Sage, 1994; Pazzani, 1996] and joining attributes [Kononenko, 1991; Pazzani, 1996; Langseth and Nielsen, 2006]. The second group adds explicit interdependencies between attributes. Sahami (1996) introduces the terminology of the *z-dependence Bayesian classifier*, in which each attribute depends upon the class and at most z other attributes. Within this framework, NB is a 0-dependence estimator. The majority of semi-naive Bayesian methods that add explicit interdependencies between attributes establish 1-dependence classifiers [Friedman et al., 1997; Keogh and Pazzani, 1999; Webb et al., 2005; Cerquides and Mántaras, 2005a; Zhang et al., 2005]. Two exceptions are NBTree [Kohavi, 1996] and Lazy Bayesian Rules (LBR) [Zheng and Webb, 2000], both of which may add any number of dependencies for an attribute. This group and the third group, which applies NB to a subset of the training instances [Langley, 1993; Frank et al., 2003], are not mutually exclusive. For example, NBTree and LBR classify instances by applying NB to a subset of the training instances, and hence they can also be categorized to the third group. The fourth group performs adjustments to the output of NB without altering its direct operation [Webb and Pazzani, 1998; Platt, 1999; Zadrozny and Elkan, 2001; Gama, 2003].

It is also useful to distinguish between *eager learning* methods [Kittler, 1986; Kononenko, 1991; Langley, 1993; Langley and Sage, 1994; Kohavi, 1996; Pazzani, 1996; Friedman et al., 1997; Webb and Pazzani, 1998; Keogh and Pazzani, 1999; Gama, 2003; Webb et al., 2005; Cerquides and Mántaras, 2005a; Zhang et al., 2005], which perform learning at training time, and *lazy learning* methods [Zheng and Webb, 2000; Frank et al., 2003], which defer learning until classification time.

In this study, we focus on variants of NB other than variants of logistic regression, the discriminative analog of NB.

3.2.1 Applying NB to a New Attribute Set

In NB, all the attributes are used during prediction, and hence all have some influence on classification. When two attributes are strongly related, the influence from these two attributes may be given too much weight, and the influence of the other attributes may be reduced, which can result in prediction bias. Deleting one of these attributes may have the effect of alleviating the problem. In addition, irrelevant attributes may also degrade the performance of NB, in effect increasing variance without decreasing bias. Hence their removal is also useful.

3.2.1.1 Backward Sequential Elimination and Forward Sequential Selection

Backward Sequential Elimination (BSE) [Kittler, 1986] and Forward Sequential Selection (FSS) [Langley and Sage, 1994] use a simple heuristic wrapper approach that seeks to minimize error on the training set in order to detect and repair harmful interdependencies. Leave-one-out cross validation error is used as a selection criterion. Starting from the full set of attributes, BSE operates by iteratively removing successive attributes, each time removing the attribute whose elimination best reduces training set error. FSS uses the reverse search direction and operates by iteratively adding successive attributes, each time adding the attribute whose addition most improves training set accuracy. BSE terminates the process when there is no accuracy improvement, while FSS performs selection continually as long as the accuracy is not reduced.

We denote the resulting attribute subset as S. Independence is assumed among the attributes in S given the class. Therefore, BSE and FSS estimate $P(y, \mathbf{x})$ by

$$\hat{P}(y, \mathbf{x}) = \hat{P}(y) \prod_{x \in S} \hat{P}(x \mid y).$$

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 $O\big(tn\big)$.

Keogh and Pazzani (1999) speed up the process of evaluating the classifiers by using a two-dimensional table, indexed by instance and class, to store the probability estimates. Each entry in the table is the estimate of the posterior probabilities that instance $\mathbf{x}$ belongs to class y. It is straightforward to update these to exclude or include the contribution of a specific attribute x_i by dividing or multiplying the entry $\langle \mathbf{x}, y \rangle$ by $\hat{P}(x_i|y)$. Hence, leave-one-out cross validation can be performed by simply taking each attribute x_i in turn, excluding or including x_i in the table of posterior probabilities, and then classifying $\mathbf{x}$. The resulting space complexity is $O\big(tn + tk + knv\big)$. The time complexity of a single leave-one-out cross validation is reduced from $O(tkn)$ to $O(tk)$ by using the speed up strategy. Therefore, the time complexity of attribute selection is $O\big(tkn^2\big)$, as leave-one-out cross validation will be performed at most $O\big(n^2\big)$ times.

BSE and FSS have identical time and space complexity to NB at classification time, although they may in practice result in significant speed-up if many attributes are deleted.

3.2.1.2 Backward Sequential Elimination and Joining

Creating new compound attributes when dependencies between attributes are detected is another approach to relax the attribute independence assumption. The Semi-naive Bayesian Classifier [Kononenko, 1991] uses an exhaustive search to join attribute values iteratively based on a statistical method for identifying correlations

between attributes. However, the reported experimental results were not compelling and the technique does not appear to have been utilized since it was first proposed.

Backward Sequential Elimination and Joining (BSEJ) [Pazzani, 1996] uses predictive accuracy on leave-one-out cross validation as a merging criterion to create new Cartesian product attributes. The value set of a new Cartesian product attribute is the Cartesian product of the value sets of the two original attributes. For instance, if attribute X_1 has two values: x_1^1 and x_1^2, and attribute X_2 has three values: x_2^1, x_2^2 and x_2^3, the new Cartesian product attribute will have six values: $x_1^1x_2^1$,$x_1^1x_2^2$, $x_1^1x_2^3$, $x_1^2x_2^1$, $x_1^2x_2^2$ and $x_1^2x_2^3$. In addition to creating new Cartesian product attributes, BSEJ deletes original attributes and also new Cartesian product attributes during a hill-climbing search. It repeatedly joins the pair of attributes or deletes the attribute such that the action most improves predictive accuracy on leave-one-out cross validation. This process terminates when there is no further accuracy improvement.

The resulting Cartesian product attribute set is denoted as M. The set of remaining original attributes that have not been either deleted or joined is indicated as R. Independence is assumed among the attributes in R and M given the class. Hence, BSEJ estimates $P(y, \mathbf{x})$ by

$$\hat{P}(y, \mathbf{x}) = \hat{P}(y) \prod_{x \in R} \hat{P}(x \,|\, y) \prod_{z \in M} \hat{P}(z \,|\, y).$$

At training time BSEJ generates a one-dimensional table of class probability estimates and a two-dimensional table of conditional attribute-value probability estimates as NB does. It also generates two-dimensional tables of conditional Cartesian product attribute-value probability estimates, indexed by class and compound attribute-value. 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, each requiring a pass through the training data to generate joint probability estimates and performing leave-one-out cross validation. 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 $O\big(kv^n\big)$.

3.2.2 Altering NB by Allowing Interdependencies between Attributes

The addition of explicit arcs between attributes allows interdependencies between attributes to be addressed directly. However, techniques for learning unrestricted Bayesian networks often fail to achieve lower error than NB and sometimes lead to higher error [Friedman et al., 1997]. Two possible reasons for this are that the large number of parameters that must be estimated for full Bayesian networks lead to high variance and that full Bayesian networks are oriented toward estimating any marginal probability rather than the being specifically focused on the task of estimating the conditional probabilities of the class attribute given a full set of other attribute values.

3.2.2.1 Tree Augmented Naive Bayes and SuperParent TAN

Tree Augmented Naive Bayes (TAN) [Friedman et al., 1997] allows each attribute to depend on at most one non-class attribute. Based on this representation, they extended a method first proposed by Chow and Liu (1996) and utilized 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. It estimates $P(y, \mathbf{x})$ by

$$\hat{P}(y, \mathbf{x}) = \hat{P}(y) \prod_{i=1}^{n} \hat{P}(x_i \,|\, y, \pi(x_i)), \tag{3.5}$$

where $\pi(x_i)$, defined in Section 2.6.1, is a value of the parent of attribute X_i.

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, 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 we need to update each entry for every combination of the two attribute-values for every instance. Creating the conditional mutual information matrix requires consideration for each pair of attributes of every pairwise combination of their respective values in conjunction with each class. The resulting time complexity is $O\big(kn^2v^2\big)$. The parent function is then generated by establishing a maximal spanning tree, time complexity $O\big(n^2 \log n\big)$. At classification time, to classify a single example has time

complexity $O(kn)$. 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 $O(knv^2)$.

SuperParent TAN (SP-TAN) [Keogh and Pazzani, 1999], a variant of TAN, uses the same representation as TAN, but utilizes a wrapper approach to construct the parent function. It uses leave-one-out cross validation error as a criterion to add arcs. Those attributes without a non-class parent are identified and labeled as *orphans*. Then an attribute called the *SuperParent* is allocated as parent of all the orphans (other than itself). There are two steps to add an arc. First, the SuperParent that most improves accuracy is selected. Next, SP-TAN assesses the effect of adding a single arc from the SuperParent to each orphan. The orphan pointed to by the best arc is called the *FavoriteChild*. This SuperParent-FavoriteChild pair is then added to the current network. SP-TAN stops adding arcs when there is no further accuracy improvement. It also uses (3.5) to classify an instance.

Since TAN and SP-TAN use different criteria to establish the parent function, TAN tends to add $n-1$ arcs, while SP-TAN may have fewer arcs between attributes. Another difference is the direction of arcs. TAN randomly selects a root attribute and directs all edges away from it. This means the direction of edges is assigned randomly. In contrast, SP-TAN makes the direction from SuperParents to their favorite children.

At training time SP-TAN needs additional space complexity $O(tn)$ for storing the training data compared with TAN. The selection of a single SuperParent is order $O(tkn^2)$, and the selection of the favorite child of the SuperParent is order $O(tkn)$, which is achieved by using the speed up strategy mentioned in Section 3.2.1.1. Keogh and Pazzani (1999) proposed another optimization to speed up the evaluating process by sorting instances according to whether they are classified correctly and testing the misclassified instances first. Hence, once the number of the misclassified instances is larger than the current best-so-far error, we can stop testing the classifier. These optimizations are effective in practice. The time complexity of forming the parent function is $O(tkn^3)$, as $O(n)$ arcs are added. SP-TAN has identical classification time and space complexity to TAN.

3.2.2.2 NBTree

NBTree [Kohavi, 1996] seeks to combine the advantages of NB and decision trees. It partitions the training data using a tree structure and establishes a local NB in each leaf. It uses 5-fold cross validation accuracy estimation as the splitting criterion. Each value of a splitting attribute has its own subtree. The utility of a node is the 5-fold cross validation accuracy of NB at this node and that of a split on an attribute equals the weighted sum of the utility of nodes generated by the split. NBTree partitions the training sample according to the test on the attribute that has the highest utility, out of these that are substantially better than the utility of the current node. A split is defined to be substantial if the relative error reduction is greater than 5% and the splitting node has at least 30 instances. When there is no substantial improvement, NBTree stops the growth of the tree.

The classical decision tree predicts the same class for all test instances that reach a leaf. In NBTree, these instances are classified using a local NB in the leaf, which only considers those attributes that are not tested on the path to the leaf and those training instances that reach the leaf. Let B be the set of the splitting attributes on the path leading to the leaf, and let L be the set of the remaining attributes, we have

$$\begin{aligned} P(Y, \mathbf{X}) &= P(B)P(Y \mid B)P(L \mid Y, B) \\ &\propto P(Y \mid B)P(L \mid Y, B), \end{aligned}$$

where $P(Y \mid B)$ is the probability of the class in the leaf, in which the attributes in B have same values, and $P(L \mid Y, B)$ is the conditional probability of the remaining attributes given the class in the leaf. Therefore, NBTree estimates $P(y, \mathbf{x})$ by

$$\hat{P}(y, \mathbf{x}) = \hat{P}(y, b) \prod_{l \in L} \hat{P}(l \mid y, b),$$

where b is a value of B. NBTree is expected to have the effect of mitigating the harmful attribute interdependence problem for each local NB if B can be selected appropriately.

In NBTree, the number of leaves possible is $O\big(t\big)$, and the height of the tree is $O\big(log_v t\big)$ if we assume the tree is a balanced tree. Therefore, there are $O\big(t/v\big)$ internal

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

3.2.2.3 Lazy Bayesian Rules

Lazy Bayesian Rules (LBR) [Zheng and Webb, 2000] adopts a lazy approach and generates a Bayesian rule according to 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. The utility of an attribute-value pair is assessed using leave-one-out cross validation in the local training samples, those examples that have the attribute values in the antecedent together with the attribute value being tested. As different attribute-value pairs cover different subsets of the training samples, it is necessary to be careful in assessing the relative effectiveness of the alternatives. For example, one attribute value might select a set of examples that are already all correctly classified by the existing antecedent whereas another might select only examples that are not. If the former made no errors and the latter made only 50% errors then it would be the latter that provided that greatest improvement even though it had the higher error. To measure each attribute-value pair on the whole local training sample, the errors of the existing local NB on the training samples that satisfy the attribute values in antecedent but not the attribute value being tested, are added to the errors of the NB on the local training examples. The attribute-value pair with the lowest error, out of these that are significantly lower than the error of the current local NB, is added to the antecedent. The difference is considered as significant if the outcome of a one-tailed pairwise sign test is better than 0.05. LBR stops adding attribute-value pairs into the antecedent if there is no significant improvement.

Let q be a value of the set of attributes in the antecedent, and let O be the set of remaining attributes, LBR estimates $P(y, \mathbf{x})$ by

$$\hat{P}(y, \mathbf{x}) = \hat{P}(y, q) \prod_{o \in O} \hat{P}(o \,|\, y, q).$$

The Bayesian rule generated by LBR can be considered as a branch of a tree built by NBTree. LBR generates a rule for each new instance, while NBtree builds a single model according to all the examples in the training data. If examples are not evenly distributed among branches, NBTree may suffer from the small disjunct problem [Holte, Acker and Porter, 1989]. As LBR uses lazy learning, it may mitigate the problem by avoiding splits on an attribute when the relevant value is infrequent. It 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 $O\big(tn{+}knv\big)$.

3.2.2.4 Averaged One-Dependence Estimators

To avoid model selection and attain the efficiency of 1-dependence classifiers, Averaged One-Dependence Estimators (AODE) [Webb et al., 2005] selects a restricted class of One-Dependence Estimators (ODEs) and aggregates the predictions of all qualified estimators within this class. A single attribute, called the *SuperParent* if we borrow the term from SP-TAN, is selected as the parent of all the other attributes in each ODE. This type of ODE is called a *SuperParent One-Dependence Estimator* (SPODE). In order to avoid unreliable base probability estimates, when classifying an instance $\mathbf{x}$ the original AODE excludes SPODEs with SuperParent x_i where the frequency of the value x_i is lower than limit $m = 30$, a widely used minimum on sample size for statistical inference purposes. However, subsequent research [Cerquides and Mántaras, 2005a] reveals that this constraint actually increases error and hence the current research uses $m = 1$.

For any attribute value x_i,

$$P(y, \mathbf{x}) = P(y, x_i)P(\mathbf{x} \mid y, x_i).$$

This equality holds for every x_i. Therefore, for any $U \subseteq \{1, \ldots, n\}$,

$$P(y, \mathbf{x}) = \frac{\sum_{i \in U} P(y, x_i)P(\mathbf{x} \mid y, x_i)}{|U|}.$$

Thus,

$$P(y, \mathbf{x}) = \frac{\sum_{i:1 \leq i \leq n \wedge F(x_i) \geq m} P(y, x_i)P(\mathbf{x} \mid y, x_i)}{|\{i : 1 \leq i \leq n \wedge F(x_i) \geq m\}|}, \tag{3.6}$$

where $F(x_i)$ is the frequency of attribute-value x_i in the training sample.

AODE utilizes (3.6) and, for each ODE, an assumption that the attributes are independent given the class and the privileged attribute value x_i, estimating $P(y, \mathbf{x})$ by

$$\hat{P}(y, \mathbf{x}) = \frac{\sum_{i:1 \leq i \leq n \wedge F(x_i) \geq m} \hat{P}(y, x_i) \prod_{j=1}^{n} \hat{P}(x_j \mid y, x_i)}{|\{i : 1 \leq i \leq n \wedge F(x_i) \geq m\}|}.$$

At training time AODE 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, space complexity $O\big(k(nv)^2\big)$. Forming the three dimensional probability 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. This process can be sped up by introducing a constant array to store estimates of $P(y, x_i)$ and $P(x_j \,|\, y, x_i)$ at training time. Therefore, at classification time, we only need to read, other than calculate, these estimates. Although this does not change the classification time complexity, in practice, it may result in substantial speed-up.

3.2.2.5 Maximum a Posteriori Linear Mixture of Generative Distributions

AODE classifies by uniformly aggregating all qualified ODEs. One natural extension to AODE is to use a linear mixture assigning a weight to each ODE. Maximum a Posteriori Linear Mixture of Generative Distributions (MAPLMG) [Cerquides and Mántaras, 2005a] assigns the weights $\mathbf{w} = \langle \mathrm{w}_1, \ldots, \mathrm{w}_n \rangle$, with which the supervised posteriori for Linear Mixture of Generative Distributions (LMG) is maximized, to the ODEs. This is an optimization problem under the constraint that $\forall l \in \{1, \ldots, n\}$, $\mathrm{w}_l \geq 0$ and $\sum_{l=1}^{n} \mathrm{w}_l = 1$. The Augmented Lagrangian method [Pedregal, 2004] is used to transform the constrained nonlinear optimization problem into a sequence of unconstrained nonlinear optimization problems, which are solved by the Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm [Avriel, 2003].

The supervised posterior for LMG using Leave-One-Out cross validation is

$$\hat{P}_{LMG}(\mathrm{w}|T) = \prod_{\langle \mathbf{x},y \rangle \in T} \left(\frac{\sum_{i=1}^{n} \mathrm{w}_i \hat{P}_i^{LOO}(\mathbf{x}, y)}{\sum_{y \in Y} \sum_{i=1}^{n} \mathrm{w}_i \hat{P}_i^{LOO}(\mathbf{x}, y)} \prod_{i=1}^{n} \mathrm{w}_i \right),$$

where

$$\hat{P}_i^{LOO}(\mathbf{x}, y) = \hat{P}(x_i, y) \prod_{j=1}^{n} \hat{P}(x_j | x_i, y),$$

which is estimated by excluding instance $\langle \mathbf{x}, y \rangle$ from T.

MAPLMG estimates $P(y, \mathbf{x})$ by

$$\hat{P}(y, \mathbf{x}) = \sum_{i=1}^{n} \mathrm{w}_i \hat{P}(y, x_i) \prod_{j=1}^{n} \hat{P}(x_j \mid y, x_i).$$

At training time, MAPLMG first estimates the generative probabilities for the instances left out. It generates a three-dimensional table of probability estimates and stores the training data to perform Leave-One-Out cross validation, space complexity of $O(tn + k(nv)^2)$. Since we need to go through the training data and consider each

parent and child attribute pair within each class, the estimation process has time complexity of $O(tkn^2)$. The second step is to maximize the supervised posterior for LMG, which has space complexity of $O(n^2)$ and time complexity of $O(tknI)$, where I is the upper limit of the number of iterations. Therefore, the overall time complexity is $O(tkn^2 + tknI)$. At classification time, MAPLMG has identical time and space complexity to AODE.

3.2.3 Applying NB to a Subset of the Training Set

Another effective approach to accommodating violations of the attribute conditional independence assumption is to apply NB to a subset of the training set, as it is possible that the assumption, although is violated in the whole training set, may hold or approximately hold in a subset of the training set. As already discussed, this group and the second group that alters NB by allowing interdependencies between attributes are not mutually exclusive. NBTree and LBR use local NBs to classify an instance and can also be classified into this group.

3.2.3.1 Recursive Bayesian Classifiers

Recursive Bayesian Classifiers (RBC) [Langley, 1993] forms NB on the training data, then partitions the training data, placing each instance in a partition corresponding to the class that NB assigns it. This process is repeated recursively on each partition forming a tree, until each leaf partition contains only instances from one class. At classification time, NB is applied to the test instance to direct it down one branch of the tree. Then the NB formed at the appropriate partition is applied, and so on, until a leaf is reached, at which point the NB for the leaf is applied to obtain a classification. While results on artificial data were promising, the reported experimental results for RBC on natural data sets are disappointing. As a result the technique has received little attention.

3.2.3.2 Locally Weighted Naive Bayes

Inspired by locally weighted linear regression [Cleveland, 1979; Atkeson, Moore and Schaal, 1997; Loader, 1999; Hastie et al., 2001], Frank, Hall and Pfahringer incorporate locally weighted learning into NB [Frank et al., 2003]. Locally Weighted Naive

Bayes (LWNB), at classification time, assigns a weight to each instance in the training set and applies NB to the subset of the training set in which all weights of instances are greater than zero. The instance weights decrease linearly with the Euclidean distance to the test instance and the number of instances in the subset is determined by a user-specified parameter h. The Euclidean distance of the test instance to the ith nearest neighbor is denoted as d_i. Before the distance is calculated, quantitative attributes are assumed to be normalized ($0 \leq x_i \leq 1$) and qualitative attributes be binarized. LWNB assigns the weight $\mathrm{w}_i = f(d_i/d_h)$ to the ith instance, where f is a weighting function:

$$f(x) = \begin{cases} 0 & \text{if } x > 1 \\ 1 - x & \text{otherwise} \end{cases}.$$

Let h' be the number of instances whose distances are less than d_h. LWNB applies NB to the training set with rescaled weights w'_i which is calculated by

$$\mathrm{w}'_i = \frac{\mathrm{w}_i \times h'}{\sum_{j=1}^{t} \mathrm{w}_j}.$$

With these rescaled weights, the total weight in the subset in which all weights are greater than zero is approximately h. That is, $\sum_{i=1}^{h'} \mathrm{w}'_i \approx h$.

In the subset, the frequency of the ith class is:

$$F(c_i) = \sum_{l=1}^{h'} \mathrm{w}'_l E(c_i, c_l),$$

where $E(a, b)$ is one if $a = b$ and zero otherwise.

The frequency of x_j (the value of the jth attribute in the test instance) given class c_i is:

$$F(x_j \,|\, c_i) = \sum_{l=1}^{h'} \mathrm{w}'_l E(c_i, c_l) E(x_j, x_j^l),$$

where x_j^l is the value of the jth attribute in the lth instance. LWNB uses (3.4) to estimate $P(y, \mathbf{x})$.

At training time, as LWNB only stores the training data, it has time and space complexity of $O(tn)$. At classification time, the time complexity of computing distances is $O(tn)$ if a linear search is performed.[3] Estimating class probability and conditional attribute-value probability has time complexity of $O(h'n)$. To classify the test example requires time of order $O(kn)$. Hence, the classification time complexity is $O(tn + kn)$. The space complexity is $O(tn + knv)$.

3.2.4 Performing Corrections to NB's Probability Estimations

The distortion of probabilities that NB's independence assumption produces might be corrected by making adjustments to the class probabilities or the attribute conditional probabilities. This line of reasoning leads to the fourth group of methods, which modify the probability outcome of NB.

3.2.4.1 Adjusted Probability Naive Bayesian Classification

Adjusted Probability Naive Bayesian Classification (APNB) [Webb and Pazzani, 1998] applies linear adjustments to the class probabilities. In the two class case, it only needs to find an adjustment for one of the classes. As the adjustment for one class will influence the adjustments for other classes in the multiple class case, APNB uses a simple hill-climbing search to find adjustments that maximize resubstitution accuracy.

If an instance $\mathbf{x}$ of class c_i is misclassified as class c_j by APNB with the current vector of adjustments Adj, there are two possible adjustments to correct that misclassification. One is an upward adjustment, which multiplies the original probability estimate for class c_i by an adjustment $> \frac{Adj_{c_j}\hat{P}(c_j|\mathbf{x})}{\hat{P}(c_i|\mathbf{x})}$, where $\hat{P}(c_j \mid \mathbf{x})$ and $\hat{P}(c_i \mid \mathbf{x})$ are the probability estimates produced by NB, and Adj_{c_z} is the adjustment for class c_z. Another is a downward adjustment, which multiplies the probability of class c_j with an adjustment $< \frac{Adj_{c_i}P(c_i|\mathbf{x})}{P(c_j|\mathbf{x})}$. APNB selects a value slightly above or below the bounds implied by these ranges, a small value (10^{-5}) being added to the relevant

[3] When space-partitioning methods, such as KD-Tree, are performed, less time is required for computing distances.

bound for upward adjustments and subtracted from the relevant bound for downward adjustments. If there is a significant accuracy improvement by using a upward or downward adjustment, another bound value for the adjustment is computed. For the upward adjustment, the upper bound value is the lowest value greater than the lower bound value, that has a higher resubstitution error than the lower bound value. For the downward adjustment, the lower bound value is the highest value less than the upper bound value, that has a higher resubstitution error than the upper bound value. The adjustment is replaced by the midpoint of the two bound values. This process is repeated until there is no accuracy improvement that passes a binomial sign test for significance at the 0.05 level.

APNB estimates $P(y, \mathbf{x})$ by

$$\hat{P}(y, \mathbf{x}) = Adj_y \hat{P}(y) \prod_{i=1}^{n} \hat{P}(x_i \mid y).$$

At training time APNB generates a one-dimensional table of class probability estimates and a two-dimensional table of conditional attribute-value probability estimates. It also stores the training data to perform leave-one-out cross validation. Hence, the space complexity is $O\big(tn + knv\big)$. In the worse case, there are $O\big(t\big)$ misclassified instances, and each possible adjustment for the instance has time complexity of $O\big(tk\big)$. This process is repeated once for each class to find a single adjustment that maximizes resubsitution accuracy. Therefore, the time complexity is $O\big(t^2k^2\big)$. It has identical time and space complexity to NB at classification time.

3.2.4.2 Iterative Bayes

Iterative Bayes (IB) [Gama, 2003] starts with the conditional attribute-value frequency table generated by NB, indexed by class and attribute-value, and iteratively updates the frequency table by cycling through all the training examples.

At each iteration, all the examples in the training set are classified by NB using the current frequency tables. The conditional attribute-value frequency table is updated through each training example. The adjustment (indicated as adj) for an example $\mathbf{x}$ is

$$\frac{1.0 - \hat{P}(y_i \,|\, \mathbf{x})}{k},$$

where y_i is the predicted class. If the example is correctly classified, adj is added to the entries $\langle y_i, x_j \rangle$, $1 \leq j \leq n$, and $adj/(k-1)$ is subtracted from the entries $\langle y_l, x_j \rangle$, $1 \leq l \leq k$, $l \neq i$ and $1 \leq j \leq n$. If the example is misclassified, adj is subtracted from the entries $\langle y_i, x_j \rangle$, $1 \leq j \leq n$, and $adj/(k-1)$ is added to the entries $\langle y_l, x_j \rangle$, $1 \leq l \leq k$, $l \neq i$ and $1 \leq j \leq n$. This process is terminated when the number of iterations exceeds 10 or the following evaluation function increases:

$$\frac{1}{t} \sum_{i=1}^{t} \left(1.0 - \operatorname*{argmax}_{y} \hat{P}(y \,|\, \mathbf{x}_i) \right),$$

where $\mathbf{x}_i$ is the ith example. It uses (3.4) to estimate $P(y, \mathbf{x})$.

At training time, IB generates a one-dimensional table of class probability estimates and a two-dimensional table of conditional attribute-value probability estimates as NB does. It also store the training data, with additional space complexity $O(tn)$. At each iteration, to update conditional frequency table requires time of order $O(tkn)$, as we need to adjust each entry for every combination of the classes and attribute-values for every example, and to perform classification for all examples also requires time of order $O(tkn)$. Therefore, the total training time complexity is $O(tkn)$. At classification time, it has identical time and space complexity to NB.

3.2.5 Complexity Summary

Table 3.3 summarizes the complexity of the algorithms discussed. We display the time complexity and the space complexity of each algorithm for each of training time and classification time.

The training time complexity of BSEJ and SP-TAN is cubic in the number of attributes, that of NBTree is quadratic in the number of instances and attributes and that of APNB is quadratic in the number of instances and classes. Hence, BSEJ and SP-TAN has very high training time if the number of attributes is large, NBTree if the number of instances and attributes are large and APNB if the number of instances and classes are large. The classification time complexity of these four algorithms is

Table 3.3: Computational complexity

	Training		**Classification**	
Algorithm	**Time**	**Space**	**Time**	**Space**
NB	$O(tn)$	$O(knv)$	$O(kn)$	$O(knv)$
BSE	$O(tkn^2)$	$O\big(tn + tk + knv\big)$	$O(kn)$	$O(knv)$
FSS	$O(tkn^2)$	$O\big(tn + tk + knv\big)$	$O(kn)$	$O(knv)$
BSEJ	$O(tkn^3)$	$O(tn + kv^n)$	$O(kn)$	$O(kv^n)$
TAN	$O(tn^2 + k(nv)^2 + n^2 logn)$	$O(k(nv)^2)$	$O(kn)$	$O(knv^2)$
SP-TAN	$O(tkn^3)$	$O(tn + k(nv)^2)$	$O(kn)$	$O(knv^2)$
NBTree	$O(t^2kn^2/v)$	$O(tkv(n - log_v t))$	$O(kn)$	$O(tkv(n - log_v t))$
LBR	$O(tn)$	$O(tn)$	$O(tkn^3)$	$O(tn + knv)$
AODE	$O(tn^2)$	$O(k(nv)^2)$	$O(kn^2)$	$O(k(nv)^2)$
MAPLMG	$O(tkn^2 + tknI)$	$O(tn + k(nv)^2)$	$O(kn^2)$	$O(k(nv)^2)$
LWNB	$O(tn)$	$O(tn)$	$O(tn + kn)$	$O(tn + knv)$
APNB	$O(tn + t^2k^2)$	$O(tn + knv)$	$O(kn)$	$O(knv)$
IB	$O(tkn)$	$O(tn + knv)$	$O(kn)$	$O(knv)$

k is the number of classes

n is the number of attributes

t is the number of training examples

v is the mean number of values for an attribute

I is the upper limit of the number of iterations for BFGS

linear in the number of classes and attributes. Therefore, once models are generated, these methods can classify test instances efficiently.

LBR and LWNB have identical training time complexity to NB, and hence they are highly efficient when few instances are to be classified. However, the high classification time complexity of LBR, $O\big(tkn^3\big)$, hampers its application when large numbers of instances are to be classified. The classification time complexity of LWNB, $O\big(tn + kn\big)$, is higher than that of the other methods except LBR, AODE and MAPLMG. As t is usually considerably greater than k and n, LWNB has substantially higher classification time relative to AODE and MAPLMG in most cases.

The training time complexity of MAPLMG, BSE and FSS is relatively higher than that of TAN, AODE and IB, whose training time complexities are moderate. AODE and MAPLMG have high classification time when the number of attributes are large, for example, in text classification. However, for many classification tasks with moderate or small number of attributes, the classification time complexity of AODE and MAPLMG is modest.

BSEJ has very high training space complexity, which is an exponential function of the number of attributes.

3.2.6 Bayesian Network Perspective

From the Bayesian Network perspective, the methods discussed can be classified into three groups, as depicted in Figure 3.1. NB, BSE, FSS, BSEJ, LWNB, APNB and IB are 0-dependence classifiers, since they only allow each attribute to depend on the class. TAN, SP-TAN, AODE and MAPLMG belong to the second group, which allows each attribute to depend on the class and at most one other attribute. For instance, in graph (b), attribute X_2 and X_i depend on attribute X_1, and X_{i+1} depends on X_i and so forth. AODE and MAPLMG are special types of 1-dependence classifiers, in which all the attributes depend on the class and the SuperParent, such as attribute X_1 in graph (c). The third group assumes independence among fewer attributes by allowing each attribute to depend on the class and at most z ($z \geq 0$) other attributes. In graph (d), independence is assumed among attributes in $O = \{X_{i_{q+1}}, \ldots, X_{i_n}\}$ given the class, and these attributes depend on all the attributes in $P = \{X_{i_1}, \ldots, X_{i_q}\}$. For NBTree, P is the set of the splitting attributes on the path leading to the leaf, and

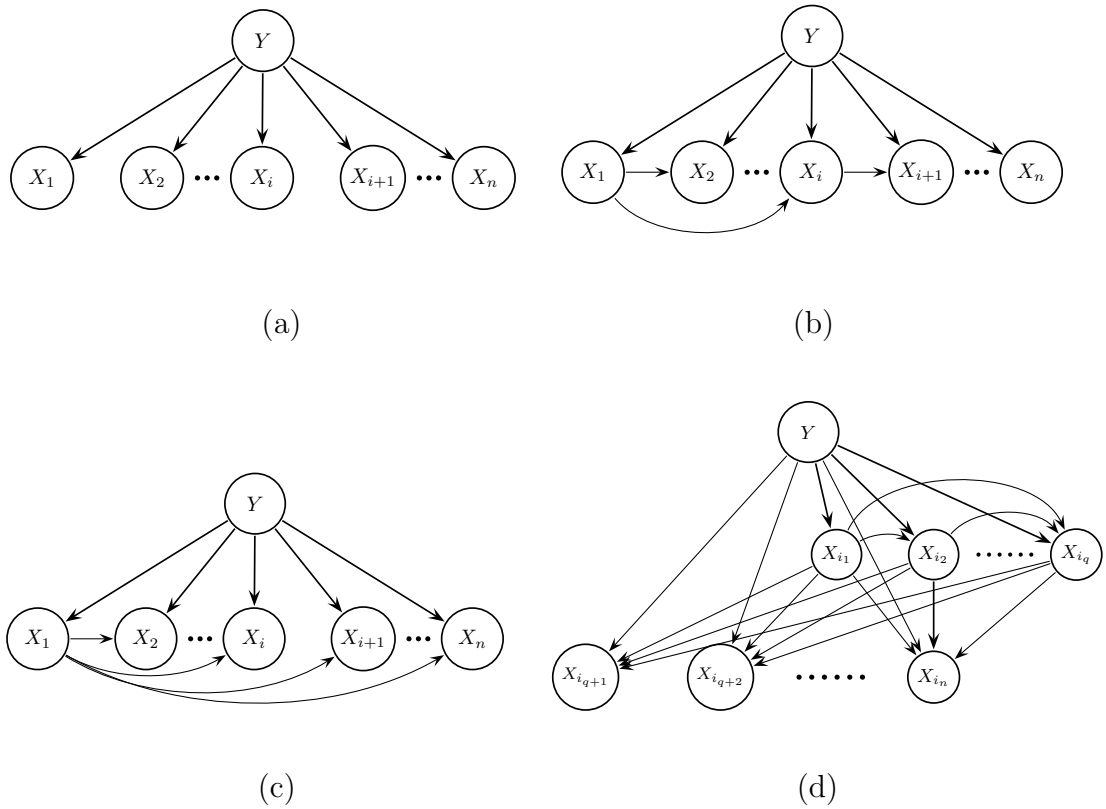


Figure 3.1: Bayesian Network (a) 0-*dependence classifier*, (b) 1-*dependence classifier*, (c) 1-*dependence classifier (SuperParent)*, and (d) z-*dependence classifier* ($z \geq 0$)

O is the set of leaf attributes. For LBR, P is the set of attributes in the antecedent, and O is the set of attributes in the consequent.

3.3 Comparison of Fifteen Methods

In this section, the performance of NB and twelve semi-naive Bayesian algorithms, BSE, FSS, BSEJ, TAN, SP-TAN, NBTree, LBR, AODE, MAPLMG,LWNB, APNB and IB will be analyzed in terms of classification error, bias, variance, root mean squared error, training time and classification time on the sixty natural domains from the UCI Repository of machine learning [Newman, Hettich, Blake and Merz, 1998].

To provide a baseline for comparison, we also compare these methods to logistic regression and LibSVM with parameter search.

3.3.1 Data Sets

Table 3.4 summarizes the characteristics of each data set used in this thesis, including the number of the instances, attributes and classes. We augment those widely used natural data sets in the literature [Langley and Sage, 1994; Pazzani, 1996; Domingos and Pazzani, 1996; Zheng and Webb, 2000; Webb et al., 2005] by twenty-three data sets. They represent a wide range of practical problems.

Connect-4 Opening is the largest data set in our data collection. It has 67557 instances with 42 attributes, each corresponding to one connect-4 square on the seven-column, six-row board. The class has 3 values: *win*, *loss* and *draw*. Each attribute has 3 values: x (player x has taken the square), o (player o has taken the square) and b (the square is blank). Each instance is a legal 8-play position in which neither player x or player o has won and the next move is not forced.

Audiology, whose task is to diagnose hearing problems, has the largest number of attributes (69) in our data collection. It has 226 instances and 24 classes. Nettalk (Phoneme) has the largest number of classes. There are 52 classes, 5438 instances and 7 attributes in Nettalk.

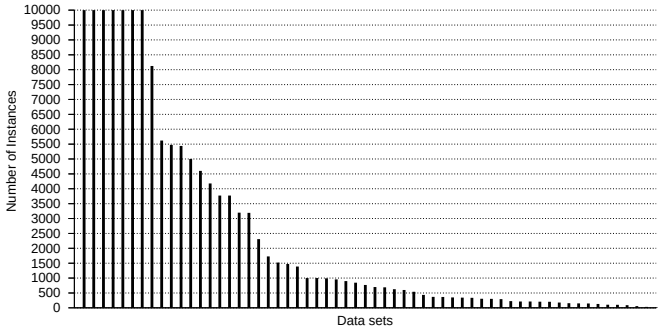
Figure 3.2 presents the number of instances, attributes and classes of the 60 data sets. Data sets are respectively sorted in descending order on the numbers of instances, attributes and classes in Figures 3.2 (a), (b) and (c). To scale Figure 3.2 (a) appropriately, we cut short the bars of 7 data sets which have more than 10000 instances. They are Connect-4 Opening (67557 instances), Adult (48842 instances), Letter Recognition (20000 instances), Magic Gamma Telescope (19020 instances), Nursery (12960 instances), Sign (12546 instances) and Pen Digits (10992 instances).

From Figure 3.2 (a) we can see that 11 data sets have more than 5000 instances, 23 data sets have more than 1000 instances and 35 data sets have more than 500 instances.

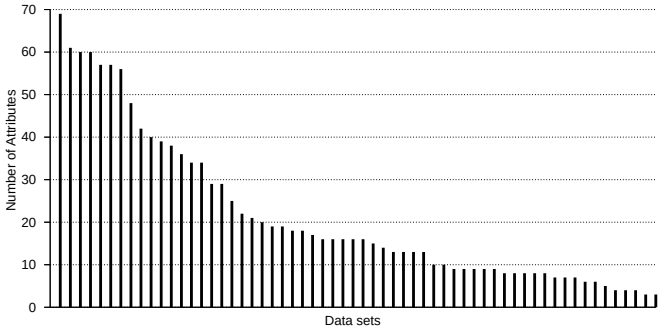
As shown in Figure 3.2 (b), 7 data sets, Audiology, Lung Cancer, Promoter Gene Sequences, Sonar Classification, SPAM E-mail, Splice-junction Gene Sequences and Syncon, have more than 50 attributes, 15 data sets have more than 30 attributes and 39 data sets have more than 10 attributes. All data sets have at least 4 attributes.

Table 3.4: Data sets used for experiments

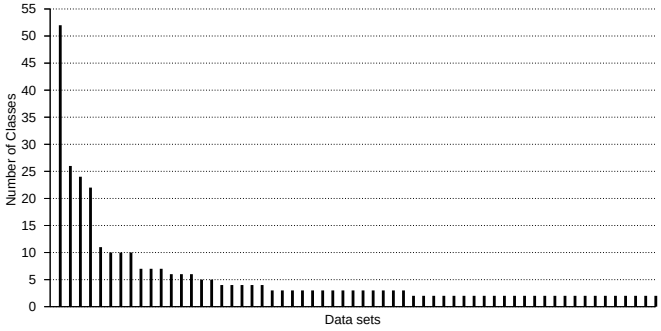
No.	Domain	Case	Att	Class	No.	Domain	Case	Att	Class
1	Abalone	4177	8	3	31	Liver Disorders (Bupa)	345	6	2
2	Adult	48842	14	2	32	Lung Cancer	32	56	3
3	Annealing	898	38	6	33	Lymphography	148	18	4
4	Audiology	226	69	24	34	Magic Gamma Telescope	19020	10	2
5	Auto Imports	205	25	7	35	Mushrooms	8124	22	2
6	Balance Scale	625	4	3	36	Nettalk(Phoneme)	5438	7	52
7	Breast Cancer (Wisconsin)	699	9	2	37	New-Thyroid	215	5	3
8	Car Evaluation	1728	6	4	38	Nursery	12960	8	5
9	Connect-4 Opening	67557	42	3	39	Optical Digits	5620	48	10
10	Contact-lenses	24	4	3	40	Page Blocks	5473	10	5
11	Contraceptive Method Choice	1473	9	3	41	Pen Digits	10992	16	10
12	Credit Screening	690	15	2	42	Pima Indians Diabetes	768	8	2
13	Cylinder Bands	540	39	2	43	Postoperative Patient	90	8	3
14	Dermatology	366	34	6	44	Primary Tumor	339	17	22
15	Echocardiogram	131	6	2	45	Promoter Gene Sequences	106	57	2
16	German	1000	20	2	46	Segment	2310	19	7
17	Glass Identification	214	9	3	47	Sick-euthyroid	3772	29	2
18	Haberman's Survival	306	3	2	48	Sign	12546	8	3
19	Heart Disease (Cleveland)	303	13	2	49	Solar Flare	1389	9	2
20	Hepatitis	155	19	2	50	Sonar Classification	208	60	2
21	Horse Colic	368	21	2	51	SPAM E-mail	4601	57	2
22	House Votes 84	435	16	2	52	Splice-junction Gene Sequences	3190	61	3
23	Hungarian	294	13	2	53	Syncon	600	60	6
24	Hypothyroid(Garavan)	3772	29	4	54	Tic-Tac-Toe Endgame	958	9	2
25	Ionosphere	351	34	2	55	Vehicle	846	18	4
26	Iris Classification	150	4	3	56	Volcanoes	1520	3	4
27	King-rook-vs-king-pawn	3196	36	2	57	Vowel	990	13	11
28	Labor Negotiations	57	16	2	58	Waveform-5000	5000	40	3
29	LED	1000	7	10	59	Wine Recognition	178	13	3
30	Letter Recognition	20000	16	26	60	Zoo	101	16	7



(a) Number of Instances



(b) Number of Attributes



(c) Number of Classes

Figure 3.2: The number of instances, attributes and classes of 60 data sets

There are 4 data sets, Nettalk, Audiology, Letter Recognition and Primary Tumor, that have more than 20 classes, 14 data sets have more than 5 classes and 25 data sets have 2 classes.

3.3.2 Experimental Methodology

The experiments in this thesis compare algorithms implemented in the Weka workbench (version 3-5-2) [Witten and Frank, 2005] on the data sets described in Section 3.3.1. Each algorithm is tested on each data set using a 50-run 2-fold cross validation. The Friedman test and Nemenyi test with 0.05 level of significance are employed to evaluate the performance of algorithms, including classification error, bias, variance, RMSE, training time and classification time.

Experiments on the algorithms except LBR and LibSVM were performed on a dual-processor 1.7 GHz Pentium 4 Linux computer with 2 Gb RAM. LBR and LibSVM were executed on a Linux Cluster based on Xeon 2.8 GHz CPUs.

3.3.2.1 Two-Fold Cross-Validation Bias-Variance Estimation

Since bias and variance are important factors that influence the classification error of a learning algorithm, bias-variance decomposition is widely used to investigate the performance of learning algorithms. There are a number of different bias-variance decomposition definitions [Kong and Dietterich, 1995; Breiman, 1996; Kohavi and Wolpert, 1996; Friedman, 1997; Webb, 2000]. In this thesis, we use the bias and variance definitions of Kohavi and Wolpert (1996) together with the repeated cross-validation bias-variance estimation method proposed by Webb (2000).

In order to maximize the variation in the training data from trial to trial we use two-fold cross validation. Figure 3.3 illustrates the estimation process. The training data is first randomized. Then, it is randomly divided into two folds, $fold_1^i$ and $fold_2^i$. $Classifier_1^i$ is generated from $fold_1^i$ and $Classifier_2^i$ is generated from $fold_2^i$. $fold_1^i$ and $fold_2^i$ are respectively used as a test set for $Classifier_2^i$ and $Classifier_1^i$. In this manner, each available instance is classified once for each two-fold cross-validation. In order to give a more accurate estimation of the average performance of an algorithm, this process is repeated 50 times which produces 100 folds. Bias, variance, error and

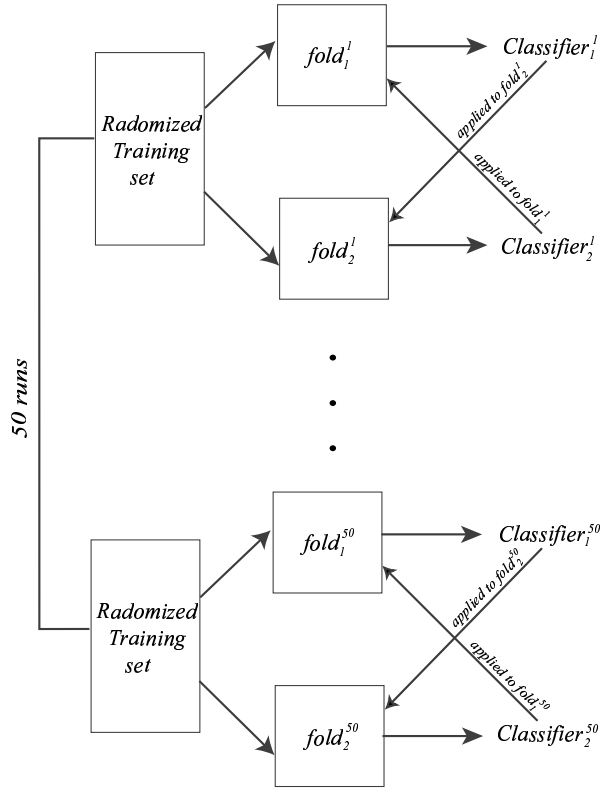


Figure 3.3: Two-fold cross-validation bias-variance estimation (50 runs)

RMSE are estimated by evaluation of the predictions of $Classifier_1^i$ and $Classifier_2^i$ when applied to $fold_2^i$ and $fold_1^i$ for $1 \leq i \leq 50$.

In Kohavi and Wolpert's method, the default bias-variance estimation method in Weka [Kohavi and Wolpert, 1996], the randomized training data are divided into a training pool and a test pool randomly. Each pool contains 50% of the data. 50 (the default number in Weka) local training sets, each containing half of the training pool, are sampled from the training pool. Hence, each local training set is only 25% of the full data set. Classifiers are generated from local training sets and bias, variance and error are estimated from the performance of the classifiers on the test pool. Figure 3.4 illustrates this process.

The repeated cross-validation bias-variance estimation method is preferred to Kohavi and Wolpert's method as it results in the use of substantially larger training

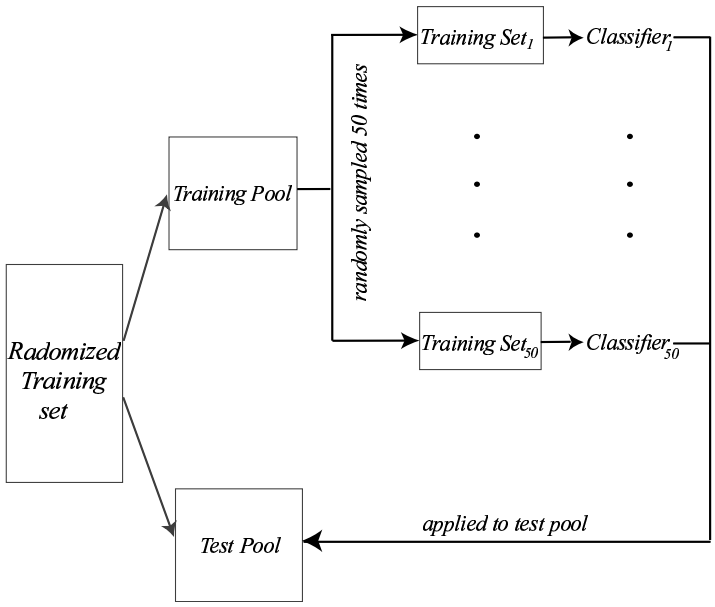


Figure 3.4: Kohavi and Wolpert's bias-variance estimation (50 runs).

sets. In addition, every case in the training data is used the same number of times for both training and testing.

3.3.2.2 Probability Estimates

The base probabilities of each algorithm, except TAN, Logistic Regression and LibSVM, are estimated using Laplace estimation [Cestnik, 1990]. [4]

In TAN, a smoothing operation was introduced and the conditional probability of x_i given its parents, $\pi(x_i)$, is calculated by

$$\hat{P}^s(x_i|\pi(x_i)) = \alpha F(x_i|\pi(x_i)) + (1-\alpha)F(x_i),$$

where $\alpha = \frac{tF(\pi(x_i))}{tF(\pi(x_i))+s}$ and $s = 5$ [Friedman et al., 1997]. It also uses Laplace estimation to estimate $P(y)$.

[4] In keeping with Weka's default probability estimation method, we use the Laplace estimation to adjust probability estimates.

3.3.2.3 Numeric Values and Missing Values

For all algorithms except Logistic Regression and LibSVM, quantitative attributes are discretized using MDL discretization [Fayyad and Irani, 1993]. In keeping with Weka's Logistic Regression, missing values for qualitative attributes are replaced with modes and those for quantitative attributes are replaced with means from the training data.

3.3.2.4 Statistics Employed

We use the Friedman test and the Nemenyi test to compare the performance of multiple algorithms and Win/Draw/Loss record to compare the performance of two algorithms. Mean metrics across all data sets are also employed to provide a simplistic overall measure of relative performance. In addition, we use the Spearman's rank correlation test to examine whether the bias proportion of error on large data sets is higher than that on small data sets.

Friedman Test. Demšar (2006) recommends the Friedman test [Friedman, 1937; Friedman, 1940] for comparisons of multiple algorithms over multiple data sets. It first calculates the ranks of algorithms for each data set separately (average ranks are assigned if there are tied values), and then compares the average ranks of algorithms over data sets. The null-hypothesis is that there is no difference in average ranks. If the null-hypothesis is rejected then it is probable that there is a true difference in the average ranks of at least two algorithms. Post-hoc tests are used to determine which pairs of algorithms have significant differences. Let R_i be the average rank of ith algorithm over data sets. The Friedman statistic derived by Iman and Davenport (1980) is

$$F_F = \frac{(D-1)\chi_F^2}{D(a-1)-\chi_F^2},$$

where

$$\chi_F^2 = \frac{12D}{a(a+1)}\left(\sum_i R_i^2 - \frac{a(a+1)^2}{4}\right),$$

a is the number of algorithms and D is the number of data sets. The Friedman statistic is distributed according to the F distribution with $a-1$ and $(a-1)(D-1)$ degrees of freedom. We reject the null-hypothesis if F_F is lager than the critical value of $F(a-1,(a-1)(D-1))$ for $\alpha = 0.05$.

Nemenyi Test. If the Friedman test rejects the null-hypothesis, the Nemenyi test is used to further analyze which pairs of algorithms are significant different. Let d_i^j be the difference between ith algorithm and jth algorithm. We assess a difference between ith algorithm and jth algorithm as significant if $d_i^j >$ Critical Difference (CD):

$$CD = q_{0.05}\sqrt{\frac{a(a+1)}{6D}},$$

where $q_{0.05}$ are the critical values that are calculated by dividing the values in the row for the infinite degree of freedom of the table of Studentized range statistics ($\alpha = 0.05$) by $\sqrt{2}$.

Win/Draw/Loss Record. When two algorithms are compared, we count the number of data sets for which one algorithm performs better, equally or worse to the other on a given measure. A standard binomial sign test, assuming that wins and losses are equiprobable, is applied to these records. We assess a difference as significant if the outcome of a one-tailed binomial sign test is less than 0.05.

Mean. The arithmetic mean across all data sets provides a gross indication of relative performance and adjunct to other statistics.

Spearman's Rank Correlation Test. We use this test to assess whether there is a relationship between the mean bias proportion of error across large data sets and small data sets over a algorithms. Mean values across large and small data sets are separately converted to ranks (average rank is used if two or more values are equal). Let d_i be the difference between the ranks on the ith out of a algorithms.

The Spearman's rank correlation coefficient is

$$r_s = 1 - \frac{6 \sum_{i=1}^{a} d_i^2}{a(a^2 - 1)}.$$

We reject the null-hypothesis that there is no relationship between the mean values across large and small data sets if r_s is greater than the critical value for a two-tailed Spearman's rank correlation test at $\alpha = 0.05$.

3.3.2.5 Logistic Regression and LibSVM

We use Weka's implementation and default setting of logistic regression. The results of LibSVM using Weka's implementation and default setting with the exception of turning on normalization of data (recommended in [Hsu, Chang and Lin, 2007]) are poor on many data sets. For instance, the errors on 2 data sets, Pen Digits and Syncon, are greater than 0.8, and errors on 13 data sets are greater than 0.5. This necessitates the use of parameter search. We perform a "grid-search" on C and γ for the RBF kernel using 5-fold cross-validation [Hsu et al., 2007]. Each pair of (C, γ) is tried ($C = 2^{-5}, 2^{-3}, \ldots, 2^{15}, \gamma = 2^{-15}, 2^{-13}, \ldots, 2^{3}$), and the one with the lowest cross-validation error is selected. However, this process has high time complexity.

3.3.2.6 Locally Weighted Naive Bayes

We use Weka's implementation of LWNB and set the number of neighbors to 50, as this number is favorable to LWNB [Frank et al., 2003].

3.3.3 Experimental Results

As LBR has very high classification time complexity, the results of LBR on the three largest data sets (Connect-4 Opening, Adult and Letter Recognition) are obtained from five runs of two-fold cross-validation.

BSEJ has a memory shortage problem on five data sets (Adult, Letter Recognition, Segment, Spam E-mail and Cylinder-bands). To estimate the results of BSEJ, we average the results of $i - 1$ iterations if the ith run is the first iteration that has the problem.

Due to the high time complexity of "grid-search" on C and γ for the RBF kernel, the results of LibSVM on Letter Recognition, MAGIC Gamma Telescope, Pen Digits, Optical Digits, Waveform-5000 and SPAM E-mail are obtained from five runs of two-fold cross-validation. In addition, at the time of writing, we still have not obtained the results of LibSVM on Adult and Connect-4 Opening, even by using two runs of two-fold cross-validation.

We first compare NB and the 12 semi-naive Bayesian methods discussed using the Friedman and Nemenyi tests on the 60 data sets. Then, we select five semi-naive Bayesian algorithms, all of which enjoy significant error advantage relative to NB, to compare with logistic regression and LibSVM on 58 data sets (exclude Adult and Connect-4 Opening). We do not compare NB and all the 12 semi-naive Bayesian algorithms to logistic regression and LibSVM, as the power of the Friedman and Nemenyi tests is low with a large number of algorithms and hence 58 data sets are not sufficient to differentiate 15 algorithms.

Following the graphical presentation of Demšar, we show the comparison of algorithms against each other with the Nemenyi test on each metric. We plot the algorithms on the left line according to their average ranks, which are indicated on the parallel right line. Critical Difference (CD) is also presented in the graphs. The lower the position of algorithms, the lower the ranks they obtain, and hence the better the performance. Algorithms are connected by a line if their differences are not significant. Detailed error, bias, variance and RMSE by data sets are presented in the Appendix A.

3.3.3.1 Comparison of NB and the 12 Semi-naive Bayesian Algorithms on 60 Data Sets

With 13 algorithms and 60 data sets, the Friedman statistic is distributed according to the F distribution with $a-1 = 13-1 = 12$ and $(a-1)(D-1) = (13-1)*(60-1) = 708$ degrees of freedom. The critical value of $F(12, 708)$ for $\alpha = 0.05$ is 1.7658. The Friedman statistics for error, bias, variance and RMSE in our experiments are 9.2524, 21.7209, 20.1314 and 13.7670 respectively, and hence we reject all the null-hypotheses.

We show the comparison of the 13 algorithms against each other using the Nemenyi test on error, bias, variance and RMSE in Figures 3.5, 3.6, 3.7 and 3.8 respectively. With 13 algorithms and 60 data sets, the Critical Difference for $\alpha = 0.05$ is CD

$= 3.313 * \sqrt{a(a+1)/(6*D)} = 3.313 * \sqrt{13(13+1)/(6*60)} = 2.3556$. Since the comparison involves 13 algorithms, the power of the Nemenyi test is low and so only large effects are likely to be apparent.

Error

MAPLMG achieves the lowest mean error rank (4.5250). LBR comes next and AODE obtains the third lowest mean error rank (5.2083 and 5.5083). FSS and NB have the

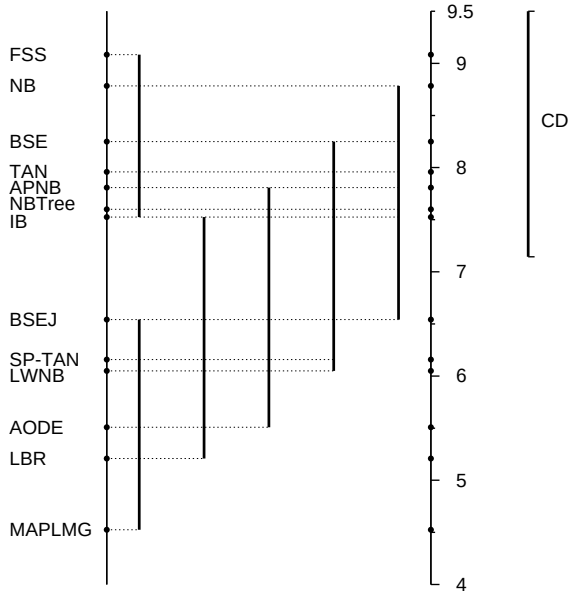


Figure 3.5: Error comparison of NB and 12 semi-naive Bayesian algorithms with the Nemenyi test on 60 data sets. CD = 2.3556.

highest and the second highest mean rank (9.0833 and 8.7833).

The Nemenyi test indicates that MAPLMG enjoys a significant advantage over all the other algorithms except LBR, AODE, LWNB, SP-TAN and BSEJ. LBR outperforms NBTree, APNB, TAN, BSE, NB and FSS, and share a similar level of error with MAPLMG, AODE, LWNB, SP-TAN, BSEJ and IB. The error differences between AODE and TAN, BSE, NB and FSS are significant. LWNB and SP-TAN have a significantly lower mean error rank compared with NB and FSS. They have a lower mean error rank compared to BSEJ, IB, NBTRee, APNB, TAN and BSE, but these differences are not significant.

FSS has a higher mean error rank compared to all the other algorithms, but the difference is not significant when it is compared to NB, BSE, TAN, APNB, NBTree and IB. Five semi-naive Bayesian algorithms, MAPLMG, LBR, AODE, LWNB and SP-TAN outperform NB on the Nemenyi test.

Bias

NBTree comes out ahead when mean bias ranks are compared (rank being 3.8833). It enjoys a significant advantage over all the other algorithms except LWNB, BSEJ and LBR.

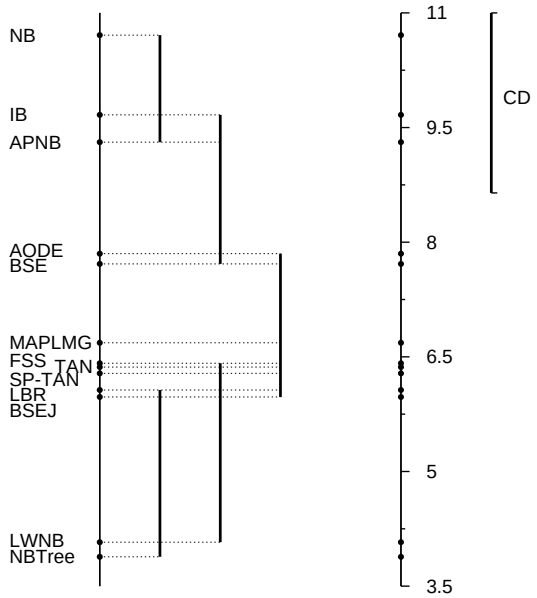


Figure 3.6: Bias comparison of NB and 12 semi-naive Bayesian algorithms with the Nemenyi test on 60 data sets. CD = 2.3556.

LWNB achieves the second lowest mean rank (4.0750) and shares a similar level of bias with NBTree, BSEJ, LBR, SP-TAN, TAN and FSS. There are small differences between BSEJ and LBR, and SP-TAN, TAN and FSS. BSEJ, LBR, SP-TAN, TAN, FSS and MAPLMG have a significant advantage over APNB, IB and NB. However, the differences between these algorithms and BSE and AODE are not significant. AODE and BSE have lower mean bias ranks compared with APNB, IB and NB, but only have a significant advantage in bias over NB, which has the highest mean bias

rank (10.7083). NB has a significant disadvantage relative to all the other algorithms but IB and APNB.

Variance

NB and AODE have the lowest and the second lowest mean variance rank (4.2167 and 4.4417 respectively). They enjoy a clear and consistent advantage in variance over the rest of algorithms except MAPLMG, APNB and IB.

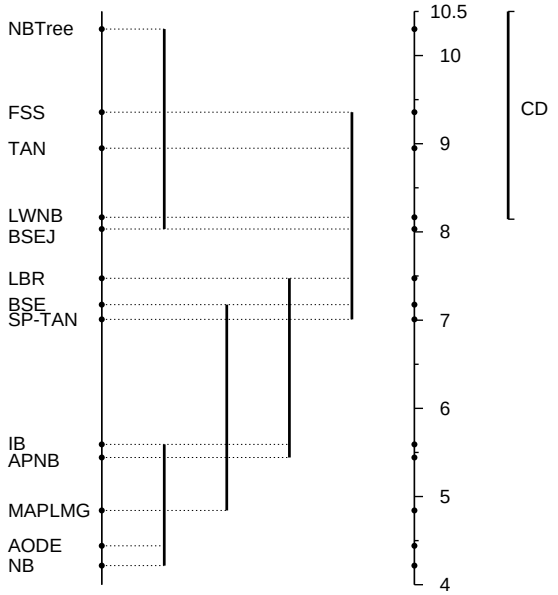


Figure 3.7: Variance comparison of NB and 12 semi-naive Bayesian algorithms with the Nemenyi test on 60 data sets. CD = 2.3556.

MAPLMG has the third lowest mean rank, shares a similar level of variance with NB, AODE, APNB, IB, SP-TAN and BSE and has a significant advantage over the remaining algorithms. APNB and IB have lower mean ranks than SP-TAN, BSE and LBR, but these differences are not significant. These two algorithms enjoy a significant advantage over BSEJ, LWNB, TAN, FSS and NBTree. SP-TAN, BSE and LBR have lower mean ranks than that of BSEJ, LWNB, TAN, FSS and NBTree, but only have a significant advantage over NBTree, which has the highest mean rank (10.3000). All the other algorithms except FSS, TAN, LWNB and BSEJ enjoy a significant advantage in variance over NBTree.

RMSE

MAPLMG achieves the lowest mean rank of RMSE (3.9333). It significantly outperforms all the rest of algorithms but AODE, LBR and SP-TAN.

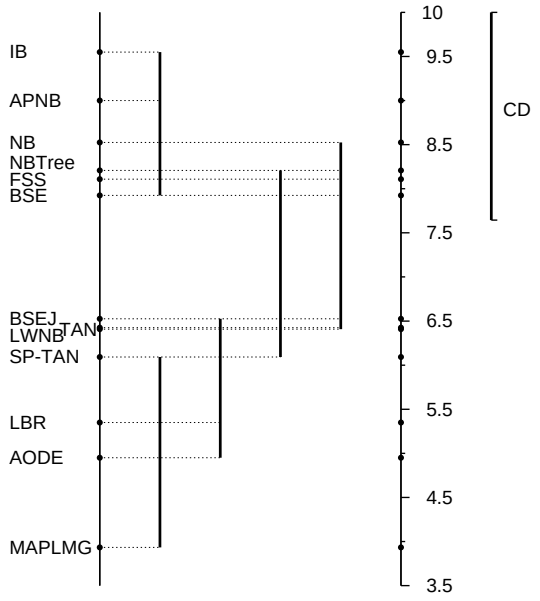


Figure 3.8: RMSE comparison of NB and 12 semi-naive Bayesian algorithms with the Nemenyi test on 60 data sets. CD = 2.3556.

AODE and LBR have lower mean ranks than SP-TAN, LWNB, TAN and BSEJ, and have a significant advantage over the other algorithms except MAPLMG. The advantage of SP-TAN relative to NB, APNB and IB is significant. There are small differences between LWNB, TAN and BSEJ, which have a substantially lower RMSE compared to APNB and IB.

IB and APNB have the highest and the second highest mean rank of RMSE (9.5500 and 9.0000). They have a significant disadvantage over all the other algorithms except NB, NBTree, FSS and BSE. Four semi-naive Bayesian algorithms, MAPLMG, AODE, LBR and SP-TAN, enjoy a significant advantage in RMSE relative to NB.

Computing Time

Some caution is required in comparing compute times for execution of implementations of alternative learning algorithms, as there is always room for uncertainty to

what extent differences in performance can be attributed to the relative efficiency with which the algorithms have been implemented as opposed to fundamental efficiencies in the actual algorithms. Nonetheless, empirical comparison of real world running time can provide a valuable adjunct to computational complexity analysis. We here present the results of such an empirical evaluation with the qualification that specific outcomes should be treated with caution.

Figures 3.9 and 3.11 provide training time and classification time comparisons of all the algorithms except LBR, which was executed on a different machine, using the Nemenyi test over 60 data sets. As BSEJ has a memory shortage problem on five data sets (Adult, Letter Recognition, Segment, Spam E-mail and Cylinder-bands), we estimate training time for each data set by multiplying the total training time on $i-1$ iterations by $50/(i-1)$ if the ith iteration is the first iteration that has the problem. Classification time is estimated by the same approach.

With 12 algorithms and 60 data sets, the Friedman statistic is distributed according to the F distribution with $a-1 = 12-1 = 11$ and $(a-1)(D-1) = (12-1)*(60-1) = 649$ degrees of freedom. The critical value of $F(11, 649)$ for $\alpha = 0.05$ is 1.8029. The null-hypotheses are rejected because the Friedman statistic for training time and classification time are 259.9665 and 203.9095 respectively. The Critical Difference using the Nemenyi test for $\alpha = 0.05$ is CD $= 3.269 * \sqrt{a(a+1)/(6*D)} = 3.269 * \sqrt{12(12+1)/(6*60)} = 2.1519$.

The mean training and classification time across 60 data sets are presented in Figures 3.10 and 3.12. These algorithms are sorted in ascending order on the mean metrics. To scale these graphs appropriately, we cut short the bars of SP-TAN in Figure 3.10 and LWNB in Figure 3.12. The mean training time of SP-TAN is 12609.89 seconds and the mean classification time of LWNB is 7560.12 seconds.

Training Time

NB and LWNB achieve the lowest and second lowest mean training time rank (1.6917 and 1.7333). Their training time is not significantly lower than that of AODE and outperforms the rest of algorithms.

AODE enjoys a clear advantage over the algorithms in which wrapper techniques are employed (BSE, APNB, FSS, BSEJ, SP-TAN, MAPLMG and NBTree). The difference between AODE and IB is also significant. It has substantially lower mean

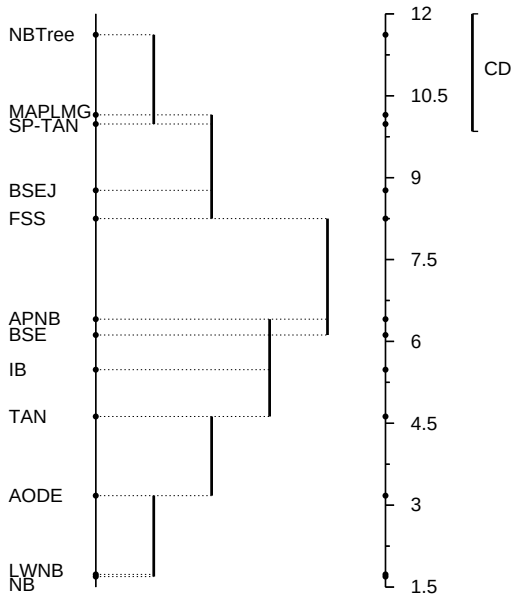


Figure 3.9: Training time comparison of NB and 11 semi-naive Bayesian algorithms (exclude LBR) with the Nemenyi test on 60 data sets. CD = 2.1519.

training time rank compared to TAN, but the difference is not significant. TAN shares a similar level of training time with AODE, IB, BSE and APNB, and has a significant advantage relative to FSS, BSEJ, SP-TAN, MAPLMG and NBTree.

IB has lower mean training time rank compared to all the algorithms using wrapper techniques, and outperforms FSS, BSEJ, SP-TAN, MAPLMG and NBTree. The difference between BSE and APNB is small. They enjoy a significant advantage over BSEJ, SP-TAN, MAPLMG and NBTree.

FSS shares a similar level of training time with BSE, APNB, BSEJ, SP-TAN and MAPLMG, and has a clear advantage over NBTree. The advantage of BSEJ compared to NBTree, but not SP-TAN and MAPLMG, is significant. NBTree has the highest mean training time rank (11.6167). All the algorithms, but MAPLMG and SP-TAN, have a significant advantage in training time over NBTree.

NBTree has very high training time on data sets with large number of instances and SP-TAN and BSEJ have very high training time on data sets with large number of attributes. On the seven largest data sets with 10000 or more instances (Connect-4 Opening, Adult, Letter Recognition, MAGIC Gamma Telescope, Nursery, Sign

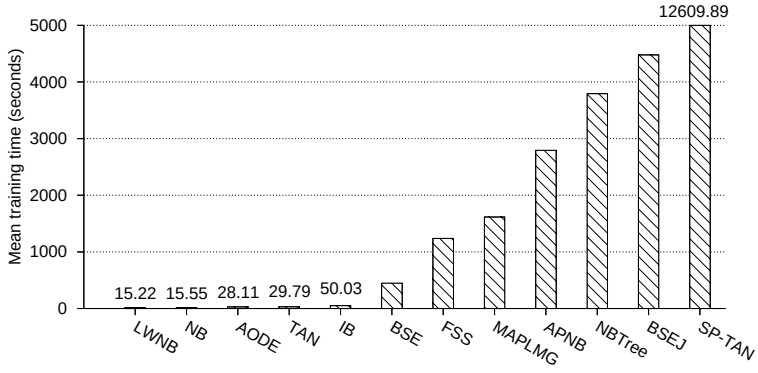


Figure 3.10: Mean training time of NB and 11 semi-naive Bayesian algorithms (exclude LBR) across 60 data sets.

and Pen Digits), NBTree has the highest training time on all these data sets except Connect-4 Opening and Letter Recognition. On Connect-4 Opening (42 attributes), the training time of SP-TAN, BSEJ and NBTree are 551821.75, 195316.21 and 17939.40 seconds respectively. The mean training time of SP-TAN and BSEJ are lower than that of NBTree when Connect-4 Opening is excluded. APNB has high training time on data sets with large number of instances and classes. On Letter Recognition, which has 20000 instances and 26 classes, the training time of APNB (49190.25 seconds) is higher than that of NBTree (40406.30 seconds).

APNB has lower training time on 45 data sets compared to FSS and 52 data sets compared to MAPLMG. Due to its high training time on data sets with a large number of instances and classes, such as Letter Recognition and Nettalk (52 classes), it has higher mean training time than FSS and MAPLMG.

Because FSS uses forward selection and continues the selection process when there is no accuracy degradation, it is substantially slower than BSE. It has higher training time than BSE on all the 60 data sets.

Classification Time

FSS, BSE, NB and IB have a clear and consistent advantage over the remaining algorithms. LWNB, MAPLMG, AODE and NBTree have a significant disadvantage relative to the rest of algorithms. LWNB, which is a lazy method, has a significant higher test time rank than all the other algorithms except MAPLMG. However, it has much higher classification time compared to MAPLMG on 58 data sets and lower

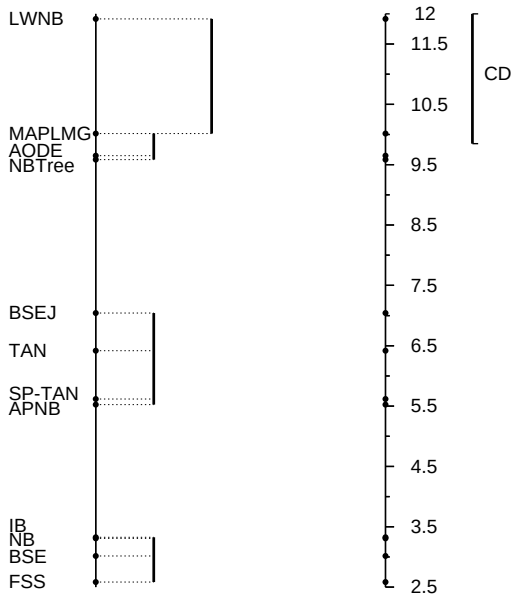


Figure 3.11: Classification time comparison of NB and 11 semi-naive Bayesian algorithms (exclude LBR) with the Nemenyi test on 60 data sets. CD = 2.1519.

classification time on Audiology and Lung Cancer, both of them have more than 55 attributes.

Since FSS and BSE generally delete attributes, they have lower mean classification time rank compared to NB. Furthermore, as FSS employs forward selection and hence usually uses a small number of attributes to classify an instance, it achieves the lowest mean classification time rank.

The Nemenyi test does not assess the differences between APNB, SP-TAN, TAN, and BSEJ as statistically significant. At classification time, APNB needs to apply linear adjustments to the class probabilities, SP-TAN and TAN require to apply parent function to each attribute and BSEJ needs to scan the joining and deleting tables. Due to the additional operations, they usually have higher classification time compared to NB, IB, BSE and FSS. Although these algorithms have a significantly higher mean classification time rank than NB, in practice, these time differences are small. For example, the time differences between APNB and NB on 53 data sets are less than 1 second.

AODE and MAPLMG have relatively high classification time because they need to average all qualified 1-dependence classifiers. Before using local NB in a leaf to

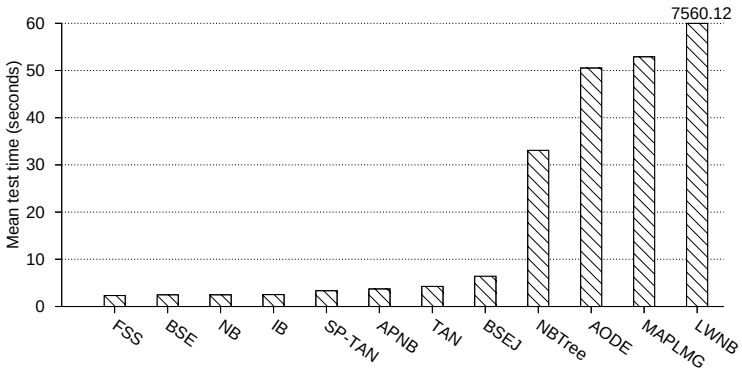


Figure 3.12: Mean classification time of NB and 11 semi-naive Bayesian algorithms (exclude LBR) across 60 data sets.

classify a test instance, NBTree needs extra time to reach the leaf and access the appropriate local NB according to the attribute values of the test instance. It shares a similar level of classification time with AODE and MAPLMG.

3.3.3.2 Bias and Variance: in Relation to the Size of Data Sets

Bias, as discussed in Section 2.2.1.2, measures how closely the learner is able to describe the decision surfaces for a domain, and variance measures the sensitivity of the learner to variations in the training sample. Generally, algorithms that learn highly parameterized models, such as NBTree, have lower bias than algorithms that form models with few parameters, such as NB, because models generated by the former usually fit training data closer than those by the latter, but higher variance in that the former models are more dependent on training data and sensitive to data variations than the latter models. In this section, we discuss how data set size interacts with bias and variance, which in turn affects error.

It is quite likely that differences between small samples are greater than those between large samples. In other words, differences between samples are expected to decrease with increasing sample size. It follows that differences between models formed from those samples are expected to decrease and hence variance is expected to decrease [Brain and Webb, 2002]. Geurts (2002) reported that the behaviors of bias with respect to the sample size is algorithm dependent. In his study, the bias of linear regression is independent of sample size, while decision trees decrease in bias with increasing sample size. Brain and Webb (2002) observed that the bias of C4.5

and its variants tends to decrease, while that of NB increases on all data sets tested except Waveform.

If variance decreases as training set size increases, the bias proportion of error may be higher on large data sets than on small data sets and the variance proportion of error may be higher on small data sets than on large data sets. In consequence, low bias algorithms may have advantage in error on large data sets and low variance algorithms on small data sets [Brain and Webb, 2002; Castillo and Gama, 2006].

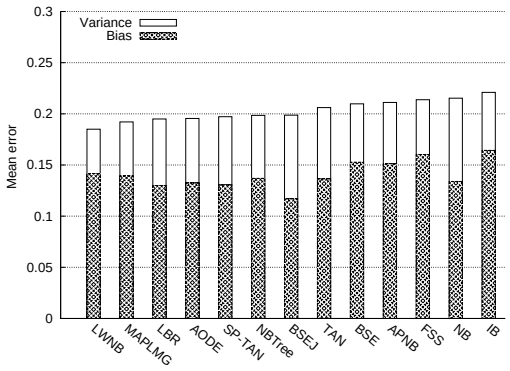
To assess whether the bias proportion of error is higher on large data sets than small data sets, we compare the results of bias as a proportion of error on the twenty-five largest data sets, each with 1000 or more cases, and on the thirty-five smallest data sets, each with less than 1000 cases. Spearman's rank test is used to test whether the difference between the results on large data sets and small data sets over the 13 algorithms is non-random. We assess a difference as significant if $r_s = 1 - 6\sum d_i^2/a(a^2-1) > 0.566$, where d_i is the difference between the rank of mean value on the two sets of data sets on the ith out of 13 algorithms, a is 13, and 0.566 is the critical value for a two-tailed test at a significance level of 0.05.

Table 3.5 provides the ranked mean bias proportion of error on the 25 largest data sets and the 35 remaining data sets over 13 algorithms. Figure 3.13 presents the mean

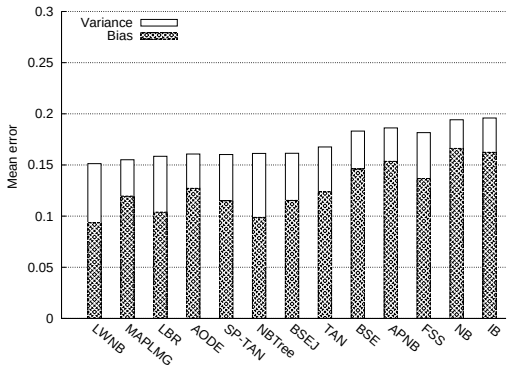
Table 3.5: Mean bias proportion of error on 25 largest and 35 smaller data sets

Bias/Error	NB	BSE	FSS	BSEJ	TAN	SP-TAN	NBTree	LBR	AODE	MAPLMG	LWNB	APNB	IB
Large data	0.856	0.798	0.753	0.715	0.739	0.718	0.612	0.655	0.791	0.771	0.619	0.824	0.829
Rank	13	10	7	4	6	5	1	3	9	8	2	11	12
Small data	0.701	0.673	0.605	0.651	0.608	0.670	0.572	0.667	0.683	0.678	0.624	0.696	0.683
Rank	13	8	2	5	3	7	1	6	11	9	4	12	10

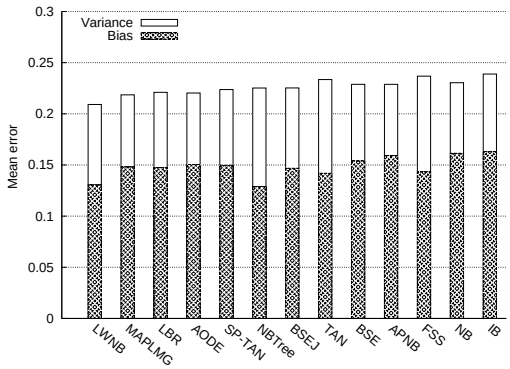
bias proportion on all, the large and the small data sets respectively. From Table 3.5 and subgraphs (b) and (c) of Figure 3.13 we can see that each mean bias proportion on the large data sets is higher than that on small data sets with the exception of LBR and LWNB. This is because the bias proportion of error of LBR on Mushroom (0.0002) and Pen digits (0.0008) is very small and that of LWNB on Mushroom is zero, and hence the mean bias proportion on the large data sets is relatively low for



(a) *Mean error on 60 data sets*



(b) *Mean error on 25 largest data sets*



(c) *Mean error on 35 smallest data sets*

Figure 3.13: Bias accounts for a larger proportion of error on large data sets. (The algorithms are sorted in ascending order on the mean error on 60 data sets)

LBR and LWNB. The r_s value for the Spearman's rank test is $0.819 > 0.566$, which suggests that bias accounts for a larger proportion of error on large data sets.

Figure 3.14 presents the error comparison of the 13 algorithms against each other with the Nemenyi test on the 25 largest data sets (CD = 3.7246). When FSS is compared to NB, APNB and BSE, TAN compared to APNB and IB, BSE compared to APNB and NBTree compared to IB on all 60 data sets, the former of each pair has a lower mean bias rank, and a higher mean error rank compared to the latter. However, the former of each pair achieves a lower mean error rank compared to the latter on 25 largest data sets.

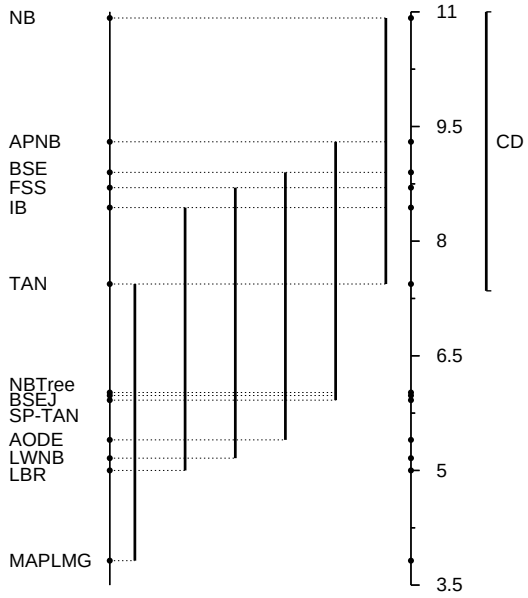


Figure 3.14: Error comparison of NB and 12 Semi-naive Bayesian algorithms with the Nemenyi test on the 25 largest data sets. CD = 3.6493.

Note that the bias advantages of NBTree and BSEJ relative to NB are significant and the error advantages are not on all 60 data sets. However, they have a significant error advantage over NB on the 25 largest data sets, despite the power of the latter test being substantially lower due to its use of fewer data points. Similarly, the error difference between LWNB and BSE is not significant on all 60 data sets, but it is on the 25 largest data sets. FSS has the seventh lowest mean bias rank, second highest mean variance rank and highest mean error rank on 60 data sets, but its mean error rank decreases substantially (moving down four positions) when the 25 largest data sets are compared. These results correspond well to the expectation that algorithms with a low bias profile, such as NBTree, tend to have lower relative error on large training sets and algorithms with a low variance profile, such as NB, tend to have lower relative error on small training sets.

In an earlier study [Zheng and Webb, 2005] which compared NB, AODE, NBTree, LBR, TAN, SP-TAN, BSEJ, BSE and FSS, AODE was the only algorithm to have a significant advantage in error over NB. However, three semi-naive Bayesian algorithms, LBR, AODE and SP-TAN, achieve a significant advantage in error over NB in the current study. AODE and SP-TAN achieve statistically significant win-draw-loss

error records over NBTree in the previous study, while their error is not significantly lower than that of NBTree in this study. The advantage of AODE over BSEJ is significant in the earlier study, it is not in the current study. Although different comparison methods (the binomial sign test for the earlier study and the Nemenyi test for this study) are employed, the main reason for the different outcomes might be that the previous study used Kohavi and Wolpert's bias-variance estimation technique, which produces smaller training sets than the technique employed by the current study, and compared the algorithms on 35 data sets with a smaller average size. Hence, the experiments might put algorithms with a low bias profile at a disadvantage.

3.3.3.3 Comparison of NB, Five Semi-naive Bayesian Algorithms, Logistic Regression and SVM on 58 Data Sets

Because using a large number of algorithms reduces the power of the Friedman and Nemenyi tests, in this section we only compare NB and five semi-naive Bayesian algorithms to logistic regression and LibSVM. These five algorithms are MAPLMG, LBR, AODE, LWNB and SP-TAN, all of which significantly reduce NB's error. As we mentioned previously, at the time of writing, we have not obtained the results of LibSVM on Adult and Connect-4 Opening due to the high time complexity of "grid-search" on C and γ for the RBF kernel. We compare the 8 algorithms on 58 data sets. Since LibSVM does not provide probability estimates, we only compare these algorithms with respect to error, bias and variance.

With 8 algorithms and 58 data sets, the Friedman statistic is distributed according to the F distribution with $a-1 = 8-1 = 7$ and $(a-1)(D-1) = (8-1)*(58-1) = 399$ degrees of freedom. The critical value of $F(7, 399)$ for $\alpha = 0.05$ is 2.0325. The null-hypotheses are rejected as the Friedman statistic for error, bias and variance are 5.8459, 16.0903 and 5.9328 respectively. The Critical Difference using the Nemenyi test for $\alpha = 0.05$ is $\text{CD} = 3.031 * \sqrt{a(a+1)/(6*D)} = 3.031 * \sqrt{8(8+1)/(6*58)} = 1.3787$.

Figures 3.15, 3.16, and 3.17 show the comparison of the 8 algorithms against each other with the Nemenyi test on error, bias and variance respectively. Logistic regression is simplified as logistic and LibSVM as SVM in each graph.

Error

LibSVM and MAPLMG achieve the lowest and second lowest mean error ranks (3.5690 and 3.6293) and outperform logistic regression and NB. The differences between LibSVM, MAPLMG, AODE, LWNB, LBR and SP-TAN are not statistically significant. Logistic regression has the second highest mean error rank (5.1293). Its

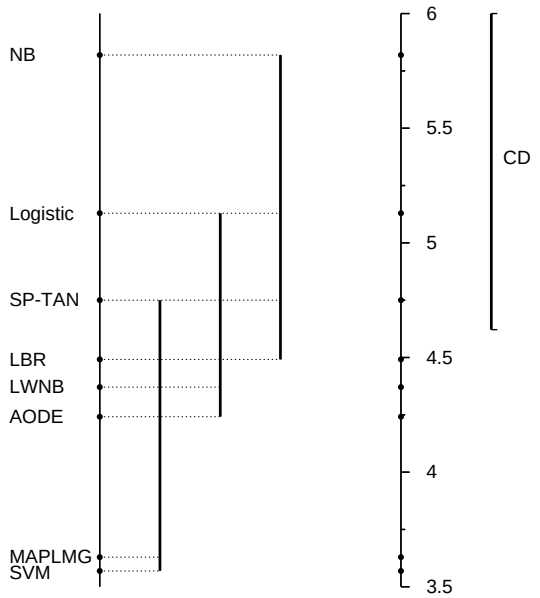


Figure 3.15: Error comparison of NB, 5 Semi-naive Bayesian algorithms, logistic regression and SVM with the Nemenyi test on 58 data sets (exclude Adult and Connect-4 Opening). CD = 1.3787.

error is not significantly different to that of AODE, LWNB, LBR, SP-TAN, and NB.

Note that LBR and SP-TAN have a significant error advantage relative to NB in the previous comparison (Figure 3.5). However, they do not have a significant advantage relative to NB in this comparison from which Adult and Connect-4 Opening are excluded. This is because LBR and SP-TAN have considerably lower error ranks than NB on the large data sets Adult and Connect-4 Opening that can not included in the current experiments. For example, the error rank of LBR on Adult is 1 while that of NB is 13. The addition of the two data sets enhance the power of the Nemenyi test to differentiate LBR and SP-TAN from NB.

Logistic regression also has substantially lower error than NB on the two data sets, but the difference between logistic regression and NB is not statistically significant on

the Nemenyi test when NB, SP-TAN, LBR, AODE, MAPLMG, LWNB and logistic regression are compared on 60 data sets. For conciseness, we do not report the results. In fact, when logistic regression is compared solely to NB, it has lower error on 34 data sets and higher on 26 data sets. This difference is also not statistically significant using a binomial sign test, $p = 0.1831$.

Bias

LWNB has the lowest mean bias rank (3.0603). It shares a similar level of bias

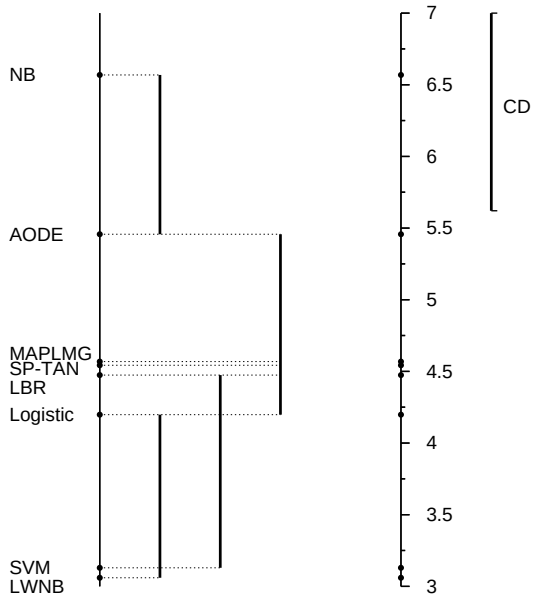


Figure 3.16: Bias comparison of NB, 5 Semi-naive Bayesian algorithms, logistic regression and SVM with the Nemenyi test on 58 data sets (exclude Adult and Connect-4 Opening). CD = 1.3787.

with LibSVM and logistic regression, and outperforms all the remaining algorithms. LibSVM has the second lowest mean bias rank (3.1293), which is significantly lower than the mean ranks of SP-TAN, MAPLMG, AODE and NB. Logistic regression has lower mean rank than LBR, SP-TAN, MAPLMG, AODE and NB, but it only has a significant advantage relative to NB.

The bias difference between AODE and NB is not significant in this comparison, but it is in the previous comparison (Figure 3.6). The reason is that the inclusion of Adult and Connect-4 Opening, on which AODE has lower mean bias ranks than NB,

makes the Nemenyi test powerful enough to detect a significant difference between AODE and NB.

When LWNB is compared to LBR and SP-TAN, it has a significant bias advantage in this comparison, but shares a similar level of bias with LBR and SP-TAN in the previous comparison (Figure 3.6). This is because the large number of algorithms (13) included in the previous comparison reduced the power of the Nemenyi test to differentiate the two methods.

Variance

NB and AODE have the lowest and second lowest mean variance rank (3.5517 and 3.5862). They have a significant advantage in variance over SP-TAN, LBR, and LWNB. MAPLMG outperforms LWNB. It has lower mean rank than LibSVM, logistic

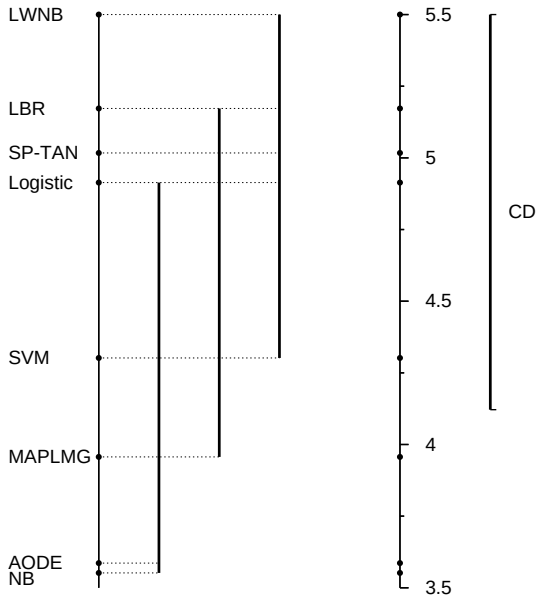


Figure 3.17: Variance comparison of NB, 5 Semi-naive Bayesian algorithms, logistic regression and SVM with the Nemenyi test on 58 data sets (exclude Adult and Connect-4 Opening). CD = 1.3787.

regression, SP-TAN and LBR, but the differences are not statistically significant.

In Figure 3.7, MAPLMG outperforms LBR. However, the current comparison does not reveal this difference. This is also because the strength of the Nemenyi test is

augmented by the addition of Adult and Connect-4 Opening, on which MAPLMG has lower variance than LBR.

3.3.4 Discussion

NB does not perform model selection, using a fixed formula to classify test instances. This results in relatively low variance. The relaxation of the attribute independence assumption may make semi-naive Bayesian methods fit the training sample closer. Consequently, they may have lower bias, but higher variance compared with NB. If a semi-naive Bayesian method can find the right balance between bias and variance, it may deliver higher classification accuracy than NB.

NBTree and LBR

NBTree, which combines decision trees with NB, is a relatively highly parameterized method. It has the lowest mean bias rank in the comparison of the naive and semi-naive Bayesian algorithms (Figure 3.6). LBR establishes a rule in a lazy manner. This rule can be seen as a branch of the tree produced by NBTree. Thus, both approaches can produce models with as high a level of dependence as there are attributes. Because LBR uses lazy learning and hence can adjust its model to each test case, it is expected to have lower bias compared with NBTree. However, Figure 3.6 shows that NBTree has considerably lower bias than LBR.

There are two main differences between these two methods. First, NBTree uses 5-fold cross validation accuracy estimation as the splitting criterion, while LBR uses Leave-One-Out cross validation accuracy estimation. Another difference between NBTree and LBR is the stopping criterion. The former stops the growth of the tree when the relative error reduction is less than 5% or the number of the instances in a splitting node is less than 30, and the latter stops the growth of the rule when there is no significant accuracy improvement as assessed by a sign test. It is quite likely that the sign test at a significance level of 0.05 is stricter than the criterion that the relative error reduction is greater than 5%. Hence, the latter might result in closer fitting compared to the former. For instance, if the best error so far is 4 and the current error is 0, the relative error reduction is $100\% > 5\%$, but this improvement fails the significance test.

To assess this expectation we compare LBR with original NBTree (indicated as NBTree$_{ns}^{5fold}$) and three variants of NBTree using the Nemenyi test. The first variant, called NBTree$_{ns}^{LOO}$ uses the same stopping criterion as original NBTree and Leave-One-Out cross validation as the evaluation function. The second variant, called NBTree$_{s}^{5fold}$, uses the same splitting criterion as original NBTree and a sign test as the stopping criterion. NBTree$_{s}^{LOO}$ uses Leave-One-Out cross validation accuracy estimate as the splitting criterion and a sign test as the stopping criterion. Figure 3.18 presents the bias comparison of the five algorithms using the Nemenyi test (CD = 0.7875).

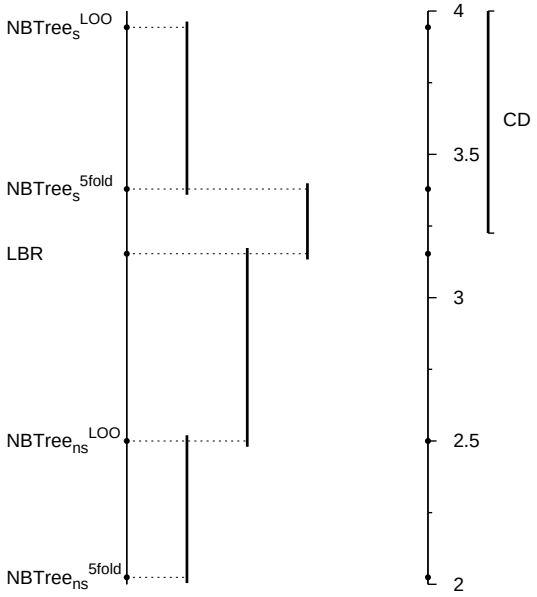


Figure 3.18: Bias comparison of NBTree, NBTree's variants and LBR with the Nemenyi test on 60 data sets. CD = 0.7875

Both NBTree$_{ns}^{5fold}$ and NBTree$_{ns}^{LOO}$ enjoy lower mean bias rank compared with LBR. [5] This result suggests that the splitting criterion is not the cause of the discrepancy from our expectation that LBR has lower bias compared with NBTree. In contrast, NBTree$_{s}^{5fold}$ and NBTree$_{s}^{LOO}$ have higher mean bias rank compared with LBR. Furthermore, NBTree$_{s}^{LOO}$, which uses the same splitting and stopping criteria

[5] Figure 3.6 does not reveal the bias difference between NBTree and LBR is statistically significant. However, the advantage of NBTree in bias is significant compared to LBR, as shown in Figure 3.18. This is because 60 data sets are not sufficient to differ NBTree and LBR when 13 algorithms are compared, while they are sufficient when 5 algorithms are compared by using the Nemenyi test.

as LBR, has a significant disadvantage relative to LBR. This result is consistent with the expectation that the relative error reduction constraint results in closer fitting than the sign test.

Since NBTree and LBR are z-dependence classifiers, they may have great potential to have lower bias on large data sets as these may have sufficient data to obtain accurate higher order probability estimates and hence to have appropriate model selection. As a consequence, these two algorithms may have an advantage in error on large data sets, however, they may not scale up well due to the high training time complexity of NBTree and high classification time complexity of LBR.

LWNB and LBR

Both LWNB and LBR can be considered as techniques that identify a relevant subset of the training set and learn a local NB therefrom. Since they use lazy learning, which may find a model that is most appropriate to the test instance, they may have relatively low bias at the cost of considerably increased classification time. LWNB has the second lowest mean bias rank and considerably lower mean bias rank than LBR (Figure 3.6). Its mean error rank is higher, but not significantly higher, than that of LBR. However, LWNB has substantially lower classification time compared to LBR. This makes LWNB a more appealing option for large data sets relative to LBR.

FSS, BSE and BSEJ

FSS is the only method that has a higher mean error rank compared to NB. It has higher variance than the remaining algorithms except NBTree. The reason might be that FSS employs forward selection, which appears to produce an attribute subset with a small number of attributes. This attribute subset tends to change greatly from sample to sample. In contrast, BSE uses backward selection and usually uses most of the attributes to classify instances. Consequently, there is often less variation between the models created by BSE than there is between those created by FSS. In addition, FSS might be susceptible to getting trapped into poor selections by local minima.

BSEJ provides substantial decrease in bias by creating new compound attributes and deleting attributes. The mean bias rank of BSEJ is slightly lower than that of

LBR. It may also have the potential to deliver low error on large data sets. However, due to the high space complexity, its Weka implementation suffers from memory shortages when large numbers of attributes are joined.

TAN and SP-TAN

Comparing SP-TAN to TAN, the former has much lower variance than the latter. One factor that may contribute to SP-TAN's lower variance is that SP-TAN stops adding arcs when there is no accuracy improvement and hence usually produces a forest, while TAN tends to establish a tree with $n-1$ arcs. SP-TAN exhibits higher accuracy compared to TAN in our data collection. Nonetheless, TAN might be superior when training time is concerned.

APNB and IB

These two methods reduce NB's bias by modifying the probability outcome of NB. Compared to other semi-naive Bayesian methods, bias reductions obtained by APNB and IB are small. For each training instance, APNB increases the probability estimation for the true class if this increase improves the leave-one-out cross validation accuracy, while IB increases the probability estimation for each attribute given the true class. These probability adjustments reduce the zero-one loss, meanwhile, increase the quadratic loss. The training cost of IB, which is linear in the number of instances, attributes and the class, is considerably lower than that of APNB, especially when the numbers of instances and classes are large.

AODE and MAPLMG

AODE reduces variance by aggregating all qualified 1-dependence classifiers. In our comparison, it shares similar levels of error and RMSE with LibSVM, MAPLMG, LBR, LWNB, SP-TAN and logistic regression, while having considerably lower training time relative to LibSVM, MAPLMG, SP-TAN and logistic regression and classification time relative to LWNB in most cases and LBR. AODE maintains the robustness and much of the efficiency of NB, and at the same time exhibits significantly higher classification accuracy and probabilistic prediction for many data sets. Therefore, it

has the potential to be a valuable substitute for NB over a considerable range of classification tasks and has received substantial attention. Indeed, at the time of writing, the paper introducing AODE [Webb et al., 2005] is the most cited paper from 2005 in the Machine Learning journal [Thomson ISI, 2008].

MAPLMG substantially reduces the bias of the base learner, AODE, at the cost of considerable increase in variance. For the data sets in the current study, the reduction in bias outweighs the increase in variance and results in an overall reduction in error. Due to the relative bias/variance profile, MAPLMG is likely to achieve lower error than AODE on large data sets. It delivers the lowest mean error and RMSE ranks of the 12 Semi-naive Bayesian methods and has slightly higher mean error rank than LibSVM, which has substantially higher training time overheads than MAPLMG. The classification time overheads of AODE and MAPLMG are, although higher than other semi-naive Bayesian methods except LWNB and LBR, modest for many classification tasks with moderate or small numbers of attributes.

NB and Logistic Regression

Logistic regression is the discriminative counterpart of NB. Our experimental results reveal that logistic regression has a significant bias advantage and variance disadvantage relative to NB. As discussed in Section 3.3.3.2, low bias algorithms appear to have an advantage in error with larger training sets, while low variance algorithms appear to have an advantage with small training sets. It follows that NB may deliver lower error than logistic regression at small data set sizes and logistic regression may achieve lower error than NB at larger data set sizes. This is consistent with Ng and Jordan's finding that NB has higher asymptotic error than logistic regression, but it can approach its asymptotic error much faster than logistic regression [Ng and Jordan, 2001].

At training time, logistic regression is considerably slower than NB. The mean training time on 60 data sets for logistic regression is 125217.82 seconds, while that for NB is 15.55 seconds. Logistic regression has higher training time than NB on every data set of our collection.

Selection Between Semi-naive Bayesian Methods

As discussed previously, bias appears to dominate error when more data is available. When predictive error is of major concern, algorithms with low bias may prove beneficial on large data sets. However, such algorithms usually have very high time complexity, and may not be attractive when efficiency is an important issue. Generally, for large training data we recommend use of the lowest bias semi-naive Bayesian method whose computational 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 data, an appropriate trade-off between bias and variance should be sought within the prevailing computational complexity constraints.

For extremely small data NB may prove best and for large data NBTree, LWNB, BSEJ and LBR may have an advantage if their computational profiles are appropriate to the task. AODE achieves very low variance and RMSE, intermediate bias, low training time complexity and modest classification time complexity. Its low variance makes AODE an attractive option for small data sets, meanwhile, its low training time complexity makes it a desirable option for large data sets. In consequence, it may prove competitive over a considerable range of classification tasks. MAPLMG further enhances AODE by reducing bias and error and improving probability prediction at the cost of substantially increased training overheads. Hence, it may excel for many classification problems if its computational cost can fall within the computational constraints of the given application.

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.

3.4 Summary

This chapter examines NB and techniques for relaxing the attribute independence assumption. It describes twelve representative semi-naive Bayesian algorithms and provides details of their time and space complexity. NBTree, SP-TAN, BSEJ and

APNB have relatively high training time complexity, while LBR and LWNB have high classification time complexity. BSEJ has very high space complexity.

In the empirical part of this chapter, we compare the algorithms using the bias-variance decomposition and RMSE on sixty natural domains from the UCI Machine Learning Repository. To provide a baseline for comparison, we also present results for logistic regression and LibSVM with parameter search. The study reveals the outstanding performance of MAPLMG on our data collection. It achieves a considerable advantage in error and probabilistic estimation over the other semi-naive Bayesian methods. AODE also demonstrates high accuracy and probabilistic prediction performance. NBTree provides substantial decrease in bias, and significantly wins against the remaining algorithms other than LWNB, BSEJ and LBR. Due to their low bias, NBTree, LWNB, BSEJ and LBR have great potential to reduce error on large data. As variance tends to decrease and bias tends to be a larger portion of error when training set size increases, we suggest using low bias methods for large data sets and low variance methods for small data sets to obtain higher classification accuracy, 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.

Chapter 4

Parent and Child Selection for AODE

The extensive comparative study in the previous chapter showed that AODE is a powerful alternative to NB, substantially improving NB's classification and probability estimation accuracy, while retaining much of its attractive simplicity and efficiency. In theory, AODE would appear to be a promising candidate for attribute selection, which has two potential beneficial effects, both improving accuracy and also reducing classification time due to the need to process fewer attributes. Its error and classification time overheads might be further reduced if harmful SPODEs were excluded.

When applied to NB, BSE has proved to be advantageous in domains with highly correlated attributes. However, its straightforward application to AODE has previously proved unfruitful [Zheng and Webb, 2006; Yang, Webb, Cerquides, Korb, Boughton and Ting, 2006]. This chapter investigates why previous approaches to attribute selection for AODE have proved ineffective, and develops novel attribute selection algorithms that do prove effective when applied to AODE.

4.1 Attribute Selection Is Effective on NB

NB treats attributes as independent and equally important. When two attributes are related, NB may place too much weight on the influence from the two attributes, and too little on the other attributes, which can result in classification bias. Deleting one of these attributes may have the effect of alleviating the problem. The results

of the Nemenyi test in Section 3.3.3 do not reveal a statistically significant difference in error between BSE and NB. This is because 60 data sets are not sufficient to differ BSE from NB when 15 algorithms are compared. To evaluate the effect of attribute selection on NB, we replicate previous experiment results and perform a win/draw/loss comparison.

Table 4.1 presents the win/draw/loss records for BSE against NB on sixty data

Table 4.1: Win/Draw/Loss: BSE vs. NB

	BSE vs. NB	
	W/D/L	p
Error	36/4/20	**0.0220**
Bias	52/3/5	**<0.0001**
Variance	11/4/45	**<0.0001**
RMSE	34/6/20	**0.0380**

Table 4.2: Win/Draw/Loss: FSS vs. NB

	FSS vs. NB	
	W/D/L	p
Error	29/3/28	0.5000
Bias	50/3/7	**<0.0001**
Variance	9/2/49	**<0.0001**
RMSE	33/2/25	0.1791

sets. Each win/draw/loss record is the number of data sets for which BSE obtains lower, the same and higher value on a corresponding metric than NB. The p value is the outcome of a one-tailed binomial sign test. We assess a difference as significant if $p < 0.05$. The win/draw/loss records for FSS against NB is presented in Table 4.2.

The error, bias and RMSE advantages and variance disadvantage of BSE relative to NB are evident. FSS significantly reduces the bias and increases the variance of NB. The increase in variance outweighs the reduction in bias and results in an overall increase in error. It is observed that FSS considerably improves NB's accuracy in domains with highly related attributes. For example, FSS reduces the error of NB from 0.1276 to 0.0552 on King-rook-vs-king-pawn.

4.2 Attribute Selection for AODE

In theory, attribute selection should also have positive effect on AODE. While an individual SPODE can factor out harmful attribute inter-dependencies in which the parent is involved, it will not help when the parent is not. When there are many more attributes than those that participate in a particular interdependency, the majority of SPODEs will not factor out the interdependency, and hence it is credible that deleting

one of the attributes should be beneficial. Why then have previous attempts [Zheng and Webb, 2006; Yang et al., 2006] to apply attribute selection to AODE proved unfruitful? There are two main differences between applying attribute selection in NB and AODE:

Different complexity and variance. An AODE model has greater complexity compared to an NB model, resulting in greater variance in estimates of performance as the model is manipulated through attribute elimination and hence reduced reliability in these estimates. The lower reliability of these estimates may result in poor choices when they are used to select attributes for elimination. To explore this issue, we evaluate the use of a statistical test to assess whether an observed difference in holdout evaluation scores should be accepted as meaningful during the attribute selection process.

Different roles. Attributes play multiple roles in an AODE model (either a parent or a child) whereas they play only a single role of child in an NB model. To explore this issue, we investigate the separate selection of attributes in each of the parent and child roles, as well as in both roles together.

4.2.1 Statistical Test

It is quite likely that small improvements in leave-one-out accuracy may be attributable to chance. In consequence it may be advantageous to use a statistical test to assess whether an improvement is significant. A standard binomial sign test is employed. Treating the examples for which an attribute addition or deletion corrects a misclassification as a *win* and one for which it misclassifies a previously correct example as a *loss*, a change is accepted if the number of wins exceeds the number of losses and the probability of obtaining the observed number of wins and losses if they were equiprobable was no more than 0.05.

Table 4.3 illustrates a hypothetical case in which the improvement is accepted as significant using the binomial sign test at level of 0.05. In this example, 11 previously misclassified examples are corrected, while 3 previously correct examples are misclassified by the attribute selection under consideration. The outcome of a one-tailed binomial sign test is $0.0287 < 0.05$. Therefore, we accept this change as significant.

Table 4.3: A change (11 wins and 3 losses) passes the binomial sign test ($p = 0.0287 < 0.05$)

Before attribute selection	After attribute selection	Number of examples
×	√	11
×	×	9
√	×	3
√	√	7

×: examples that are misclassified
√: examples that are classified correctly

4.2.2 Parent and Child Roles in an AODE Model

In NB, all attributes play only a single role of child (Figure 4.1). Attribute selection

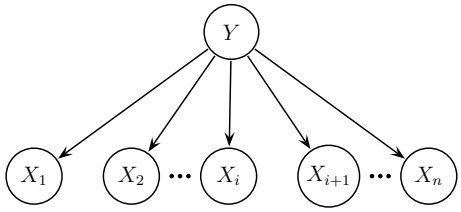
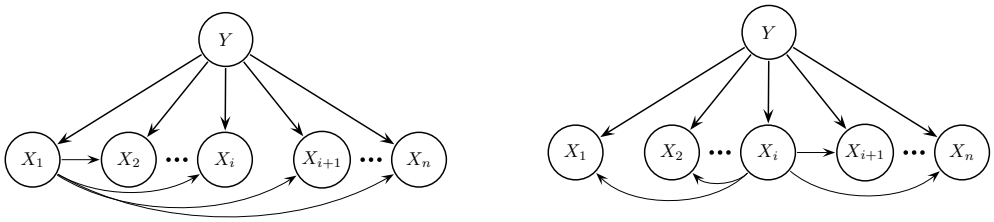


Figure 4.1: Single role: all attributes in an NB model are children

for NB is straightforward: removing or adding a complete attribute from the attribute set. However, as shown in Figure 4.2, one attribute is both a parent in one SPODE model and a child in another SPODE model. Since AODE averages the predictions of all SPODEs, an attribute can be either a parent or a child in an AODE model. It is possible that deletion might have quite different impact in the context of each of these roles.

To formalize the various attribute selection strategies we introduce into AODE the use of a *parent* (p) and a *child* (c) set, each of which contains the set of indices of attributes that can be employed in respectively a parent or child role in the AODE. The number of indices in each set is denoted respectively as $|\,p\,|$ and $|\,c\,|$. We define



(a) X_1 is the SuperParent (b) X_i is the SuperParent

Figure 4.2: Multiple roles: attributes can be either a parent or a child in an AODE model

$\text{AODE}_{p,c}$ as

$$\underset{y}{\text{argmax}} \left(\sum_{i \in p : F(x_i) \geq m} \hat{P}(y, x_i) \prod_{j \in c} \hat{P}(x_j \mid y, x_i) \right).$$

4.3 Child Selection Might Have Greater Effect than Parent Selection

In $\text{AODE}_{p,c}$, a linear function is used to combine constituent SPODEs, and a multiplicative function is used to combine attributes within each SPODE. Large improvements are possible due to the multiplicative influence, and hence exclusion of a child may have greater effect than exclusion of a parent.

In addition, assuming that attribute x_i is related to other attributes, and that these harmful interdependencies can be detected and repaired by BSE or FSS, the exclusion of x_i from c may have influence on $|\,p\,| - 1$ SPODEs, while the exclusion of x_i from p will only factor out the effect of the single SPODE in which x_i is the parent.

4.4 AODE with BSE and FSS

In the context of AODE, BSE and FSS use Leave-One-Out cross validation error on AODE as a selection criterion. Each available selection is attempted and the one

that results in the lowest error is implemented. The process is repeated for successive attributes until a stopping criterion has been met.

4.4.1 Four Types of Attribute Selection for AODE

There are four different types of attribute selection for AODE:

Parent selection. This approach selects a subset of complete constituent SPODEs. Parent selection with BSE is called *Parent Elimination* (PE) and with FSS *Parent Addition* (PA) [Yang et al., 2006].

Child selection. This approach selects a subset of attributes which are employed in child roles within constituent SPODEs. We call child selection with BSE *Child Elimination* (CE) and with FSS *Child Addition* (CA).

Parent and child selection. This approach selects a subset of attributes for use in any role in the classifier. This strategy with BSE and FSS are called *Parent and Child Elimination* (P∧CE) [Zheng and Webb, 2006] and *Parent and Child Addition* (P∧CA) respectively.

Parent or child selection. This approach performs any one of the other types of attribute selections (parent selection, child selection or parent and child selection) at each step. Parent or child selection with BSE and FSS are denoted as *Parent or Child Elimination* (P∨CE) and *Parent or Child Addition* (P∨CA) respectively.

The four types of attribute addition initialized p or c to the empty set and the accuracy of resulting classifiers, called *initial accuracy*, was set to NB's Leave-One-Out cross validation accuracy [Zheng and Webb, 2007]. Subsequent research shows that this value always prevents adding attributes at initial stage and increases error. Thus, the current research sets the initial accuracy to zero for attribute addition algorithms. For the four types of attribute elimination, the initial accuracy is set to AODE's Leave-One-Out cross validation accuracy. Stop on First Nonimprovement (SFN) and Stop on First Reduction (SFR) (refer to Section 2.5.2.3) are incorporated into attribute elimination and attribute addition respectively as these produce the best performance (For conciseness, results of attribute elimination with SFR and

attribute addition with SFN are not presented). When a statistical test described in Section 4.2.1 is used, we only consider significant improvements.

4.4.1.1 Parent Elimination and Parent Addition

Table 4.4 outlines PE and PA. PE begins with p and c initialized to the full set of $\{1 \ldots n\}$. It deletes attribute indexes from p, effectively deleting a single SPODE at each step. PA starts with p and c initialized to the empty and full sets respectively. It adds attribute indexes to p, effectively adding a single SPODE at each step.

Algorithm: **Parent Elimination**	Algorithm: **Parent Addition**
1. SET p to $\{1 \ldots n\}$	1. SET p to $\emptyset$
2. SET c to $\{1 \ldots n\}$	2. SET c to $\{1 \ldots n\}$
3. Evaluate AODE	3. SET the initial accuracy to zero
4. Consider deleting every $i \in p$ from p and evaluate the current $\text{AODE}_{p,c}$	4. Consider adding every $i \in \{1 \ldots n\} \setminus p$ to p and evaluate the current $\text{AODE}_{p,c}$
5. IF there is an attribute deletion which improves accuracy THEN	5. IF there is an attribute addition which does not reduce accuracy THEN
5.1 Delete the attribute index whose deletion improves accuracy most from p	5.1 Add the attribute index whose addition results in the highest accuracy to p
5.2 Go to step 4	5.2 Go to step 4
6. ELSE	6. ELSE
6.1 Return current $\text{AODE}_{p,c}$	6.1 Return current $\text{AODE}_{p,c}$
7. ENDIF	7. ENDIF

Table 4.4: Parent Elimination (PE) with Stop on First Nonimprovement (SFN) and Parent Addition (PA) with Stop on First Reduction (SFR). When a statistical test is employed, only significant improvements satisfy the condition of step 5.

4.4.1.2 Child Elimination and Child Addition

CE and CA are described in Table 4.5. CE starts with p and c initialized to the full set of $\{1 \dots n\}$. It deletes attribute indexes from c, effectively deleting an attribute from within every SPODE at each step. CA begins with p and c initialized to the full and empty sets respectively. It adds attribute indexes to c, effectively adding an attribute to within every SPODE at each step.

Algorithm: **Child Elimination**	Algorithm: **Child Addition**
1. SET p to $\{1 \dots n\}$	1. SET p to $\{1 \dots n\}$
2. SET c to $\{1 \dots n\}$	2. SET c to $\emptyset$
3. Evaluate AODE	3. SET the initial accuracy to zero
4. Consider deleting every $i \in c$ from c and evaluate the current AODE$_{p,c}$	4. Consider adding every $i \in \{1 \dots n\} \setminus c$ to c and evaluate the current AODE$_{p,c}$
5. IF there is an attribute deletion which improves accuracy THEN	5. IF there is an attribute addition which does not reduce accuracy THEN
5.1 Delete the attribute index which improves accuracy most from c	5.1 Add the attribute index whose addition results in the highest accuracy to c
5.2 Go to step 4	5.2 Go to step 4
6. ELSE	6. ELSE
6.1 Return current AODE$_{p,c}$	6.1 Return current AODE$_{p,c}$
7. ENDIF	7. ENDIF

Table 4.5: Child Elimination (CE) with Stop on First Nonimprovement (SFN) and Child Addition (CA) with Stop on First Reduction (SFR). When a statistical test is employed, only significant improvements satisfy the condition of step 5.

4.4.1.3 Parent and Child Elimination and Parent and Child Addition

Starting with the full set for both p and c, P∧CE [Zheng and Webb, 2006] at each step deletes the same value from both p and c, thus eliminating it from use in any role in the classifier. Beginning with the empty set for both p and c, P∧CA at each step adds the same value to both p and c, hence selecting it for use in any role in the classifier. P∧CE with SFN and P∧CA with SFR are presented in Table 4.6.

Algorithm: **Parent and Child Elimination**	Algorithm: **Parent and Child Addition**
1. SET p to $\{1 \ldots n\}$	1. SET p to $\emptyset$
2. SET c to $\{1 \ldots n\}$	2. SET c to $\emptyset$
3. Evaluate AODE	3. SET the initial accuracy to zero
4. Consider deleting every $i \in p$ from p and c and evaluate the current $\text{AODE}_{p,c}$	4. Consider adding every $i \in \{1 \ldots n\} \setminus p$ to p and c and evaluate the current $\text{AODE}_{p,c}$
5. IF there is an attribute deletion which improves accuracy THEN	5. IF there is an attribute addition which does not reduce accuracy THEN
5.1 Delete the attribute index which improves accuracy most from p and c	5.1 Add the attribute index whose addition results in the highest accuracy to p and c
5.2 Go to step 4	5.2 Go to step 4
6. ELSE	6. ELSE
6.1 Return current $\text{AODE}_{p,c}$	6.1 Return current $\text{AODE}_{p,c}$
7. ENDIF	7. ENDIF

Table 4.6: Parent and Child Elimination (P∧CE) with Stop on First Nonimprovement (SFN) and Parent and Child Addition (P∧CA) with Stop on First Reduction (SFR). When a statistical test is employed, only significant improvements satisfy the condition of step 5.

4.4.1.4 Parent or Child Elimination and Parent or Child Addition

Table 4.7 outlines P∨CE and P∨CA. P∨CE performs any one of the other types of attribute eliminations (PE, CE or P∧CE) in each iteration, selecting the option that best reduces error. P∨CA performs any one of the other types of attribute additions (PA, CA or P∧CA) in each iteration, selecting the option that most improves the accuracy.

Algorithm: **Parent or Child Elimination**	Algorithm: **Parent or Child Addition**
1. SET p to $\{1 \ldots n\}$	1. SET p to $\emptyset$
2. SET c to $\{1 \ldots n\}$	2. SET c to $\emptyset$
3. Evaluate AODE	3. SET the current accuracy to zero
4. Consider deleting every $i \in p$ from p, every $j \in c$ from c and every $k \in p \cap c$ from p and c and evaluate the current $\text{AODE}_{p,c}$	4. Consider adding every $i \in \{1 \ldots n\} \setminus p$ to p, every $j \in \{1 \ldots n\} \setminus c$ to c and every $k \in \{1 \ldots n\} \setminus p \cup c$ to p and c and evaluate the current $\text{AODE}_{p,c}$
5. IF there is an attribute deletion which improves accuracy THEN	5. IF there is an attribute addition which improves accuracy THEN
5.1 Delete the attribute index which improves accuracy most from p or c according to which deletion is performed	5.1 Add the attribute index which improves accuracy most to p or c according to which addition is performed
5.2 Go to step 4	5.2 Go to step 4
6. ELSE	6. ELSE
6.1 Return current $\text{AODE}_{p,c}$	6.1 Return current $\text{AODE}_{p,c}$
7. ENDIF	7. ENDIF

Table 4.7: Parent or Child Elimination (P∨CE) with Stop on First Nonimprovement (SFN) and Parent or Child Addition (P∨CA) with Stop on First Reduction (SFR). When a statistical test is employed, only significant improvements satisfy the condition of step 5.

For ease of understanding, we use a hypothetical example with 4 attributes (X_1, X_2, X_3 and X_4) to explain models established by different attribute selection algorithms.

For PE and PA, $c = \{1, 2, 3, 4\}$. Assuming that PE deletes X_2 and X_4 (or PA adds X_1 and X_3). The resulting p is $\{1, 3\}$. AODE$_{p,c}$ classifies an instance by averaging the predictions of the two SPODEs illustrated in Figure 4.3.

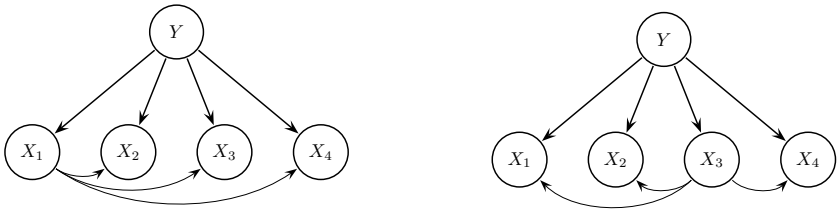


Figure 4.3: The SPODES for $p = \{1, 3\}$ and $c = \{1, 2, 3, 4\}$

For CE and CA, p is the entire set ($p = \{1, 2, 3, 4\}$). If CE deletes X_2 and X_4 (or CA adds X_1 and X_3), then $c = \{1, 3\}$. The SPODEs in the AODE$_{p,c}$ model are presented in Figure 4.4.

Figure 4.4: The SPODES for $p = \{1, 2, 3, 4\}$ and $c = \{1, 3\}$

For P∧CE and P∧CA, p is equivalent to c. If P∧CE deletes X_2 and X_4 (or P∧CA adds X_1 and X_3), $p = \{1, 3\}$ and $c = \{1, 3\}$. Figure 4.5 illustrates the SPODES in the AODE$_{p,c}$ model in this case.

Assuming that P∨CE deletes X_3 and X_4 from p and X_2 from c (or P∨CA adds X_1 and X_2 to p and adds X_1, X_3 and X_4 to c). The resulting p and c are $\{1, 2\}$

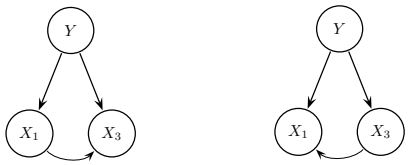


Figure 4.5: The SPODES for $p = \{1, 3\}$ and $c = \{1, 3\}$

and $\{1, 3, 4\}$ respectively. The SPODES in the AODE$_{p,c}$ model are illustrated in Figure 4.6.

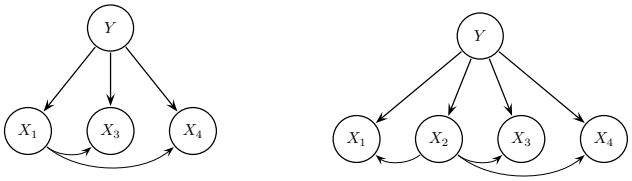


Figure 4.6: The SPODES for $p = \{1, 2\}$ and $c = \{1, 3, 4\}$

4.4.2 Complexity

At training time PA and PE generate a three-dimensional table of probability estimates, as AODE does. They must also store the training data, with additional space complexity $O(tn)$, to perform leave-one-out cross validation on AODE. A three-dimensional table, indexed by instance, class and attribute, is introduced to speed up the process of evaluating the classifiers, with space complexity $O(tkn)$. Therefore, the resulting space complexity is $O(tkn + k(nv)^2)$. Deleting attributes has time complexity of $O(tkn^2)$, as a single leave-one-out cross validation is order $O(tk)$ and it is performed at most $O(n^2)$ times. They have identical time and space complexity to AODE at classification time.

As child selection requires modifying the probability estimates for $|\,p\,|$ SPODEs at each step, it has higher training time complexity than that of parent selection, which only considers one SPODE at each step. The training time complexity of

the strategies involving child selection is $O\big(tkn^3\big)$, as a single leave-one-out cross validation is order $O(tkn)$. They have identical space complexity and classification time complexity to PA and PE.

4.5 Experimental Results

We experimentally evaluate the performance of different types of attribute selection on 60 data sets. The main goal in this comparison is to assess the efficacy of the statistical test and study the influence of the use of different types of attribute selection in AODE. We compare the classification performance of AODE with different attribute selection techniques to AODE using the experimental method described in Section 3.3.2.

Two variants of attribute selection were evaluated, one employing a binomial sign test and the other not. Algorithms using a binomial sign test are superscripted by ^{S}. Since algorithms with a binomial sign test have identical outcomes on many data sets to AODE, 60 data sets are not sufficient to differentiate 17 algorithms by using the Friedman and Nemenyi tests. Therefore, we use pairwise win/draw/loss comparison. The number of times that an algorithm performs better, equally or worse to the others is summarized into pairwise win/loss/draw records which are presented in Tables 4.8, 4.9, 4.10 and 4.11. [1] Algorithms are sorted in descending order on the value of wins minus losses against AODE on each metric. Each entry compares the algorithm with which the row is labelled (L) against the algorithm with which the column is labelled (C). We assess a difference as significant if the outcome of a one-tailed binomial sign test is less than 0.05.

Error

P$\wedge$CE^S, CE^S and P$\vee$CE^S enjoy a significant advantage in error over AODE ($p = 0.0318$, $p = 0.0481$ and $p = 0.0481$ respectively). Attribute addition algorithms (both with and without a statistical test) always have a significant disadvantage to AODE with the exception that PA and P$\wedge$CA share a similar level of error with AODE. No

[1] To avoid breaking the flow of the main text, these tables are presented at the end of this chapter.

significant differences are identified when the rest of algorithms are compared with AODE.

The algorithms using attribute elimination share a similar level of error with the exception that PE and P∨CE outperform CE and P∨CE outperform P∧CE. The advantage of all the attribute elimination algorithms is significant compared with all the attribute addition algorithms but PA and P∧CA . PA has a significant advantage over CA, P∧CA and P∨CA (with and without statistical test). It also outperforms PA^S. The advantage of P∧CA over PA^S, P∧CA^S, CA^S and CA is significant.

The reason the performance of child addition is disappointing might be that it is susceptible to getting trapped into poor selections by local minima during the first several additions. For example, on Letter Recognition, the average number of attributes added (fifty runs) for CA is 1.063. That is, there is accuracy reduction after adding less than 2 attributes (on average) to the c set, which results in the poor error of 0.5071 (the error of AODE is 0.1365). In contrast, the average number of attributes deleted for CE is 0.052 and the error of CE is 0.1362. On 16 data sets, the attribute addition ratio of CA, which is obtained by dividing the number of attributes added by the number of attributes across all iterations, is less than 0.1. On all the 60 data sets, attribute elimination ratio of CE, which is obtained by dividing the number of attributes deleted by the number of attributes across all iterations, is less than 0.5.

Bias

All the attribute elimination algorithms, except PE^S, have a significant advantage in bias over AODE and PE^S. P∧CE , CE and P∨CE outperform PE and the four attribute elimination algorithms with a statistical test.

All the attribute addition algorithms with a statistical test, except PA^S, have a significant disadvantage in bias relative to AODE and PA^S. The three attribute addition algorithms without a statistical test (PA , P∧CA and P∨CA) have a significant bias advantage over AODE and CA .

Variance

AODE enjoys a significant advantage over all the algorithms except the four attribute elimination algorithms with a statistical test (PE^S, CE^S, P∧CE^S and P∨CE^S). These

four algorithms share a similar level of variance and have a significant advantage over all the other attribute selection algorithms.

Note that PA^S has very low bias and high variance. Generally, statistical tests are employed to reduce variance. However, for algorithms employing forward selection, using a statistical test may increase variance in that less attributes are included. In addition, as SFR is incorporated into attribute addition, PA continues adding parents so long as there is no accuracy reduction. Unsurprisingly, PA^S has higher variance than PA. It is observed that large accuracy improvements occur during the first several parent additions. This might be a possible reason PA^S achieves relatively lower bias. Different from parent addition, child addition is highly susceptible to local minima, hence it always has high bias and variance.

RMSE

No attribute selection algorithms have significant advantage in RMSE compared to AODE. All attribute addition algorithms, but P∧CA , have a significant disadvantage relative to AODE. Attribute elimination algorithms with a statistical test have a significant advantage over all the attribute addition algorithms except P∧CA.

The error, bias and variance win/draw/loss results are different from those presented in Zheng and Webb (2007), in which the algorithms are compared using 56 data sets and the accuracy of four types of attribute addition is initialized to NB's accuracy estimated by Leave-One-Out cross validation. These differences do not affect win/draw/loss outcomes.

Information loss function penalizes a classifier extremely if it predicts a very small probability for the actual class. In this thesis, a method that penalizes this case more moderately, RMSE is used to compare the accuracy of probability estimates. Similar probability prediction results to the earlier paper can be obtained if the information loss function is employed.

Continue Search and Select Best

To observe the behaviors of parent and child selection, we also examine the attribute selection techniques with CSSB. Due to the significantly increasing variance, all of these selection approaches have proved ineffective. Figure 4.7 shows the error ratio

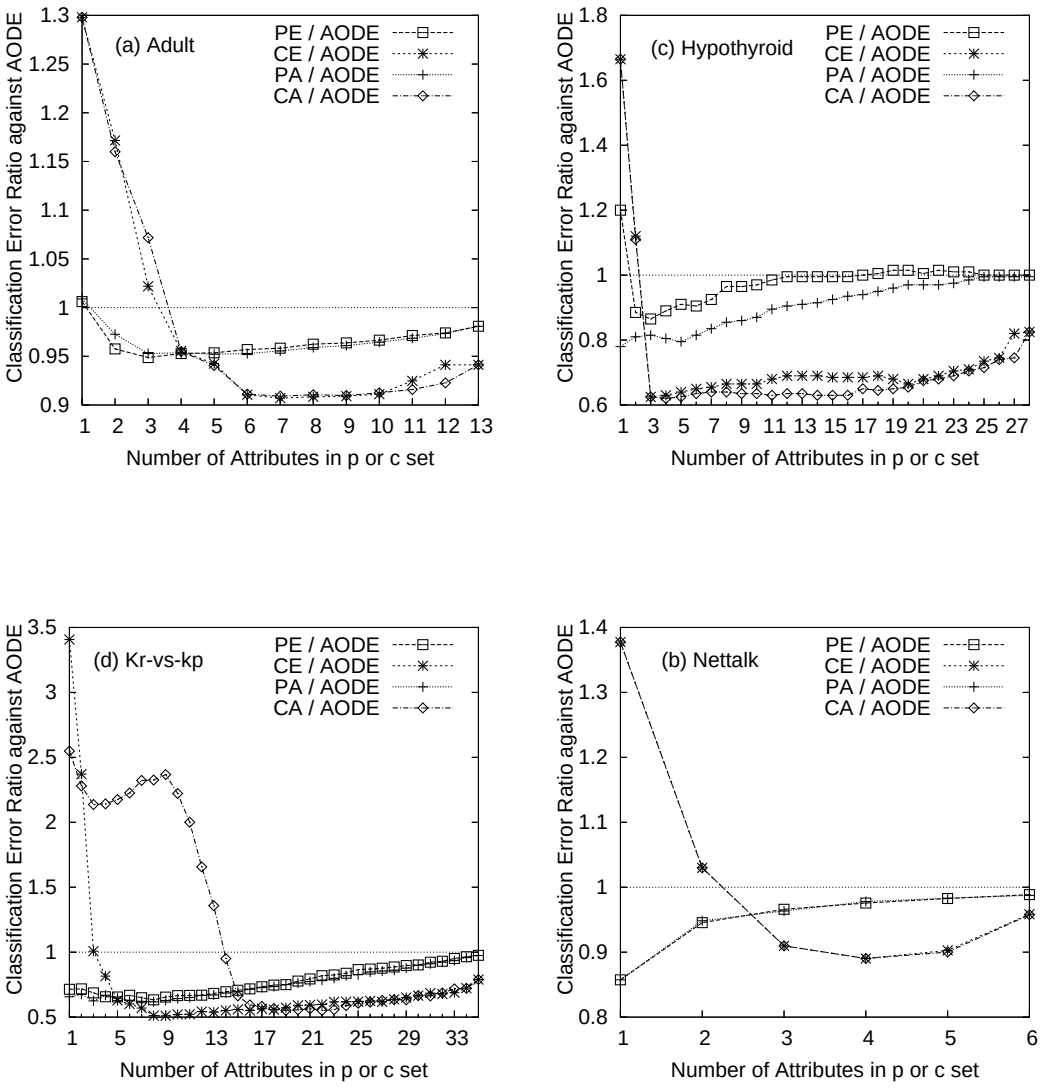


Figure 4.7: Error ratio of parent and child selection using CSSB against AODE, as function of the number of attributes

of PA [Yang et al., 2006], PE [Yang et al., 2006], CA and CE against AODE as a function of the number of attributes on 4 data sets with more than 3000 instances, in which both selection of parent and child have lower error compared with AODE (for conciseness, we have skipped other 6 data sets). The values on the x-axis are the number of attributes in the p set for PA and PE, and the number of attributes in the c set for CA and CE. The values on the y-axis are the classification error of each selection algorithm divided by that for AODE. The smaller the ratio, the more accuracy improvement can be obtained.

Slight error differences between PA and PE are observed as shown in the graph (win/draw/loss being 27/8/25). Notice that PA tends to achieve the minima at an early stage, while PE appears to reach it at a late stage.

CE has greater error reduction compared with PE until there are a small number of children left, after which it increases error sharply. The error ratios for PE and CE for the first attribute elimination are 0.98 and 0.94, 0.99 and 0.96, 1 and 0.83, and 0.98 and 0.79 for Adult, Nettalk, Hypothyroid and King-rook-vs-king-pawn respectively.

The performance of CA fluctuates over the first several attribute additions for King-rook-vs-king-pawn. Similar behavior is observed for many other data sets in our collection. In this circumstance, CA with SFN or SFR is likely to get trapped into local minima.

4.6 Summary

AODE efficiently induces classifiers that have competitive classification performance with other state-of-the-art semi-naive Bayesian algorithms. Its error and classification time overheads might be further reduced if harmful SPODEs are excluded. In view of their effectiveness with NB, it is surprising that previous applications of BSE to AODE have proved ineffective. In this chapter we explore two explanations of this phenomenon. One is that AODE has higher variance compared with NB, and hence appropriate variance management is required. Another is that child selection appears to have greater effect than parent selection, as SPODEs are combined using a linear function but attributes within a SPODE are combined using a multiplicative function.

We describe four types of attribute selection for AODE and provide details of their time and space complexity. Our extensive experiments suggest that the types of

attribute elimination that remove child attributes from within the constituent ODEs can significantly reduce bias and error, but only if a statistical test is employed to provide variance management. In contrast, elimination of complete constituent ODEs does not consistently provide error reduction. The types of attribute addition that add child attributes to within the constituent ODEs do not provide any positive benefits, possibly due to being misled early in the search by local minima. These results suggest that the elimination of a child is more effective than the elimination of a parent, leading to effective approaches to further enhance AODE's accuracy.

Table 4.8: Win/Draw/Loss records of error on 60 data sets for parent and child selection. The algorithms are sorted in descending order on the value of wins minus losses against AODE on error.

	Error															
W/L/D	P∧CE^S	CE^S	P∨CE^S	PE	PE^S	CE	P∨CE	P∧CE	P∧CA	PA	PA^S	P∨CA	P∨CA^S	CA	P∧CA^S	CA^S
P∧CE^S																
CE^S	7/45/8															
P∨CE^S	8/48/4	6/48/6														
PE	27/10/23	27/10/23	27/10/23													
PE^S	6/41/13	6/41/13	6/40/14	20/12/28												
CE	27/6/27	27/6/27	27/6/27	**21/3/36**	27/6/27											
P∨CE	28/3/29	28/3/29	28/3/29	25/6/29	28/4/28	**35/7/18**										
P∧CE	24/6/30	25/4/31	24/4/32	21/5/34	26/5/29	30/7/23	**20/7/33**									
P∧CA	28/1/31	28/1/31	28/1/31	**22/0/38**	28/1/31	27/1/32	**19/1/40**	26/1/33								
PA	26/2/32	26/2/32	26/2/32	**22/2/36**	27/2/31	34/4/22	24/2/34	30/2/28	**39/0/21**							
PA^S	**14/1/45**	**15/1/44**	**14/1/45**	**12/1/47**	**17/1/42**	**12/1/47**	**7/1/52**	**11/1/48**	**20/2/38**	**6/2/52**						
P∨CA	**17/0/43**	**16/0/44**	**17/0/43**	**15/1/44**	**17/0/43**	**20/1/39**	**17/0/43**	**20/0/40**	26/1/33	**19/0/41**	25/0/35					
P∨CA^S	**14/0/46**	**13/0/47**	**14/0/46**	**13/1/46**	**14/0/46**	**16/0/44**	**14/0/46**	**16/0/44**	25/0/35	**16/0/44**	**21/0/39**	6/40/14				
CA	**12/0/48**	**12/0/48**	**12/0/48**	**11/0/49**	**12/0/48**	**10/1/49**	**8/0/52**	**11/0/49**	**15/3/42**	**9/0/51**	**18/0/42**	**21/2/37**	**20/3/37**			
P∧CA^S	**10/0/50**	**10/0/50**	**10/0/50**	**10/0/50**	**11/0/49**	**8/1/51**	**6/1/53**	**10/0/50**	**7/0/53**	**7/0/53**	**12/0/48**	**18/3/39**	**21/4/35**	**20/3/37**		
CA^S	**7/1/52**	**7/1/52**	**7/1/52**	**6/1/53**	**8/1/51**	**5/1/54**	**5/1/54**	**6/1/53**	**6/0/54**	**6/1/53**	**7/3/50**	**15/0/45**	**16/0/44**	**17/1/42**	23/2/35	
AODE	**5/41/14**	**5/42/13**	**5/42/13**	20/12/28	3/53/4	27/6/27	28/4/28	29/5/26	31/1/28	31/2/27	**42/1/17**	**43/0/17**	**46/0/14**	**48/0/12**	**49/0/11**	**51/1/8**

Table 4.9: Win/Draw/Loss records of bias on 60 data sets for parent and child selection. The algorithms are sorted in descending order on the value of wins minus losses against AODE on bias.

	Bias															
W/L/D	PA	P∧CE	P∨CE	CE	PA^S	P∧CA	P∨CA	PE	P∨CE^S	CE^S	P∧CE^S	PE^S	CA	P∨CA^S	P∧CA^S	CA^S
PA																
P∧CE	**10/2/48**															
P∨CE	22/3/35	**48/5/7**														
CE	**17/2/41**	**43/4/13**	19/13/28													
PA^S	**36/3/21**	**43/5/12**	35/2/23	**42/1/17**												
P∧CA	30/0/30	**40/0/20**	30/2/28	**38/0/22**	32/0/28											
P∨CA	30/0/30	**41/0/19**	33/1/26	**39/0/21**	34/0/26	**19/33/8**										
PE	**6/2/52**	**19/5/36**	**8/7/45**	**16/4/40**	**18/2/40**	**19/0/41**	**20/0/40**									
P∨CE^S	**5/2/53**	**10/3/47**	**7/3/50**	**7/5/48**	**11/1/48**	**14/0/46**	**14/0/46**	20/9/31								
CE^S	**5/2/53**	**10/3/47**	**7/3/50**	**7/5/48**	**11/1/48**	**14/0/46**	**14/0/46**	20/9/31	**2/49/9**							
P∧CE^S	**4/2/54**	**8/4/48**	**7/3/50**	**7/5/48**	**11/1/48**	**14/0/46**	**14/0/46**	**17/10/33**	**0/50/10**	5/48/7						
PE^S	**4/2/54**	**8/3/49**	**8/3/49**	**8/5/47**	**11/2/47**	**14/0/46**	**14/0/46**	**13/13/34**	**2/41/17**	**3/40/17**	**3/41/16**					
CA	**21/0/39**	26/0/34	24/0/36	26/0/34	**20/0/40**	**20/3/37**	**22/0/38**	25/0/35	26/0/34	26/0/34	26/0/34	27/0/33				
P∨CA^S	**11/2/47**	**14/1/45**	**12/2/46**	**13/1/46**	**11/0/49**	**10/0/50**	**9/3/48**	**20/2/38**	**19/1/40**	**19/1/40**	**19/1/40**	**19/1/40**	**20/1/39**			
P∧CA^S	**4/0/56**	**9/0/51**	**5/0/55**	**5/1/54**	**4/0/56**	**4/2/54**	**5/1/54**	**9/0/51**	**10/0/50**	**10/0/50**	**11/0/49**	**13/0/47**	**16/0/44**	24/2/34		
CA^S	**5/1/54**	**7/1/52**	**5/1/54**	**7/1/52**	**5/2/53**	**7/0/53**	**7/0/53**	**8/1/51**	**7/3/50**	**7/3/50**	**8/3/49**	**9/2/49**	**13/2/45**	**20/0/40**	23/1/36	
AODE	**4/2/54**	**8/3/49**	**8/3/49**	**8/5/47**	**11/2/47**	**14/0/46**	**14/0/46**	**13/13/34**	**2/41/17**	**3/40/17**	**3/41/16**	1/53/6	33/0/27	**40/1/19**	**47/0/13**	**48/2/10**

Table 4.10: Win/Draw/Loss records of variance on 60 data sets for parent and child selection. The algorithms are sorted in descending order on the value of wins minus losses against AODE on variance.

	Variance															
W/L/D	PE^S	CE^S	P∨CE^S	P∧CE^S	P∧CA^S	P∨CA^S	PE	CA^S	P∧CE	P∧CA	P∨CE	CE	CA	P∨CA	PA	PA^S
PE^S																
CE^S	8/39/13															
P∨CE^S	7/41/12	4/46/10														
P∧CE^S	6/40/14	7/45/8	7/48/5													
P∧CA^S	**19/2/39**	**19/2/39**	**19/3/38**	**19/2/39**												
P∨CA^S	**17/2/41**	**17/2/41**	**17/3/40**	**17/2/41**	0/57/3											
PE	**13/11/36**	**16/9/35**	**16/9/35**	**16/10/34**	**39/0/21**	**41/0/19**										
CA^S	**16/1/43**	**17/1/42**	**17/1/42**	**17/1/42**	25/1/34	26/0/34	**16/1/43**									
P∧CE	**14/4/42**	**16/3/41**	**17/3/40**	**16/4/40**	**39/0/21**	**41/0/19**	**20/4/36**	**43/1/16**								
P∧CA	**14/0/46**	**14/0/46**	**14/0/46**	**14/0/46**	33/0/27	34/0/26	**16/0/44**	36/0/24	**17/0/43**							
P∨CE	**11/5/44**	**10/5/45**	**10/5/45**	**10/5/45**	35/2/23	**36/2/22**	**16/4/40**	**43/1/16**	**13/5/42**	**41/0/19**						
CE	**11/5/44**	**11/4/45**	**10/4/46**	**11/4/45**	36/0/24	**37/0/23**	**18/3/39**	**43/1/16**	**11/4/45**	**43/0/17**	21/8/31					
CA	**12/0/48**	**12/0/48**	**12/0/48**	**12/0/48**	24/0/36	25/0/35	**14/0/46**	29/1/30	**15/0/45**	**20/4/36**	**15/0/45**	**15/0/45**				
P∨CA	**12/0/48**	**12/0/48**	**12/0/48**	**12/0/48**	27/1/32	27/1/32	**13/0/47**	32/0/28	**13/0/47**	27/1/32	**15/0/45**	**15/0/45**	35/0/25			
PA	**7/2/51**	**7/2/51**	**7/2/51**	**7/2/51**	36/0/24	**37/0/23**	**8/2/50**	**43/1/16**	**12/3/45**	36/0/24	**16/3/41**	**19/2/39**	**43/0/17**	**44/0/16**		
PA^S	**7/1/52**	**5/1/54**	**5/1/54**	**5/1/54**	30/0/30	31/0/29	**7/1/52**	**40/2/18**	**7/1/52**	23/1/36	**6/2/52**	**5/3/52**	36/0/24	36/0/24	**8/2/50**	
AODE	6/53/1	14/39/7	13/41/6	14/40/6	**39/2/19**	**41/2/17**	**37/10/13**	**43/1/16**	**42/4/14**	**46/0/14**	**44/5/11**	**45/4/11**	**48/0/12**	**48/0/12**	**51/2/7**	**53/1/6**

Table 4.11: Win/Draw/Loss records of RMSE on 60 data sets for parent and child selection. The algorithms are sorted in descending order on the value of wins minus losses against AODE on RMSE.

	RMSE														
W/L/D	CE^S	P∧CE^S	P∨CE^S	P∨CE	PE^S	CE	P∧CE	PE	P∧CA	PA	P∨CA	P∧CA^S	PA^S	CA	P∨CA^S	CA^S
CE^S																
P∧CE^S	6/45/9															
P∨CE^S	7/49/4	7/50/3														
P∨CE	31/2/27	31/2/27	31/2/27													
PE^S	6/42/12	8/40/12	8/40/12	27/2/31												
CE	28/6/26	28/6/26	28/6/26	**19/7/34**	28/6/26											
P∧CE	26/4/30	27/5/28	26/4/30	**20/4/36**	28/4/28	24/7/29										
PE	20/11/29	20/12/28	20/12/28	29/3/28	20/12/28	33/2/25	35/2/23									
P∧CA	24/1/35	23/1/36	23/1/36	**17/0/43**	24/1/35	**21/1/38**	**21/1/38**	**20/0/40**								
PA	**21/2/37**	**21/2/37**	**21/2/37**	26/2/32	**21/2/37**	32/2/26	29/2/29	**19/4/37**	**40/0/20**							
P∨CA	**12/2/46**	**12/3/45**	**11/2/47**	**18/0/42**	**11/2/47**	**18/2/40**	**17/0/43**	**12/0/48**	**19/1/40**	**17/1/42**						
P∧CA^S	**11/0/49**	**11/0/49**	**11/0/49**	**9/0/51**	**12/0/48**	**10/0/50**	**10/0/50**	**12/0/48**	**8/0/52**	**8/0/52**	25/1/34					
PA^S	**10/3/47**	**10/3/47**	**9/3/48**	**9/2/49**	**10/3/47**	**11/2/47**	**11/2/47**	**9/3/48**	**21/1/38**	**4/3/53**	30/0/30	**48/0/12**				
CA	**11/0/49**	**11/0/49**	**11/0/49**	**8/0/52**	**11/0/49**	**10/0/50**	**10/0/50**	**9/0/51**	**9/2/49**	**11/0/49**	**23/0/37**	**37/0/23**	**17/0/43**			
P∨CA^S	**11/1/48**	**11/2/47**	**10/1/49**	**17/0/43**	**10/1/49**	**16/1/43**	**16/0/44**	**11/0/49**	**21/0/39**	**15/1/44**	7/39/14	31/3/26	26/0/34	36/0/24		
CA^S	**8/0/52**	**8/0/52**	**8/0/52**	**6/0/54**	**9/0/51**	**6/0/54**	**7/0/53**	**8/0/52**	**4/0/56**	**7/0/53**	**18/0/42**	**23/0/37**	**8/0/52**	**13/2/45**	**17/0/43**	
AODE	6/42/12	7/41/12	8/40/12	27/2/31	1/55/4	26/6/28	28/4/28	28/12/20	35/1/24	**37/2/21**	**47/2/11**	**48/0/12**	**47/3/10**	**49/0/11**	**49/1/10**	**51/0/9**

Chapter 5

Subsumption Resolution

The previous chapter has shown that the accuracy of NB can be significantly improved by the addition of BSE and so can be the accuracy of AODE when a statistical test is employed. However, due to repeated accuracy evaluation of attribute subsets on algorithms to which it is applied, BSE tends to have high computational overheads, especially when applied to algorithms with high classification time complexity. In response to this drawback, we develop a new type of semi-naive Bayesian operation, Subsumption Resolution (SR), that efficiently and effectively eliminates a special type of closely related attributes at classification time.

This chapter first introduces three extreme types of interdependencies between attributes: the generalization, substitution and duplication relationships. The relationships between them are examined. Further, we discuss the relationships between surjection and generalization, bijection and substitution. Next, we present Subsumption Resolution, a technique that can efficiently identify pairs of attribute values such that one is a generalization of the other and deletes the generalization at classification time. We show that this is the theoretically correct adjustment for such an interdependence relationship and demonstrate experimentally that it can in practice considerably improve both classification accuracy and the precision of conditional probability estimates. We study the effect of SR on classification accuracy and probabilistic prediction by applying it to NB and AODE and compare this method to five state-of-the-art semi-naive Bayesian methods. Finally, we investigate the effect of SR on NB and AODE by using both labeled and unlabeled data.

5.1 The Generalization, Substitution and Duplication Relationships

One extreme type of interdependency between attributes results in a value of one being a generalization of a value of the other. For example, let *Gender* and *Pregnant* be two attributes. *Gender* has two values: *female* and *male*, and *Pregnant* has two values: *yes* and *no*. If *Pregnant=yes*, it follows that *Gender=female*. Therefore, *Gender=female* is a generalization of *Pregnant=yes*. Likewise, *Pregnant=no* is a generalization of *Gender=male*. We formalize this relationship as:

Definition 1. (Generalization and Specialization) *For two attribute values x_i and x_j, if $P(x_j \mid x_i) = 1.0$ then x_j is a* generalization *of x_i and x_i is a* specialization *of x_j.*

In a special case when x_i is a generalization and specialization of x_j, x_i is a substitution of x_j.

Definition 2. (Substitution) *For two attribute values x_i and x_j, if $P(x_j \mid x_i) = 1.0$ and $P(x_i \mid x_j) = 1.0$, x_i is a substitution of x_j and so is x_j of x_i. For two attributes X_i and X_j, we say that X_i is a substitution of X_j if the following condition holds:*

$\forall a \exists b P(x_j^b \mid x_i^a) = P(x_i^a \mid x_j^b) = 1.0$*, where* $a, b \in \{1, \ldots, |X_i|\}$*, x_i^a is the ath value of X_i and x_j^b is the bth value of X_j.*

Definition 3. (Duplication) *For two attribute values x_i and x_j, if x_i is a substitution of x_j and $x_i = x_j$ then x_i is a duplication of x_j. For two attributes X_i and X_j, we say that X_i is a duplication of X_j if $x_i = x_j$ for all instances.*

In Table 5.1, because $P(X_j{=}0 \mid X_i{=}0) = 1.0$ and $P(X_i{=}1 \mid X_j{=}1) = 1.0$, $X_j{=}0$ is a generalization of $X_i{=}0$ and $X_i{=}1$ is a generalization of $X_j{=}1$.

Table 5.2 illustrates an example of substitution. $P(X_j{=}2 \mid X_i{=}0) = 1.0$ and $P(X_i{=}0 \mid X_j{=}2) = 1.0$, hence $X_j{=}2$ is a substitution of $X_i{=}0$ and so is $X_i{=}0$ of $X_j{=}2$. Likewise, $X_j{=}0$ is a substitution of $X_i{=}1$ and so is $X_i{=}1$ of $X_j{=}0$. As both $X_i{=}0$ and $X_i{=}1$ have substitutions, X_i is a substitution of X_j. In Table 3.2 (Section 3.1.4), X_0 is a substitution of X_1, which is a negation of X_0.

As illustrated in Table 5.3, X_i is a duplication of X_j.

Table 5.1: Generalization

X_i	X_j
0	0
0	0
0	0
1	0
1	0
1	0
1	1
1	1

Table 5.2: Substitution

X_i	X_j
0	2
0	2
0	2
0	2
1	0
1	0
1	0
1	0

Table 5.3: Duplication

X_i	X_j
0	0
0	0
0	0
0	0
1	1
1	1
1	1
1	1

From the perspective of set theory, the condition that $P(x_j \mid x_i) = 1.0$ is equivalent to $T_{x_i} \subseteq T_{x_j}$, where T_{x_i} and T_{x_j} are the sets of the training cases with value x_i and x_j respectively. Therefore, the alternative of Definition 1 is:

Definition 1′. (Generalization and Specialization) *For two attribute values x_i and x_j, if $T_{x_i} \subseteq T_{x_j}$ then x_j is a* generalization *of x_i and x_i a* specialization *of x_j.*

As $P(x_j \mid x_i) = P(x_i \mid x_j) = 1.0 \equiv T_{x_i} = T_{x_j}$ (i.e. $T_{x_i} \subseteq T_{x_j}$ and $T_{x_j} \subseteq T_{x_i}$), Definition 2 is equivalent to:

Definition 2′. (Substitution) *For two attribute values x_i and x_j, if $T_{x_i} = T_{x_j}$, x_i is a substitution of x_j and so is x_j of x_i. For two attributes X_i and X_j, we say that X_i is a substitution of X_j if the following condition holds:*

> *$\forall a \exists b T_{x_i^a} = T_{x_j^b}$, where $a, b \in \{1, \ldots, |X_i|\}$, $T_{x_i^a}$ is the set of training cases with the ath value of X_i and $T_{x_j^b}$ is the set of training cases with the bth value of X_j.*

Proposition 1. *If X_j is a surjection of X_i, then every x_i has a generalization x_j.*

Proof. Because X_j is a surjection of X_i, we have $X_j = f(X_i)$. Then $\forall a \in \{1, \ldots, |X_i|\}$, $\exists b \in \{1, \ldots, |X_j|\}$ such that $f(x_i^a) = x_j^b$, where x_i^a is the ath value of X_i and x_j^b is the bth value of X_j. It follows that $P(x_j^b \mid x_i^a) = P(f(x_i^a) \mid x_i^a) = 1.0$. Therefore, every x_i has a generalization x_j.

□

Proposition 2. *X_j is a bijection of X_i if and only if X_j is a substitution of X_i.*

Proof. Necessary condition: we assume that X_j is a bijection of X_i and we shall prove that X_j is a substitution of X_i.

$X_j = f(X_i)$. Then $\forall a \in \{1, \ldots, |X_i|\}, \exists b \in \{1, \ldots, |X_j|\}$ such that $f(x_i^a) = x_j^b$. It follows that $P(x_j^b \mid x_i^a) = P(f(x_i^a) \mid x_i^a) = 1.0$. As f is bijective, it has an inverse function: f^{-1}. Then $f^{-1}(x_j^b) = x_i^a$ and $P(x_i^a \mid x_j^b) = P(f^{-1}(x_j^b) \mid x_j^b) = 1.0$. Therefore, X_j is a substitution of X_i.

Sufficient condition: we assume that X_j is a substitution of X_i and we shall prove that X_j is a bijection of X_i.

If X_j is a substitution of X_i, then we have $\forall a \exists b P(x_j^b \mid x_i^a) = P(x_i^a \mid x_j^b) = 1.0$, where $a, b \in \{1, \ldots, |X_i|\}$. Assuming that $\exists c \in \{1, \ldots, |X_j|\}, c \neq b$ such that $P(x_j^c \mid x_i^a) = 1.0$. But if $P(x_j^b \mid x_i^a) = 1.0$ and $P(x_j^c \mid x_i^a) = 1.0$, it follows that $b = c$. That is, every element of X_i corresponds to one and only one element of X_j. Because $\forall b \exists a P(x_j^b \mid x_i^a) = 1.0$, each element in X_j is mapped to by some elements of X_i. Therefore, X_j is a surjection of X_i.

Assuming that $\exists d \in \{1, \ldots, |X_i|\}, d \neq a$ such that $P(x_j^b \mid x_i^d) = 1.0$. If $P(x_i^a \mid x_j^b) = 1.0$ and $P(x_j^b \mid x_i^d) = 1.0$, then $P(x_i^a \mid x_i^d) = 1.0$. This can only be true when $a = d$. That is, every element of the X_j is mapped to by exactly one element of X_i. Therefore, X_j is a bijection of X_i.

□

Propositions 3 and 4 follow immediately from the definitions of the generalization, substitution and duplication relationships.

Proposition 3. *The duplication and substitution relationships are symmetrical and transitive.*

Proposition 4. *The generalization relationship is transitive.*

It is interesting that the specialization-generalization relationship can be defined in terms of the definitions of Generalization, Specialization, Substitution and Duplication. Assume that *Generalization*, *Specialization*, *Substitution* and *Duplication* are four attributes, all of which have two values: *yes* and *no*. We have $P(Substitution{=}yes \mid Duplication{=}yes) = 1.0$, $P(Generalization{=}yes \mid Substitution{=}yes) = 1.0$ and $P(Specialization{=}yes \mid Substitution{=}yes) = 1.0$.

Therefore, *Duplication=yes* is a specialization of *Substitution=yes*, which is a specialization of *Generalization=yes* and *Specialization=yes*. Since the specialization–generalization relationship is transitive, *Duplication=yes* is also a specialization of *Generalization=yes* and *Specialization=yes*. Figure 5.1 illustrates the relationship between them.

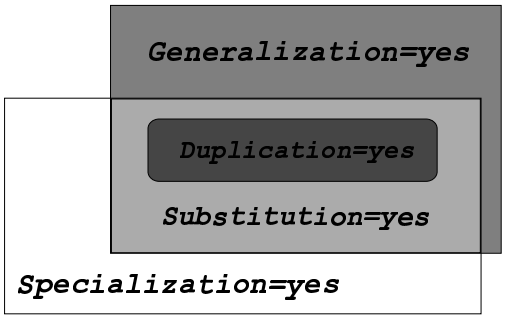


Figure 5.1: Relationship between Duplication, Substitution, Generalization and Specialization.

The generalization relationship is very common in many forms of real world data. For example, *City=Sydney* is a specialization of *Country=Australia* and *CountryCode*=61 is a substitution of *Country=Australia*. Given an example with *City=Sydney*, *Country=Australia* and *CountryCode*=61, NB will effectively give three times the weight to evidence relating to *Country=Australia* relative to the situation if only one of these attributes were considered. Ignoring such redundancy may reduce NB's accuracy. The next section is devoted to resolving this problem.

5.2 Subsumption Resolution (SR)

Subsumption Resolution (SR) [Zheng and Webb, 2006] identifies at classification time pairs of attribute values such that one appears to subsume (be a generalization of) the other. It then deletes the generalization.

5.2.1 Deletion of Generalizations

Theorem 1. *If x_j is a generalization of $x_i, 1 \leq i \leq n, 1 \leq j \leq n, i \neq j$ then $P(y \mid x_1, \ldots, x_n) = P(y \mid x_1, \ldots, x_{j-1}, x_{j+1}, \ldots x_n)$.*

Proof. Note, $\forall Z$, given $P(x_j \mid x_i) = 1.0$, it follows that $P(Z \mid x_i, x_j) = P(Z \mid x_i)$ and hence $P(x_i, x_j, Z) = P(x_i, Z)$. Therefore,

$$\begin{aligned} & P(y \mid x_1, \ldots, x_n) \\ = \; & \frac{P(y, x_1, \ldots, x_n)}{P(x_1, \ldots, x_n)} \\ = \; & \frac{P(y, x_1, \ldots, x_{j-1}, x_{j+1}, \ldots, x_n)}{P(x_1, \ldots, x_{j-1}, x_{j+1}, \ldots, x_n)} \\ = \; & P(y \mid x_1, \ldots, x_{j-1}, x_{j+1}, \ldots, x_n) \end{aligned}$$

□

Given $P(y \mid x_1, \ldots, x_n) = P(y \mid x_1, \ldots, x_{j-1}, x_{j+1}, \ldots x_n)$, deleting the generalization x_j from a Bayesian classifier should not be harmful. Further, such deletion may improve a classifier's estimates if the classifier makes unwarranted assumptions about the relationship of x_j to the other attributes when estimating intermediate probability values, such as NB's independence assumption.

To illustrate this, consider the data presented in Table 5.4 for a hypothetical exam-

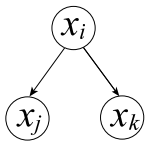
Table 5.4: NB is adversely affected by the presence of generalizations.

Gender	*Pregnant*	*MaleHormone*	*Normal*
male	no	3	yes
female	yes	3	yes
female	yes	2	yes
female	yes	2	yes
male	no	1	no
female	no	3	no
female	no	4	no
female	yes	4	no

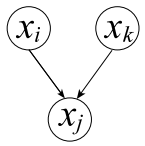
ple with three attributes *Gender*, *Pregnant* and $MaleHormone$ and class $Normal$. *Pregnant=yes* is a specialization of *Gender=female* and *Gender=male* is a specialization of *Pregnant=no*. As these two attributes are highly related, NB will misclassify $\langle$*Gender=male*, *Pregnant=no*, $MaleHormone$=3$\rangle$ as $Normal$=*no*, even though it occurs in the training data. In effect NB double counts the evidence from *Pregnant=no*, due to the presence of its specialization *Gender=male*. The new object can be correctly classified as $Normal$=*yes* by deleting attribute value *Pregnant=no*.

In contrast, if *Gender=female* we cannot make any definite conclusion with respect to the value of *Pregnant*, nor about the value of *Gender* if *Pregnant=no*. If both of these values (*Gender=female* and *Pregnant=no*) are present, deleting either one will lose information. Therefore, both should be used for classification if neither attribute-value is a generalization of the other. In the case when x_i is a substitution of x_j, only one of the two attribute-values is used for classification.

Figure 5.2 illustrates the process of deleting attribute values in the four different

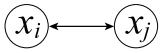


(a) Multiple Children



(b) Multiple Parents

(c) Transitive Relation



(d) Substitution

Figure 5.2: Deletion of generalizations. (a) *Multiple Children: deleting children x_j and x_k*, (b) *Multiple Parents: deleting child x_j*, (c) *Transitive Relation: Deleting x_j and x_k*, and (d) *Substitution: deleting either x_i or x_j.*

cases. Parent nodes in the figure are specializations of child nodes. In the case

of Figure 5.2 (a), we delete both x_j and x_k by applying Theorem 1 twice. When an attribute value is a generalization of other attribute values (in Figure 5.2 (b), x_j is a generalization of x_i and x_k), we delete the attribute value (x_j). If x_k is a generalization of x_j and so is x_j of x_i, as illustrated in Figure 5.2 (c), we delete x_k and x_j. In Figure 5.2 (d), x_i is a substitution of x_j (and so is x_j of x_i), we delete either x_i or x_j.

Attribute selection is a well-studied problem in machine learning [Langley, 1994; Kohavi and John, 1997; Blum and Langley, 1997; Guyon and Elisseeff, 2003; Liu and Yu, 2005]. An in-depth theoretical analysis of interdependencies between attributes can be found in [Jakulin, 2005]. The main difference between SR and other attribute selection methods is that SR deletes attribute values while other methods delete complete attributes. Section 5.6.6 shows experimentally that deletion of attribute values and complete attribute may result in different effect on classification.

5.2.2 Criterion for Identifying Generalizations

SR requires a method for inferring from the training data whether one attribute value is a generalization of another. It uses the criterion

$$|T_{x_i}| = |T_{x_i,x_j}| \geq l$$

to infer that x_j is a generalization of x_i, where $|T_{x_i}|$ is the number of training cases with value x_i, $|T_{x_i,x_j}|$ is the number of training cases with both values, and l is a user-specified minimum frequency. Figure 5.3 illustrates the criterion from the perspective of set theory, as $|T_{x_i}| = |T_{x_i,x_j}|$ is equivalent to $T_{x_i} \subseteq T_{x_j}$. That is, if $|T_{x_i}| \geq l$ and $T_{x_i} \subseteq T_{x_j}$, x_j is a generalization of x_i. In the case when $|T_{x_i}| \geq l$ and $T_{x_i} = T_{x_j}$, x_j is a substitution of x_i. It might be thought that a less arbitrary technique than a "magic number" l might be obtained using either statistical or information theoretic approaches, but each would still require an arbitrary critical value, such as α.

5.2.3 Lazy Learning

It is superficially attractive to pre-check the generalization relation at training time. The time complexity of creating the dependency matrix is $O(n^2v^2)$, as it requires for

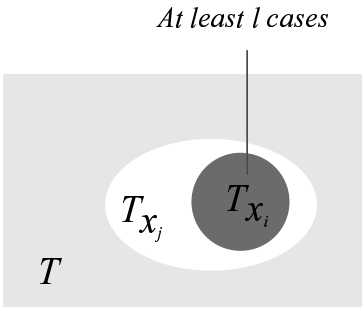


Figure 5.3: Criterion to infer that x_j is a generalization of x_i. T_{x_i} and T_{x_j} are the sets of the training cases with value x_i and x_j respectively.

each pair of attributes, every pairwise combination of their respective values to be considered. At classification time, scanning the dependency matrix to delete attributes has time complexity of $O(n^2)$. However, if we check attribute-value pairs for generalization relationships at classification time (so there is no additional computation for dependency matrix at training time), time complexity is $O(n^2)$ as well. Therefore, SR delays the computation of elimination until classification time and deletes different attributes depending upon which attribute values are instantiated in the object being classified.

Unlike other lazy learning algorithms, SR does not need to keep all the training examples at classification time as it only utilizes the tables of probability estimates formed at training time to delete attributes. Hence, it has lower memory requirements than other lazy learning algorithms.

5.3 NB and AODE with SR

SR is suited to algorithms without model selection, such as NB and AODE, as the attribute value elimination can be directly applied to the algorithms at classification time. However, it might not be suited to algorithms with model selection, such as decision trees, as it may require different models to be built by the algorithm for each object to be classified. This is because it is not known whether it is appropriate to include an attribute-value into a model until it is known which other attribute-values are present for an instance.

NB and AODE are incremental. When a new instance is available, they do not need to reexamine all earlier training instances but need only update probability estimates. SR detects and eliminates highly dependent attribute values at classification time using a two-dimensional table of probability estimates indexed by attribute values generated at training time. This probability estimates table can be updated incrementally. In consequence, SR does not interfere with NB and AODE's capacity for incremental learning.

5.3.1 NB with SR

In the context of NB, SR deletes generalization attribute values if a specialization is detected and applies NB to the resulting attribute value set. We denote NB with Subsumption Resolution as NB^{SR}.

Classification of instance $\mathbf{x} = \langle x_1, \ldots, x_n \rangle$ consists of two steps:

1. Set R to $\{x_{1 \leq i \leq n} \mid \neg \exists x_{1 \leq j \leq n}\, x_i \neq x_j \wedge P(x_i \mid x_j) = 1.0\}$.
2. Estimate $P(y, \mathbf{x})$ by

$$\hat{P}(y, \mathbf{x}) = \hat{P}(y) \prod_{x_i \in R} \hat{P}(x_i \mid y).$$

To identify the specialization-generalization relationship, NB^{SR} must generate at training time a two-dimensional table of probability estimates for each attribute-value, conditioned by each other attribute-value in addition to the two probability estimate tables generated by NB, space complexity $O(knv + (nv)^2)$. The time complexity of forming the additional two-dimensional probability estimate table is $O(tn^2)$. Classification of a single example requires considering each pair of attributes to detect dependencies and is of time complexity $O(n^2 + kn)$. The space complexity is $O(knv + (nv)^2)$.

5.3.2 AODE with SR

When SR is applied to AODE, the resulting classifier acts as AODE except that it deletes generalization attribute-values from use in any role in the classifier if a specialization is detected, and aggregates the predictions of all qualified ODEs using

the remaining attribute-values. We denote AODE with Subsumption Resolution as AODE^{SR}.

Classification consists of two steps:

1. Set R to $\{x_{1\leq i\leq n} \mid \neg\exists x_{1\leq j\leq n}\, x_i \neq x_j \wedge P(x_i \mid x_j) = 1.0\}$.

2. Estimate $P(y, \mathbf{x})$ by

$$\hat{P}(y, \mathbf{x}) = \frac{\sum_{x_i \in R \wedge F(x_i) \geq m} \hat{P}(y, x_i) \prod_{x_j \in R} \hat{P}(x_j \mid y, x_i)}{|\{x_i \in R : F(x_i) \geq m\}|}.$$

AODE^{SR} has identical time and space complexity to AODE. At training time it behaves identically to AODE. At classification time, it must check all attribute-value pairs for generalization relationships, an additional operation of time complexity $O(n^2)$. However, the time complexity of AODE at classification time is $O(kn^2)$ and so this additional computation does not increase the time complexity.

5.4 NB and AODE with BSE

The previous chapter has shown that the types of attribute elimination that remove child attributes from within the constituent ODEs can significantly reduce bias and error of AODE, but only if a statistical test is employed to provide variance management. CE^S, P$\wedge$CE^S and P$\vee$CE^S share a similar level of error and RMSE. P$\vee$CE^S performs three types of attribute eliminations and selects the option that most improves accuracy at each step, hence in practice it has substantially higher training time overheads compared to CE^S and P$\wedge$CE^S. P$\wedge$CE^S obtains lower error than CE^S eight times and higher error seven times. For simplicity, in this chapter, we only consider P$\wedge$CE^S. We use AODE^{BSE} to indicate P$\wedge$CE^S and NB^{BSE} to indicate NB with BSE.

5.5 Complexity Summary

Table 5.5 summarizes the complexity of NB, NB^{SR}, NB^{BSE}, AODE, AODE^{BSE} and AODE^{SR}. AODE^{BSE} and NB^{BSE} have the highest and second highest training

time complexity. NB and NB^{BSE} have relatively low classification time complexity. AODE^{BSE} has the highest training space complexity.

Table 5.5: Complexity of NB and AODE with SR and BSE

	Training		**Classification**	
Algorithm	**Time**	**Space**	**Time**	**Space**
NB	$O(tn)$	$O(knv)$	$O(kn)$	$O(knv)$
NB^{SR}	$O(tn^2)$	$O(knv + (nv)^2)$	$O(n^2 + kn)$	$O(knv + (nv)^2)$
NB^{BSE}	$O(tkn^2)$	$O(tn + knv)$	$O(kn)$	$O(knv)$
AODE	$O(tn^2)$	$O(k(nv)^2)$	$O(kn^2)$	$O(k(nv)^2)$
AODE^{SR}	$O(tn^2)$	$O(k(nv)^2)$	$O(kn^2)$	$O(k(nv)^2)$
AODE^{BSE}	$O(tkn^3)$	$O\big(tkn + k(nv)^2\big)$	$O(kn^2)$	$O(k(nv)^2)$

5.6 Experimental Results

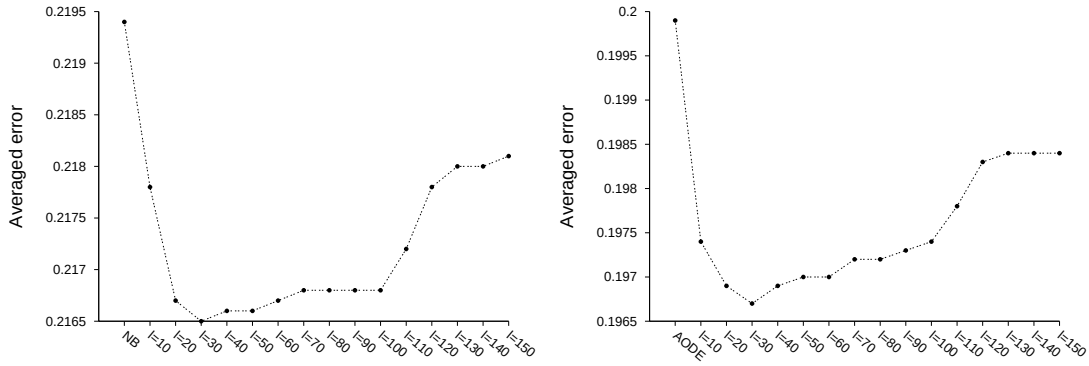
In this section, we evaluate the efficacy of SR by applying it to NB and AODE. We first perform an empirical study to select a minimum frequency for identifying generalizations. Pairwise win/draw/loss comparisons for NB and its variants (NB with SR and BSE) and AODE and its variants (AODE with SR and BSE) come next. Then, we use the Friedman and Nemenyi tests to compare NB^{SR} and AODE^{SR} with five semi-naive Bayesian algorithms, all of which significantly reduce NB's error. The experiments are run in the same manner as the experiments described in Section 3.3.2.

5.6.1 Minimum Frequency for Identifying Generalizations

As there does not appear to be any formal method to select an appropriate value for l, we perform an empirical study to select it. We present the error and RMSE results in the range of $l = 10$ to $l = 150$ with an increment of 10.

5.6.1.1 Mean Error and RMSE

Averaged results across all data sets provides a simplistic overall measure of relative performance. We present the averaged error and RMSE of NB^{SR} and AODE^{SR} across 60 data sets as a function of l in Figure 5.4 and 5.5. The values on the x-axis are the minimum frequencies and the values on the y-axis are the averaged error and RMSE across 60 data sets. In order to provide comparison with NB and AODE, we also include NB and AODE's error and RMSE in each graph.



(a) NB and NB^{SR} with l (b) AODE and AODE^{SR} with l

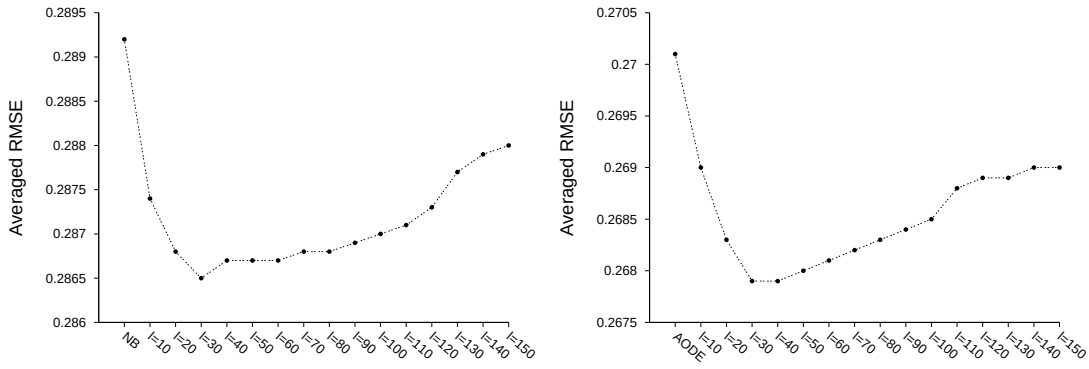
Figure 5.4: Averaged error across 60 data sets, as function of l.

As can be seen that NB^{SR} and AODE^{SR} enjoy lower mean error and RMSE at all settings of l compared to NB and AODE respectively.

5.6.1.2 Error

Table 5.6 presents the win/draw/loss records of error for NB against NB^{SR} and AODE against AODE^{SR}. The p value is the outcome of a one-tailed binomial sign test. We assess a difference as significant if p is less than 0.05. Boldface numbers indicate that wins against losses are statistically significant.

SR significantly improves NB and AODE's accuracy when $20 \leq l \leq 150$. It marginally improves AODE's accuracy when $l = 10$.



(a) NB and NB^{SR} with l (b) AODE and AODE^{SR} with l

Figure 5.5: Averaged RMSE across 60 data sets, as function of l.

5.6.1.3 RMSE

The win/draw/loss records of RMSE for NB against NB^{SR} and AODE against AODE^{SR} are presented in Table 5.7.

At all settings of l, NB and AODE's RMSE can be significantly reduced by the addition of SR.

5.6.1.4 Minimum Frequency Selection

A larger value of l can reduce the risk of incorrectly inferring that one value subsumes another, but at the same time reduces the number of true generalizations that are detected. In the current work, the setting $l = 30$ is selected as it is a widely used heuristic for the minimum number of examples from which an inductive inference should be drawn.

Table 5.6: Win/Draw/Loss comparison of Error (NB vs. NB^{SR} with l and AODE vs. AODE^{SR} with l)

	NB^{SR}														
W/D/L	l=10	l=20	l=30	l=40	l=50	l=60	l=70	l=80	l=90	l=100	l=110	l=120	l=130	l=140	l=150
NB	20/12/28	**14/16/30**	**11/19/30**	**9/23/28**	**11/22/27**	**9/25/26**	**7/27/26**	**7/30/23**	**8/32/20**	**6/34/20**	**4/37/19**	**5/38/17**	**5/37/18**	**5/39/16**	**6/38/16**
p	0.1562	**0.0113**	**0.0022**	**0.0013**	**0.0069**	**0.0030**	**0.0007**	**0.0026**	**0.0178**	**0.0047**	**0.0013**	**0.0085**	**0.0053**	**0.0133**	**0.0262**
	AODE^{SR}														
W/D/L	l=10	l=20	l=30	l=40	l=50	l=60	l=70	l=80	l=90	l=100	l=110	l=120	l=130	l=140	l=150
AODE	18/12/30	**16/15/29**	**11/18/31**	**8/20/32**	**5/22/33**	**5/23/32**	**7/24/29**	**4/28/28**	**6/26/28**	**6/26/28**	**6/29/25**	**6/30/24**	**6/31/23**	**5/31/24**	**5/32/23**
p	0.0557	**0.0362**	**0.0014**	**0.0001**	**<0.0001**	**<0.0001**	**0.0002**	**<0.0001**	**0.0001**	**0.0001**	**0.0004**	**0.0007**	**0.0012**	**0.0003**	**0.0005**

Table 5.7: Win/Draw/Loss comparison of RMSE (NB vs. NB^{SR} with l and AODE vs. AODE^{SR} with l)

	NB^{SR}														
W/D/L	l=10	l=20	l=30	l=40	l=50	l=60	l=70	l=80	l=90	l=100	l=110	l=120	l=130	l=140	l=150
NB	**11/14/35**	**9/18/33**	**7/22/31**	**7/24/29**	**6/26/28**	**4/28/28**	**4/30/26**	**4/30/26**	**3/34/23**	**2/39/19**	**4/37/19**	**4/38/18**	**4/39/17**	**4/39/17**	**3/41/16**
p	**0.0003**	**0.0001**	**0.0001**	**0.0002**	**0.0001**	**<0.0001**	**<0.0001**	**<0.0001**	**<0.0001**	**0.0001**	**0.0013**	**0.0022**	**0.0036**	**0.0036**	**0.0022**
	AODE^{SR}														
W/D/L	l=10	l=20	l=30	l=40	l=50	l=60	l=70	l=80	l=90	l=100	l=110	l=120	l=130	l=140	l=150
AODE	**12/12/36**	**11/14/35**	**10/15/35**	**5/19/36**	**4/20/36**	**3/20/37**	**3/21/36**	**3/24/33**	**3/25/32**	**3/25/32**	**3/28/29**	**3/30/27**	**3/30/27**	**3/30/27**	**3/32/25**
p	**0.0004**	**0.0003**	**0.0001**	**<0.0001**	**<0.0001**	**<0.0001**	**<0.0001**	**<0.0001**	**<0.0001**	**<0.0001**	**<0.0001**	**<0.0001**	**<0.0001**	**<0.0001**	**<0.0001**

5.6.2 Effect of SR on NB

To explore the effect of SR, we compare the error, bias, variance and RMSE of NB^{SR} using the minimum frequency of 30 to NB. We also provide a comparison with NB^{BSE}. Two variants of BSE are evaluated, one employing a binomial sign test and the other not. Both significantly improve upon NB's accuracy. NB^{BSE} without a binomial sign test enjoys a significant error advantage over with (win/draw/loss being 36/4/20). Therefore, we only present the results of NB^{BSE} without a binomial sign test.

Table 5.8 presents the win/draw/loss records for NB^{SR} against NB and NB^{BSE} on sixty data sets. The win/draw/loss records for NB^{BSE} against NB and NB^{SR} is shown in Table 5.9.

Table 5.8: Win/Draw/Loss: NB^{SR} vs. NB and NB^{BSE}

	NB		**NB^{BSE}**	
	W/D/L	p	W/D/L	p
Error	30/19/11	**0.0022**	25/4/31	0.2522
Bias	31/18/11	**0.0014**	12/3/45	**<0.0001**
Variance	13/23/24	**0.0494**	43/4/13	**<0.0001**
RMSE	31/22/7	**0.0001**	27/4/29	0.4469

Table 5.9: Win/Draw/Loss: NB^{BSE} vs. NB and NB^{SR}

	NB		**NB^{SR}**	
	W/D/L	p	W/D/L	p
Error	36/4/20	**0.0220**	31/4/25	0.2522
Bias	52/3/5	**<0.0001**	45/3/12	**<0.0001**
Variance	11/4/45	**<0.0001**	13/4/43	**<0.0001**
RMSE	34/6/20	**0.0380**	29/4/27	0.4469

Figure 5.6 graphs the relative error, bias, variance and RMSE of the three classifiers. The values on the y-axis are the outcome for NB^{BSE} divided by that for NB.

The values of the x-axis are the outcome for NB^{SR} divided by that for NB. Each point on the graph represents one data set. Points on the left of the vertical line at NB^{SR}/ NB= 1 in each subgraph are those for which NB^{SR} has better results than NB. Points below the horizontal line at NB^{BSE}/ NB = 1 indicate that NB^{BSE} wins in those domains compared with NB. Points below the line $X = Y$ represent that NB^{SR} has higher values than those of NB^{BSE}.

To scale Figure 5.6 appropriately, we present results on all data sets except Mushroom, on which SR and BSE reduce NB's error, bias and RMSE substantially. The error of NB on Mushroom is 0.0519 and that of NB^{SR} and NB^{BSE} are 0.0208 and 0.0108 respectively. The RMSE of NB is 0.1986 and that of NB^{SR} and NB^{BSE} are 0.1221 and 0.1091 respectively.

Error

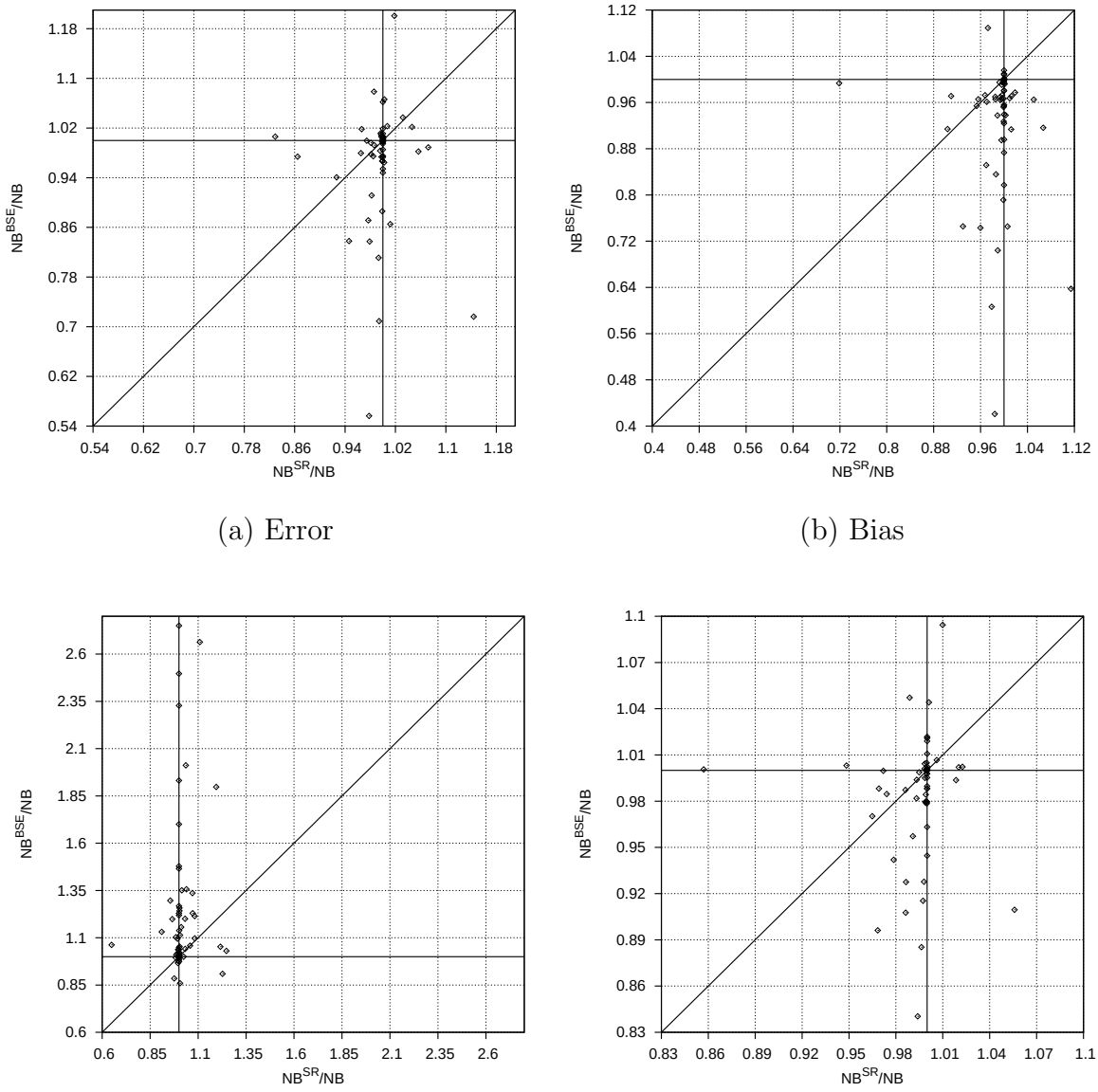
The win/draw/loss records indicate that both NB^{BSE} and NB^{SR} enjoy a significant advantage over NB. NB^{BSE} frequently achieves lower error than NB^{SR}. However, the frequency of these differences does not quite achieve the level of statistical significance. From the error graph in Figure 5.6 we can see that the majority of the points are below the horizontal line at NB^{BSE}/ NB = 1 and on the left of the vertical line at NB^{SR}/ NB = 1.

Bias

The win/draw/loss records for bias show that the advantages of NB^{BSE} and NB^{SR} are significant compared to NB. The advantage of NB^{BSE} in bias over NB^{SR} is also significant. Most of points are on the left of the vertical line at NB^{SR}/ NB = 1 and below the line $X = Y$ in the bias graph.

Variance

NB enjoys a significant variance advantage over NB^{BSE} and NB^{SR}. The advantage of NB^{SR} in variance over NB^{BSE} is clear. In the variance graph, most points are on the right of the vertical line at NB^{SR}/ NB = 1 and above the line $X = Y$.



(a) Error (b) Bias

(c) Variance (d) RMSE

Figure 5.6: Comparison of error, bias, variance and RMSE for NB, NB^{SR} and NB^{BSE}.

RMSE

Both NB^{BSE} and NB^{SR} significantly reduce the RMSE of NB. NB^{BSE} frequently achieves lower RMSE than NB^{SR}. However, there is no significant difference identified. Most of points are on the left of the vertical line at NB^{SR}/ NB = 1 and below the horizontal line at NB^{BSE}/ NB = 1 in the RMSE graph.

5.6.3 Effect of SR on AODE

The win/draw/loss records for AODE^{SR} against AODE and AODE^{BSE} on sixty data sets are shown in Table 5.10. Table 5.11 presents the win/draw/loss records for AODE^{BSE} against AODE and AODE^{SR}.

Table 5.10: Win/Draw/Loss: AODE^{SR} vs. AODE and AODE^{BSE}

	AODE		**AODE** ^{BSE}	
	W/D/L	p	W/D/L	p
Error	31/18/11	**0.0014**	27/15/18	0.1163
Bias	38/18/4	**<0.0001**	34/15/11	**0.0004**
Variance	14/17/29	**0.0158**	17/14/29	0.0519
RMSE	35/15/10	**0.0001**	35/12/13	**0.0010**

Table 5.11: Win/Draw/Loss: AODE^{BSE} vs. AODE and AODE^{SR}

	AODE		**AODE**^{SR}	
	W/D/L	p	W/D/L	p
Error	14/41/5	**0.0318**	18/15/27	0.1163
Bias	16/41/3	**0.0022**	11/15/34	**0.0004**
Variance	6/40/14	0.0577	29/14/17	0.0519
RMSE	12/41/7	0.1796	13/12/35	**0.0010**

The relative error, bias, variance and RMSE of the three classifiers are presented in Figure 5.7. Each point on the graph represents one of the 60 data sets. The

values of the x-axis are the outcome for AODE^{SR} divided by that for AODE, and on the y-axis are the outcome for AODE^{BSE} divided by that for AODE. Points on the left of the vertical line at AODE^{SR}/ AODE = 1 in each subgraph are those for which AODE^{SR} has lower metrics than AODE. Points below the horizontal line at AODE^{BSE}/ AODE = 1 are those for which AODE^{BSE} has better results than AODE. Points above the line $X = Y$ indicate that AODE^{SR} has lower values than those of AODE^{BSE}.

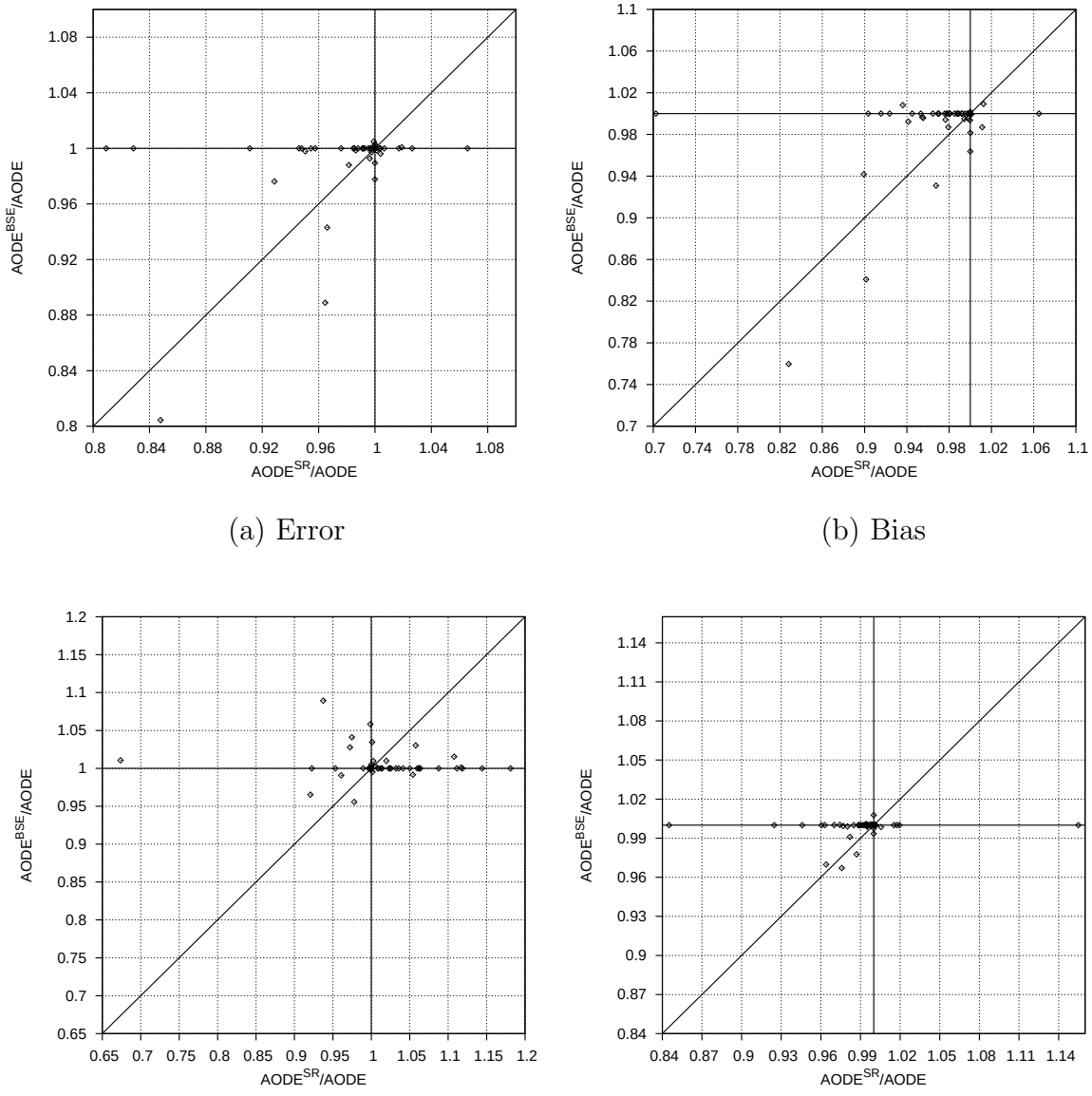
Error

The win/draw/loss records indicate that SR and BSE significantly reduce error of AODE. AODE^{SR} frequently obtains lower error than AODE^{BSE}, but this difference is not significant. The error graph in Figure 5.7 shows that the majority of points are on the left of the line at AODE^{SR}/ AODE = 1. As AODE^{BSE} has identical error to AODE on 41 data sets, most points are on the horizontal line at AODE^{BSE}/ AODE = 1.

The error ratios of AODE^{SR} and AODE^{BSE} over AODE on King-rook-vs-king-pawn (the point at the bottom of the graph) are 0.8478 and 0.8044 respectively. The error ratio of AODE^{BSE} over AODE^{SR} is 0.9489. That is, both AODE^{SR} and AODE^{BSE} reduce the error of AODE considerably, while AODE^{BSE} has substantial lower error than AODE^{SR} on this data set. Similar results can be observed when NB, NB^{BSE} and NB^{SR} are compared. It is possible that BSE can rectify a wider range of interdependencies than SR, which can only detect a special dependency relationship. Hence, BSE might have an advantage on King-rook-vs-king-pawn, in which there are strong dependencies between the presence and position of pieces on the board. However, it appears that BSE does not scale well to the data sets with many attributes, such as Audiology which has 69 attributes. The error of AODE^{SR} and AODE^{BSE} on Audiology are 0.3126 and 0.3863 respectively. That of NB^{SR} and NB^{BSE} are 0.3204 and 0.3888 respectively.

Bias

AODE^{SR} has a significant bias advantage over AODE and AODE^{BSE}. The advantage of AODE^{BSE} compared to AODE is significant. The majority of points in the bias graph are on the left of the line at AODE^{SR}/ AODE = 1 and above the line $X = Y$.



(a) Error (b) Bias

(c) Variance (d) RMSE

Figure 5.7: Comparison of error, bias, variance and RMSE for AODE, AODE^{SR} and AODE^{BSE}.

Variance

AODE enjoys a significant advantage in variance compared to AODE^{SR} and a marginal advantage compared to AODE^{BSE}. The advantage of AODE^{BSE} over AODE^{SR} is marginal. In the variance graph, most points are on the right of the line at AODE^{SR}/ AODE = 1.

RMSE

SR significantly reduces AODE's RMSE. It also has a significant advantage over AODE^{BSE}, which shares a similar level of RMSE with AODE. Most points on the RMSE graph are to the left of the line at AODE^{SR}/ AODE = 1, above the line $X = Y$ and on the horizontal line at AODE^{BSE}/ AODE = 1. On Mushroom (the point on the right of the graph), the RMSE of AODE, AODE^{SR} and AODE^{BSE} are 0.0226, 0.0261 and 0.0226 respectively.

5.6.4 Compute Time Results

Empirical running time is presented here to provide a adjunct to computational complexity analysis in Section 5.5. The pairwise win/draw/loss records on training time are presented in Table 5.12 and that on classification time are presented in Table 5.13. The mean training and classification time across 60 data sets are presented in Figures 5.8 and 5.9 respectively.

Table 5.12: Win/Draw/Loss records on training time.

	Training time				**Training time**		
W/L/D	NB	NB^{SR}	AODE^{BSE}	W/L/D	AODE	AODE^{SR}	AODE^{BSE}
NB				AODE			
NB^{SR}	**4/1/55**			AODE^{SR}	26/1/33		
NB^{BSE}	**0/0/60**	**0/0/60**		AODE^{BSE}	**0/0/60**	**0/0/60**	

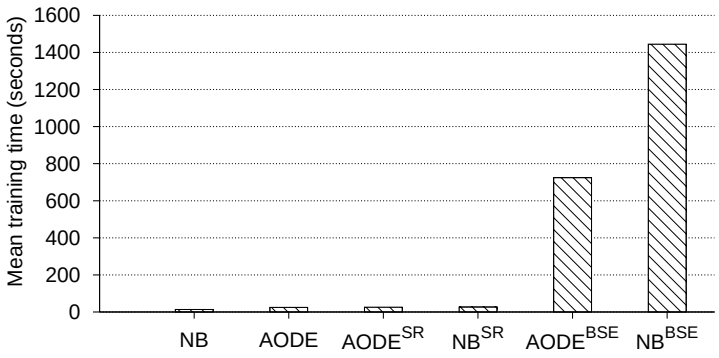


Figure 5.8: Mean training time across 60 data sets.

5.6.4.1 Training Time

The disadvantage of NB^{BSE} in training time relative to NB and NB^{SR} is evident (in no cases, NB^{BSE} has less training time than NB and NB^{SR}). Similar results are observed when AODE^{BSE} is compared to AODE and AODE^{SR}. When SR is applied to NB, it has a significant training time disadvantage relative to NB. In the context of AODE, it shares a similar level of training time with AODE.

NB has the lowest mean training time (13.43 seconds). The differences between AODE, AODE^{SR} and NB^{SR} are small (24.73, 26.13 and 26.39 seconds respectively). As discussed in Section 5.6.2, we present the results of NB^{BSE} without a binomial sign test as it outperforms NB^{BSE} with a binomial sign test. AODE^{BSE} employs a binomial sign test to assess whether an accuracy improvement is significant. Algorithms with a binomial test usually have lower training time than those without, as if the selection of the attribute whose elimination or addition most improves leave-one-out accuracy does not pass the binomial test, the algorithms with the sign test terminate the attribute selection, resulting in less iterations and hence lower training time. For example, an attribute deletion corrects 8 misclassifications and misclassifies 2 previously correct examples. The outcome of one-tailed binomial sign test is $0.0547 > 0.05$. Algorithms with the sign test terminate the attribute selection because this improvement is assessed as insignificant, while those without perform attribute selection continually as long as there is an accuracy improvement. This may explain why NB^{BSE} has the highest mean training time (1444.63 seconds) and has higher training time than AODE^{BSE} on many data sets.

Table 5.13: Win/Draw/Loss records on classification time.

	Classification time				**Classification time**		
W/L/D	NB	NB^{SR}	NB^{BSE}	W/L/D	AODE	$AODE^{SR}$	$AODE^{BSE}$
NB				AODE			
NB^{SR}	**4/1/55**			$AODE^{SR}$	30/2/28		
NB^{BSE}	**35/4/21**	**55/1/4**		$AODE^{BSE}$	33/2/25	28/1/31	

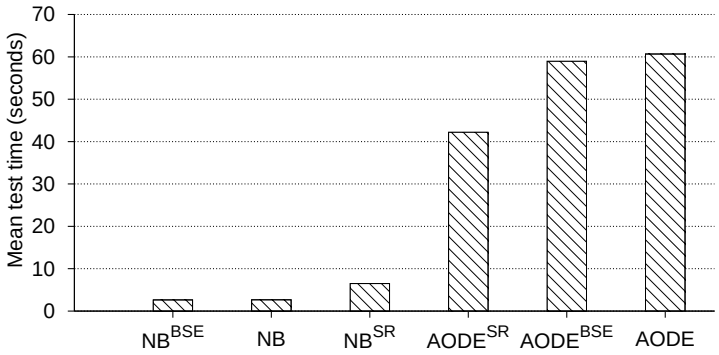


Figure 5.9: Mean classification time across 60 data sets.

5.6.4.2 Classification Time

Compared to NB^{SR}, NB and NB^{BSE} enjoy a considerable advantage in classification time. The advantage of NB^{BSE} over NB is also significant. No statistically significant classification time difference is identified between AODE, $AODE^{SR}$ and $AODE^{BSE}$. Since $AODE^{SR}$ has an additional step to detect the generalization relationship, it is expected to have higher classification time than AODE. However, due to the large number of attribute values deleted (refer to elimination ratios in Section 5.6.5), which results in substantial speed-up, it has lower classification time on 30 data sets compared to AODE.

NB^{BSE} has the lowest mean classification time (2.62 seconds). NB and NB^{SR} obtain the second and third lowest mean classification time (2.64 and 6.5 seconds).

As AODE^{SR} deletes many of attributes on some data sets, such as Connect-4 Opening, it has lower mean classification time than that of AODE^{BSE} and AODE.

5.6.5 Elimination Ratio

To observe the number of attributes that are deleted by SR and BSE, we calculate the elimination ratio of SR and BSE respectively.

5.6.5.1 Elimination Ratio of SR

Figures 5.10 shows average attribute elimination ratios for SR (denoted as r^{SR}) on each data set. This value is obtained by dividing the number of attributes deleted by the number of attributes across all the test examples and iterations:

$$r^{SR} = \frac{\sum_{i=1}^{u} \sum_{t=1}^{|T|} d_t^i}{|T|nu},$$

where u is the number of iterations (it is 50 in our experiment), d_t^i is the number of attributes deleted for the tth instance in the ith iteration.

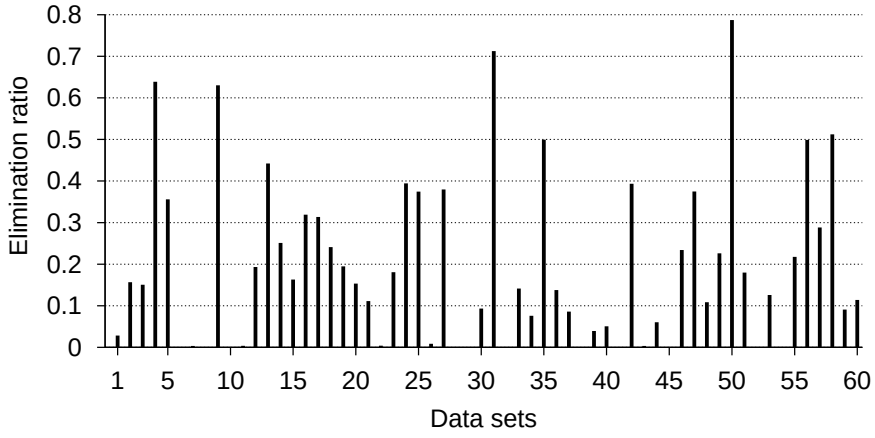


Figure 5.10: Average attribute elimination ratio of SR (The data sets are in the number sequence of Table 3.4)

An elimination ratio of zero represents no deletions. The larger the elimination ratio, the more attributes that are deleted. The data sets in Figures 5.10 are in

the number sequence of Table 3.4. Since the attributes deleted do not change from classification algorithm to algorithm, NB^{SR} and AODE^{SR} have identical elimination ratios on the same data set. As illustrated in Figure 5.10, elimination occurs on 49 out of 60 data sets, which suggests that the specialization–generalization relation is common in real world data. For almost 10% data sets, over 50% of attribute values are eliminated. For more than 50% of data sets, more than 10% of attribute values are eliminated.

The elimination ratios on 5 data sets (Audiology, Connect-4 Opening, Liver Disorders, Sonar Classification and Waveform-5000) are greater than 0.5. NB^{SR} has lower error and RMSE than NB on Audiology and Connect-4 Opening, and identical error and RMSE to NB on Liver Disorders and Waveform-5000. It has higher error but lower RMSE than NB on Sonar Classification. Compared to AODE, AODE^{SR} has lower error and RMSE on Audiology, Connect-4 Opening and Waveform-5000. It has identical error and RMSE to AODE on Liver Disorders. On Sonar Classification, it has higher error but lower RMSE.

Due to the large number of attribute values deleted, which results in substantial speed-up, AODE^{SR} has lower classification time on all the 5 data sets and another 25 data sets compared to AODE. For example, the elimination ratio on Connect-4 Opening is 0.6302 (26 attributes out of 42 attributes are deleted). The classification time of AODE on Connect-4 Opening is 1418.59 seconds, while that of AODE^{SR} is 379.16 seconds.

5.6.5.2 Elimination Ratio of BSE

Figures 5.11 shows average attribute elimination ratios for BSE (denoted as r^{BSE}) on each data set obtained by dividing the number of attributes deleted by the number of attributes across all iterations:

$$r^{BSE} = \frac{\sum_{i=1}^{u} d^i}{nu},$$

where d^i is the number of attributes deleted in the ith iteration.

BSE considers specific classification algorithms in the process of deletion, which may result in different attribute subsets for different algorithms. All elimination ratios

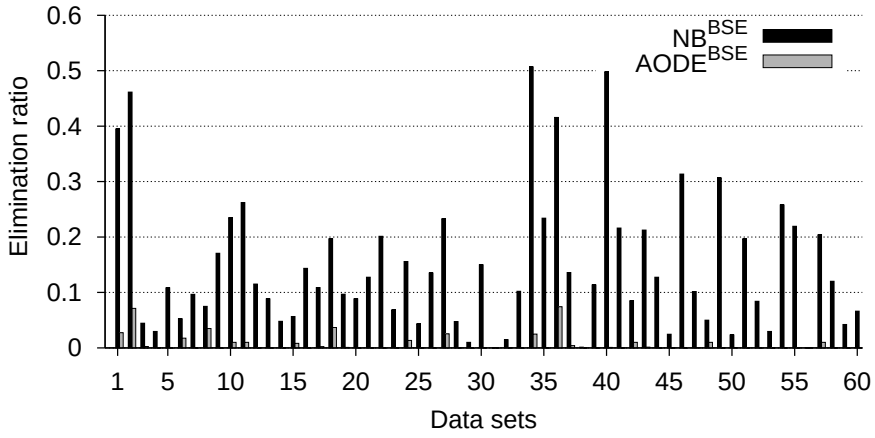


Figure 5.11: Average attribute elimination ratio of NB^{BSE} and AODE^{BSE} (The data sets are in the number sequence of Table 3.4)

of NB^{BSE} and AODE^{BSE} are less than 0.55. For NB^{BSE}, elimination occurs on all data sets except Liver Disorders and Volcanoes. The elimination ratios of NB^{BSE} on three data sets (Adult, Nettalk and Page Blocks) are lager than 0.4. It reduces NB's error and RMSE substantially on these three data sets. More than 10% of attributes are eliminated on 50% data sets. As AODE^{BSE} employs a binomial sign test, the elimination ratios for all data sets are less than 0.1. On 23 data sets, the elimination ratios of AODE^{BSE} are larger than zero. NB^{BSE} has lager elimination ratios than AODE^{BSE} on all data sets except Liver Disorders and Volcanoes, on which the elimination ratio of NB^{BSE} and AODE^{BSE} are zero.

5.6.6 Lazy and Eager Elimination

Eager learning algorithms perform learning at training time, while lazy learning algorithms delay learning until classification time. BSE deletes attributes whose elimination can improve leave-one-out classification accuracy at training time and classifies a new example using the selected attribute subset. In contrast, SR deletes attributes at classification time depending upon which attribute values are instantiated in the object being classified and classifies a new example using the remaining attributes. The attribute subset selected by SR does not change from classification algorithm to algorithm, while that by BSE is different from algorithm to algorithm.

Lazy learning algorithms usually seek to find a model that is most appropriate to the test example. SR deletes attributes that appear to subsume other attributes for each test example, and hence uses different attribute subsets to classify test examples. This may have a different effect on classification compared to deleting complete attributes. For instance, in the data set German, the 20th attribute is $ForeignWorker$ which has two values, *yes* and *no* and the 3th attribute is $CreditHistory$ with the 5 values $A30$, $A31$, $A32$, $A33$ and $A34$. For all the instances in German, if $CreditHistory$ is $A33$, which is "delay in paying off in the past", then $ForeignWorker$ is *yes*. SR can detect this relationship and delete $ForeignWorker$ when these two values are instantiated in the test example. Both $CreditHistory$ and $ForeignWorker$ are used if $ForeignWorker$=*no*.

When we apply SR to NB but restrict its application to deleting only values of $ForeignWorker$ (indicated as NB^{SR}_{FW}), it has error of 0.2633. The error of NB^{BSE} and NB are 0.2651 and 0.2635 respectively. If we delete $ForeignWorker$ for all test examples (indicated as $\text{NB}_{\_FW}$), it has error of 0.2644. In the context of AODE, the elimination of the complete attribute $ForeignWorker$ (indicated as $\text{AODE}_{\_FW}$) increases error of AODE from 0.2587 to 0.2607. AODE has error of 0.2586 by the addition of SR with restriction of deleting only values of $ForeignWorker$ (indicated as AODE^{SR}_{FW}). AODE^{BSE} has identical error to AODE. Table 5.14 presents these

Table 5.14: Errors on German

Error			
NB	$\text{NB}_{\_FW}$	NB^{SR}_{FW}	NB^{BSE}
0.2635	0.2644	0.2633	0.2651
Error			
AODE	$\text{AODE}_{\_FW}$	AODE^{SR}_{FW}	AODE^{BSE}
0.2587	0.2607	0.2586	0.2587

error results.

In this case, the elimination of the complete attribute $ForeignWorker$ increases the errors of NB and AODE, while the elimination of attribute $ForeignWorker$ when SR detects the generalization relationship between $CreditHistory$ and

$ForeignWorker$ decreases the errors. These observations suggest that lazy elimination may have a different effect from eager elimination. When a value of an attribute is highly related to some values, but not all values, of another attribute, elimination of the complete former attribute may lose information, while elimination of the former attribute-value may have positive effect.

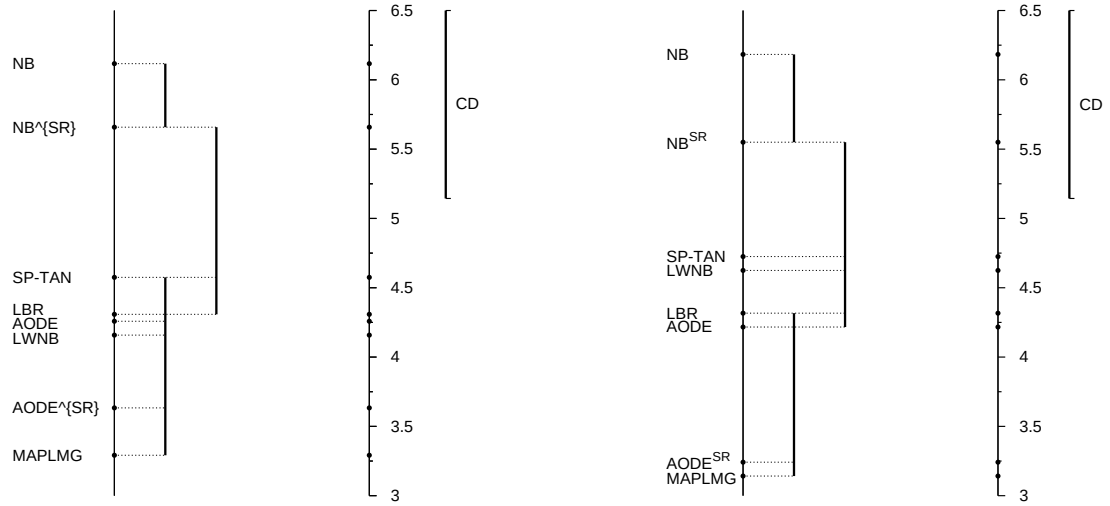
5.6.7 Comparing NB^{SR} and AODE^{SR} with Other Semi-naive Bayesian Classifiers

In this section, we use the Friedman and Nemenyi tests to compare NB^{SR} and AODE^{SR} with NB and the five semi-naive Bayesian methods that significantly reduce NB's error on sixty data sets. As LBR was executed on a different machine, we exclude LBR from the time comparison.

The Friedman statistics for error (10.5755), bias (24.0746), variance (11.6310) and RMSE (12.9193) are greater than the critical value of $F(7, 413)$ for $\alpha = 0.05$ (2.0318). The Friedman statistics for training time (143.8973) and classification time (236.1678) are greater than the critical value of $F(6, 413)$ for $\alpha = 0.05$ (2.1242). Therefore, we reject all the null-hypotheses. The Critical Differences using the Nemenyi test for 8 and 7 algorithms are 1.3555 and 1.1631 respectively ($\alpha = 0.05$). Figure 5.12 presents the outcomes of error and RMSE, Figure 5.13 bias and variance and Figure 5.14 training time and classification time.

5.6.7.1 NB^{SR}

NB^{SR} has lower mean ranks of bias, error and RMSE compared to NB, and higher compared to the remaining algorithms. The error differences between NB^{SR} and NB, SP-TAN and LBR are not statistically significant. MAPLMG, AODE^{SR}, AODE and LWNB have significantly lower mean error ranks than NB^{SR}. Two algorithms, MAPLMG and AODE^{SR} have significant advantage in RMSE over NB^{SR}, which shares a similar level of RMSE with NB, SP-TAN, LWNB, LBR and AODE. NB^{SR} has significantly higher mean bias ranks than all the other algorithms except NB and AODE. It has the third lowest mean variance rank, shares a similar level of variance with NB, AODE, MAPLMG and AODE^{SR}, and outperforms the rest of algorithms.



(a) Error (b) RMSE

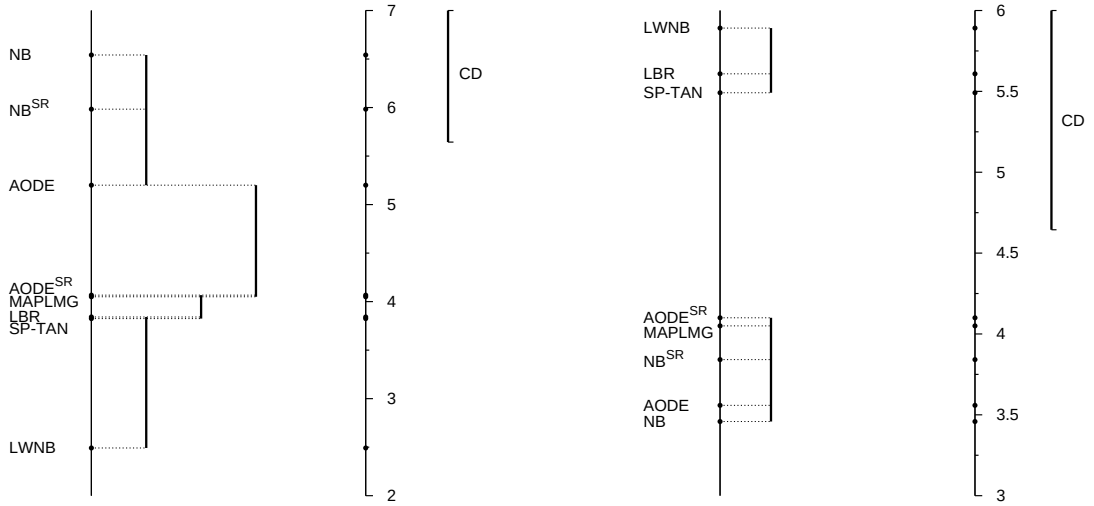
Figure 5.12: Error and RMSE comparison of NB^{SR}, AODE^{SR}, NB, MAPLMG, LBR, AODE, LWNB and SP-TAN with the Nemenyi test on 60 data sets. CD = 1.3555.

NB^{SR} has significant training time disadvantage relative to NB and advantage over SP-TAN and MAPLMG. There is no significant training time differences between NB^{SR} and LWNB, AODE and AODE^{SR}. NB^{SR} shares a similar level of classification time with SP-TAN and has a significantly higher mean classification time rank than NB and significantly lower than AODE, AODE^{SR}, MAPLMG and LWNB.

5.6.7.2 AODE^{SR}

AODE^{SR} achieves the second lowest mean rank for error and RMSE. It has higher mean error rank than MAPLMG, lower than LWNB, AODE, LBR and SP-TAN and significantly lower than NB^{SR} and NB. With respect to RMSE, it has slightly higher mean rank than MAPLMG and a significant advantage over all the remaining algorithms except AODE and LBR.

The bias difference between AODE^{SR} and MAPLMG is very small. They have significant higher mean bias rank than LWNB and lower than NB^{SR} and NB. The bias differences between AODE^{SR}, MAPLMG, SP-TAN and LBR are also small. AODE^{SR} has higher mean variance rank than NB, AODE, NB^{SR} and MAPLMG,



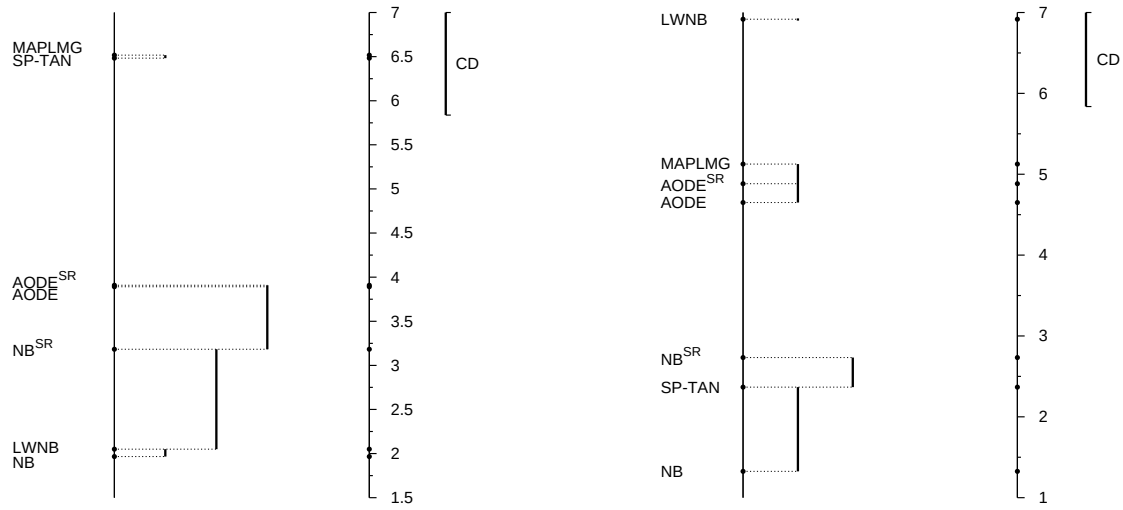
(a) Bias (b) Variance

Figure 5.13: Bias and variance comparison of NB^{SR}, AODE^{SR}, NB, MAPLMG, LBR, AODE, LWNB and SP-TAN with the Nemenyi test on 60 data sets. CD = 1.3555.

but the differences are not significant. The variance advantage of AODE^{SR} over LBR, LWNB and SP-TAN is clear.

Turning to computing time, there is only slight training time difference between AODE^{SR} and AODE. They share a similar level of training time with NB^{SR} and have a clear disadvantage relative to NB and LWNB. The disadvantages in training time of MAPLMG and SP-TAN relative to the rest of algorithms are evident. Four groups of algorithms can be identified when classification time is compared. MAPLMG, AODE^{SR} and AODE have significantly lower mean ranks than LWNB and significantly higher than all the other algorithms.

The win/draw/loss comparisons in Sections 5.6.2 and 5.6.3 showed that SR can significantly reduce the error, bias and RMSE of NB and AODE. However, in this comparison, SR substantially, but not significantly, reduces those of NB and AODE. This is because this comparison involves 8 algorithms and hence the Nemenyi test is not powerful enough to detect significant differences between NB^{SR} and NB and AODE^{SR} and AODE.



(a) Training time (b) Classification time

Figure 5.14: Computing time comparison of NB^{SR}, AODE^{SR}, NB, MAPLMG, AODE, LWNB and SP-TAN with the Nemenyi test on 60 data sets. CD = 1.1631.

5.6.7.3 Discussion

SR is effective at reducing NB's bias, error and RMSE. However, the computational complexity of NB^{SR} is similar to that of AODE, while the improvement in accuracy it produces is not generally as great. In contrast, the bias, error and RMSE of AODE can be substantially reduced by the addition of SR with identical computational complexity to AODE. AODE^{SR} demonstrates competitive error and RMSE with MAPLMG, a powerful variant of AODE. The advantage of AODE^{SR} in computation efficiency relative to MAPLMG is substantial. In addition, AODE^{SR}, but not MAPLMG, inherits AODE's capacity for incremental learning.

Compared to LBR, AODE^{SR} has considerably lower classification time in all cases and to LWNB in cases when the number of attributes is not large. It is also space efficient as it only uses frequency tables formed at training time to perform classification and hence does not need to retain training data. Compared to SP-TAN, it provides considerably faster training. AODE^{SR} has significant RMSE advantage over LWNB and SP-TAN, but not LBR.

AODE^{SR} enjoys competitive classification accuracy, probabilistic prediction and computation efficiency. In consequence, it may prove desirable for many classification tasks.

5.7 Semi-Supervised Subsumption Resolution (SSSR)

Learning algorithms that make use of both labeled and unlabeled data for training are referred to as *semi-supervised learning algorithms*. Since SR only needs to use frequencies of two attribute values to detect whether one is a generalization of another, it can use not only labeled but also unlabeled data to identify the generalization relationship. Compared to labeled data, unlabeled data is relatively easy to collect. For example, unlabeled web pages can be inexpensively collected using web crawlers, while hand labeled web pages are very expensively to obtain. As discussed previously, variance may decrease with increasing training sample size. Consequently, the addition of large amounts of unlabeled data is expected to reduce the variance of SR. We call Subsumption Resolution that uses both labeled and unlabeled data for training *Semi-Supervised Subsumption Resolution* (SSSR). When SSSR is applied to NB (or AODE), the resulting classifier acts as NB^{SR} (or AODE^{SR}) except that unlabeled data is also used to identify and delete generalizations. We denote NB with SSSR as NB^{SSSR} and AODE as AODE^{SSSR}.

To evaluate the effect of SSSR, we compare NB^{SSSR} to NB^{SR} and AODE^{SSSR} to AODE^{SR}. In these experiments, training data is randomly divided into approximately equal-sized training set and test set, and a test set without class information is used as unlabeled data [1]. When training data has quantitative attributes, MDL discretization module in Weka discretizes them using both training set and test set (including class information) for SSSR, while only training set for SR. Cut points generated from training set and those from training set together with test set are usually different. Therefore, the resulting training set for SSSR and that for SR may be different.

[1] In a semi-supervised setting, an algorithm performs learning using labeled and unlabeled data and makes predictions for unseen examples. In a transductive setting, an algorithm predicts the labels of known unlabeled examples. In these experiments, since we use test data as unlabeled data, NB^{SSSR} and AODE^{SSSR} perform transductive learning. However, SR can be naturally used on unseen data and perform semi-supervised learning.

In addition, when training data has missing values, mode for qualitative attributes and mean for quantitative attributes in training set may be different from those in training set together with test set. To avoid this problem, these experiments are run in the same manner as those described in Section 3.3.2 except that training data with quantitative attributes is preprocessed by using MDL discretization and with missing values by replacing them using mode or mean in the whole training data.

Table 5.15 presents the win/draw/loss records for NB^{SSSR} against NB^{SR} on sixty

Table 5.15: Win/Draw/Loss:NB^{SSSR} vs. NB^{SR}

	$\textbf{NB}^{SSSR}$ **vs.** $\textbf{NB}^{SR}$	
	W/D/L	p
Error	10/24/26	**0.0057**
Bias	8/24/28	**0.0006**
Variance	21/28/11	0.0551
RMSE	11/24/25	**0.0144**

Table 5.16: Win/Draw/Loss: AODE^{SSSR} vs. AODE^{SR}

	$\textbf{AODE}^{SSSR}$ **vs.** $\textbf{AODE}^{SR}$	
	W/D/L	p
Error	12/24/24	**0.0326**
Bias	12/25/23	**0.0448**
Variance	22/26/12	0.0607
RMSE	11/23/26	**0.0100**

data sets and Table 5.16 for AODE^{SSSR} against AODE^{SR}. The addition of unlabeled data substantially, but not significantly, reduces the variance of SR. However, the increase in bias outweighs the reduction in variance and results in an overall increase in error and RMSE. While the use of unlabeled data to improve SR appears in principle to be a promising idea, further development is required.

5.8 Summary

BSE uses a wrapper approach to identify and remove harmful interdependencies and has been applied profitably in domains with highly correlated attributes. However, it imposes high training time overheads on learning algorithms, especially those with high classification time complexity. Based on a theoretical analysis of attribute-value interdependencies, we have proposed a novel technique, SR, to efficiently detect the specialization–generalization relationship, a special form of interdependency, and delete generalizations at classification time. Unlike other lazy methods, it is time and space efficient.

We investigate the effect of BSE and SR on classification accuracy and probabilistic prediction by applying them to NB and AODE. Extensive experimental results (win/draw/loss records) show that both significantly improve upon NB's accuracy and probability estimates and AODE's accuracy. However, the probability estimates of AODE can be significantly enhanced by the addition of SR, but not BSE. The advantage of SR in probabilistic prediction and computation efficiency over BSE is significant in the context of AODE. In addition, SR inherits NB and AODE's capacity for incremental learning and is suited to semi-supervised learning. Nonetheless, it is only applicable to algorithms without model selection, such as NB and AODE. BSE is more widely applicable, but does not support incremental and semi-supervised learning and has high computational overheads. We believe that the appropriate conclusion to draw from our results is that SR is effective at reducing error, rather than that it is necessarily superior to the BSE strategy in this respect in the AODE context.

We use the Friedman and Nemenyi tests to compare NB^{SR} and AODE^{SR} with five state-of-the-art semi-naive Bayesian methods. The results show that AODE^{SR} competes favorably with MAPLMG, LBR, AODE, LWNB and SP-TAN on our data collection. It achieves competitive classification accuracy without incurring the high training time overheads of MAPLMG and SP-TAN and high classification time overheads of LWNB and LBR. Further, it enjoys a considerable advantage in probabilistic estimation over LWNB and SP-TAN. We believe AODE^{SR} provides a reasonable trade-off between classification accuracy and computational efficiency and appears to be a promising approach for a wide range of classification problems.

Chapter 6

Near-Subsumption Resolution

The previous chapter has addressed the specialization-generalization relationship and the Subsumption Resolution technique to resolve this type of interdependence problem. In practice, errors are common in data, especially in large data. It is possible that noisy or erroneous data might prevent detection of a specialization–generalization relationship. Also, in many cases an attribute value cannot be classified with full certainty, but can with strong possibility, as the generalization of another. Further, when we assume a specialization–generalization relationship exists in the sample, it is likely that the relationship is actually a near specialization–generalization relationship in the population from which the sample is drawn. In consequence, it is useful to investigate the effect of elimination of near-generalizations.

In this chapter, we first introduce the near specialization-generalization relationship, which relaxes the perfect relationship by allowing some uncertainty. Next, we extend Subsumption Resolution to Near-Subsumption Resolution and apply it to NB and AODE. An extensive empirical study is performed to investigate the effect of different lower bounds on the strength of allowed near-generalizations. Finally, we examine the reasons elimination of near-generalizations often proves profitable based on three exemplar data sets.

6.1 The Near Specialization-generalization Relationship

For two attribute values x_i and x_j, if $P(x_j \mid x_i) = 1.0$, x_j is classified as the generalization of x_i. We extend the definition to a near-generalization by relaxing the condition from equal to approximately equal:

Definition 4. *(Near-Generalization and Near-Specialization) For two attribute values x_i and x_j, if $P(x_j \mid x_i) \approx 1.0$ then x_j is a* near-generalization *of x_i and x_i is a* near-specialization *of x_j.*

There are many real world examples in which one attribute value approximately subsumes another. For instance, given attribute $Short{-}term{-}visitor$ with two values of *yes* and *no* and attribute $Local{-}number{-}registered$ with two values of *yes* and *no*. It is logical to infer that most short term visitors do not register a local phone number under their names, while most local residents register their own phone numbers. However, some short term visitors register a local phone number and some local residents do not. Hence, $Local{-}number{-}registered{=}no$ is a near-generalization of $Short{-}term{-}visitor{=}yes$ and $Short{-}term{-}visitor{=}no$ is a near-generalization of $Local{-}number{-}registered{=}yes$.

6.2 Near-Subsumption Resolution (NSR)

SR can be simply extended to manipulate the near specialization–generalization relationship by relaxing the criterion to accept generalizations. This extension, called Near-Subsumption Resolution (NSR), identifies at classification time pairs of attribute values such that one appears to approximately subsume (be a near-generalization of) the other. It then deletes the near-generalization.

6.2.1 Lower Bounds

A lower bound is required on when to accept a near specialization-generalization relationship. We use $P(x_j \mid x_i)$ to estimate how approximately x_i and x_j have the specialization–generalization relationship. Let r be a user-specified lower bound value, $0 \leq r \leq 1.0$. If $1.0 > P(x_j \mid x_i) \geq r \approx 1.0$, x_j is a near-generalization of x_i.

For all Z, given $P(x_j \mid x_i) \approx 1.0$, we have

$$P(x_i, x_j, Z) \approx P(x_i, Z),$$

and hence

$$P(y \mid x_1, \ldots, x_n) \approx P(y \mid x_1, \ldots, x_{j-1}, x_{j+1}, \ldots x_n).$$

If an appropriate r is selected, removing x_j from a Bayesian classifier might positively affect a Bayesian classifier. However, there does not appear to be any satisfactory a priori method to select an appropriate value for r. The classification performance is sensitive to the selection of lower bounds to accept the near specialization-generalization relationship. Deleting weak near-generalizations might prove effective on some data sets, while only eliminating strong near-generalizations may prove more desirable on other data sets.

6.2.2 Criterion for Identifying Near-Generalizations

NSR uses the following criterion

$$|T_{x_i,x_j}| \geq l \wedge |T_{x_i}| > |T_{x_i,x_j}| \geq r|T_{x_i}|$$

to infer that x_j is a near-generalization of x_i. We use SR's default $l = 30$ as the minimum frequency at which to accept the near specialization-generalization relationship. Figure 6.1 illustrates the criterion from the perspective of set theory. If the number of elements in intersection of T_{x_i} and T_{x_j} is greater than l and than r multiplies the number of elements in T_{x_i}, but less than the number of elements in T_{x_i}, x_j is a near-generalization of x_i.

6.3 NB and AODE with NSR

Compared to SR, NSR only changes the criterion to accept the relationship and hence does not incur any additional computational complexity. We denote NB with Near-Subsumption Resolution as NB_r^{SR} and AODE with Near-Subsumption Resolution as AODE_r^{SR}, where r is the critical r-value used to accept or reject a candidate near-generalization.

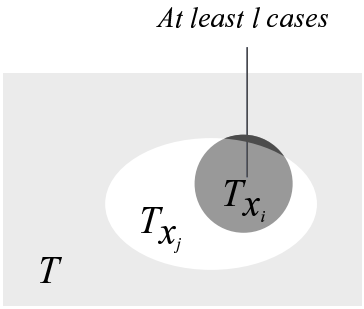


Figure 6.1: Criterion to infer that x_j is a near-generalization of x_i. T_{x_i} and T_{x_j} are the sets of the training cases with value x_i and x_j respectively. In the current work $l = 30$.

6.3.1 NB with NSR

In the context of NB, NSR deletes near-generalization attribute values if a near-specialization is detected. It also deletes generalization attribute values. Independence is assumed among the resulting attribute-values given the class.

Classification of instance $\mathbf{x} = \langle x_1, \ldots, x_n \rangle$ consists of two steps:

1. Set R to $\{x_{1 \leq i \leq n} \mid \neg \exists x_{1 \leq j \leq n}\, x_i \neq x_j \wedge P(x_i \mid x_j) \geq r\}$.

2. Estimate $P(y, \mathbf{x})$ by

$$\hat{P}(y, \mathbf{x}) = \hat{P}(y) \prod_{x_i \in R} \hat{P}(x_i \mid y).$$

6.3.2 AODE with NSR

When NSR is applied to AODE, the resulting classifier acts as AODE^{SR} except that it deletes near-generalization attribute-values from use in any role in the classifier if a near-specialization is detected, and aggregates the predictions of all qualified classifiers using the remaining attribute-values.

Classification consists of two steps:

1. Set R to $\{x_{1 \leq i \leq n} \mid \neg \exists x_{1 \leq j \leq n}\, x_i \neq x_j \wedge P(x_i \mid x_j) \geq r\}$.

2. Estimate $P(y, \mathbf{x})$ by

$$\hat{P}(y, \mathbf{x}) = \sum_{x_i \in R \wedge F(x_i) \geq m} \hat{P}(y, x_i) \prod_{x_j \in R} \hat{P}(x_j \mid y, x_i).$$

6.4 Effect of Different Lower Bounds: An Empirical Study

The key issue for the near specialization–generalization relationship is how to select an appropriate lower bound at which to accept the relationship. An empirical study is performed to investigate whether an appropriate value of r can be found. As initial exploratory investigations found NB_r^{SR} using values smaller than 0.990 frequently obtains higher error compared to NB^{SR} and the difference between AODE_r^{SR} using values smaller than 0.990 and AODE^{SR} is not statistically significant, we present the error and RMSE results in the range of $r = 0.999$ to $r = 0.990$ with a decrement of 0.001.

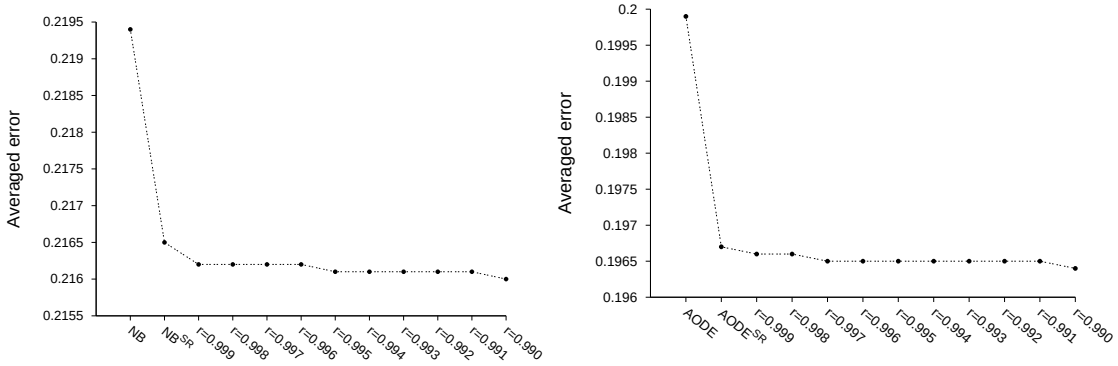
6.4.1 Mean Error and RMSE

The averaged error and RMSE of NB_r^{SR} and AODE_r^{SR} across 60 data sets as a function of r are presented in Figures 6.2 and 6.3. The values on the x-axis are the lower bounds to accept a candidate near-generalization and the values on the y-axis are the averaged error and RMSE across 60 data sets. The error and RMSE of NB and NB^{SR} are included in Figure 6.2 and that of AODE and AODE^{SR} are included in Figure 6.3.

NB_r^{SR} has lower mean error and RMSE at all settings of r compared to NB and NB^{SR}. Similarly, at all settings of r, AODE_r^{SR} has lower mean error and RMSE compared to AODE and AODE^{SR}.

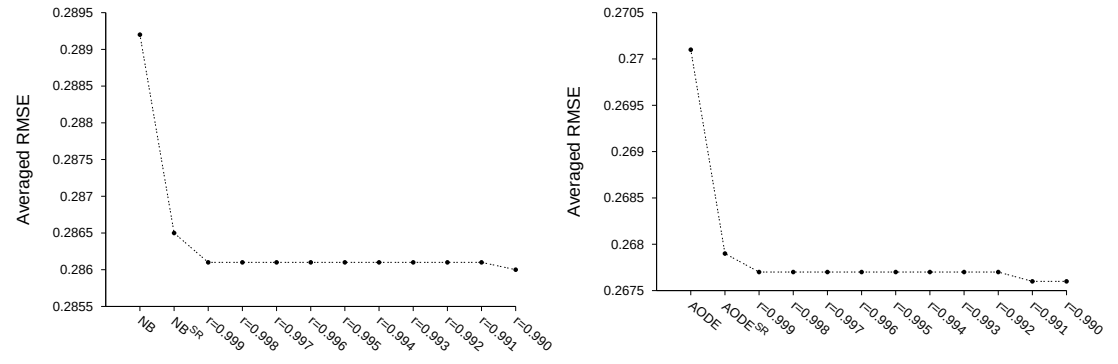
6.4.2 NB_r^{SR}

Table 6.1 presents the win/draw/loss records of error for NB and NB^{SR} against NB_r^{SR}. The p value is the outcome of a one-tailed binomial sign test. Table 6.2 presents the win/draw/loss records of RMSE for NB and NB^{SR} against NB_r^{SR}.



(a) NB, NB^{SR} and NB_r^{SR} with r (b) AODE, AODE^{SR} and AODE_r^{SR} with r

Figure 6.2: Averaged error across 60 data sets, as function of r.



(a) NB, NB^{SR} and NB_r^{SR} with r (b) AODE, AODE^{SR} and AODE_r^{SR} with r

Figure 6.3: Averaged RMSE across 60 data sets, as function of r.

Error

For every setting of r, $0.990 \leq r \leq 0.999$, NB_r^{SR} enjoys a significant error advantage over NB. The error differences between NB_r^{SR} and NB^{SR} are small.

RMSE

The RMSE advantage of NB_r^{SR}, at all settings of r, compared to NB is clear. It shares a similar level of RMSE with NB^{SR}.

6.4.3 AODE_r^{SR}

Tables 6.3 and 6.4 present the win/draw/loss records of error and RMSE for AODE and AODE^{SR} against AODE_r^{SR}.

Error

At all settings of r, the error advantage of AODE_r^{SR} relative to AODE is statistically significant. It enjoys a significant error advantage compare to AODE^{SR} at all settings of r except $r = 0.999$, $r = 0.998$ and $r = 0.992$. The advantage of $\text{AODE}_{0.992}^{SR}$ compared to AODE^{SR} is marginal ($p = 0.0669$).

RMSE

AODE_r^{SR} significantly reduces AODE's RMSE for every setting of r. It also significantly reduces the RMSE of AODE^{SR} at all settings of r except $r = 0.999$ and $r = 0.998$. The advantage of AODE^{SR} compared to $\text{AODE}_{0.998}^{SR}$ is marginal ($p = 0.0898$).

Table 6.1: Win/Draw/Loss comparison of error: NB and NB^{SR} vs NB$_r^{SR}$

	NB$_r^{SR}$									
W/D/L	r=0.999	r=0.998	r=0.997	r=0.996	r=0.995	r=0.994	r=0.993	r=0.992	r=0.991	r=0.990
NB	**10/19/31**	**10/20/30**	**9/21/30**	**11/19/30**	**11/18/31**	**11/18/31**	**12/19/29**	**14/19/27**	**14/19/27**	**14/19/27**
p	**0.0007**	**0.0011**	**0.0005**	**0.0022**	**0.0014**	**0.0014**	**0.0058**	**0.0298**	**0.0298**	**0.0298**
NB^{SR}	3/53/4	4/52/4	5/49/6	7/46/7	8/42/10	9/41/10	12/38/10	12/39/9	11/39/10	13/36/11
p	0.5000	0.6367	0.5000	0.6047	0.4073	0.5000	0.4159	0.3318	0.5000	0.4194

Table 6.2: Win/Draw/Loss comparison of RMSE: NB and NB^{SR} vs NB$_r^{SR}$

	NB$_r^{SR}$									
W/D/L	r=0.999	r=0.998	r=0.997	r=0.996	r=0.995	r=0.994	r=0.993	r=0.992	r=0.991	r=0.990
NB	**7/23/30**	**7/23/30**	**7/23/30**	**8/21/31**	**8/21/31**	**9/20/31**	**10/20/30**	**10/20/30**	**10/20/30**	**10/19/31**
p	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0003**	**0.0011**	**0.0011**	**0.0011**	**0.0007**
NB^{SR}	3/55/2	3/53/4	5/49/6	5/47/8	4/48/8	6/43/11	8/41/11	9/39/12	9/37/14	9/35/16
p	0.5000	0.5000	0.5000	0.2905	0.1938	0.1662	0.3238	0.3318	0.2024	0.1148

Table 6.3: Win/Draw/Loss comparison of error: AODE and AODE^{SR} vs AODE$_r^{SR}$

	AODE$_r^{SR}$									
W/D/L	r=0.999	r=0.998	r=0.997	r=0.996	r=0.995	r=0.994	r=0.993	r=0.992	r=0.991	r=0.990
AODE	**11/18/31**	**11/17/32**	**11/17/32**	**12/17/31**	**12/18/30**	**12/18/30**	**12/18/30**	**12/18/30**	**11/19/30**	**11/19/30**
p	**0.0014**	**0.0010**	**0.0010**	**0.0027**	**0.0040**	**0.0040**	**0.0040**	**0.0040**	**0.0022**	**0.0022**
AODE^{SR}	2/55/3	2/52/6	**2/48/10**	**4/44/12**	**5/41/14**	**5/41/14**	**6/39/15**	7/38/15	**8/33/19**	**7/32/21**
p	0.5000	0.1445	**0.0193**	**0.0384**	**0.0318**	**0.0318**	**0.0392**	0.0669	**0.0261**	**0.0063**

Table 6.4: Win/Draw/Loss comparison of RMSE: AODE and AODE^{SR} vs AODE$_r^{SR}$

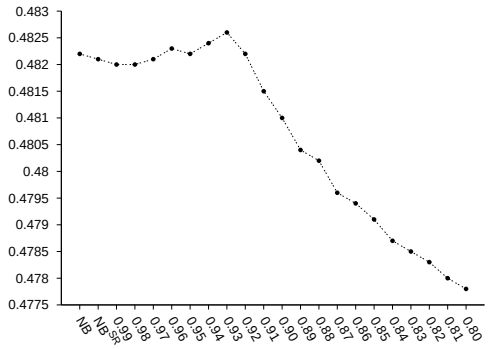
	AODE$_r^{SR}$									
W/D/L	r=0.999	r=0.998	r=0.997	r=0.996	r=0.995	r=0.994	r=0.993	r=0.992	r=0.991	r=0.990
AODE	**10/15/35**	**10/15/35**	**10/15/35**	**10/14/36**	**10/14/36**	**10/14/36**	**10/14/36**	**10/14/36**	**10/14/36**	**10/14/36**
p	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**
AODE^{SR}	1/56/3	2/51/7	**2/48/10**	**2/46/12**	**2/41/17**	**2/41/17**	**4/38/18**	**2/39/19**	**3/37/20**	**3/35/22**
p	0.3125	0.0898	**0.0193**	**0.0065**	**0.0004**	**0.0004**	**0.0022**	**0.0001**	**0.0002**	**0.0001**

6.4.4 Learning Curve

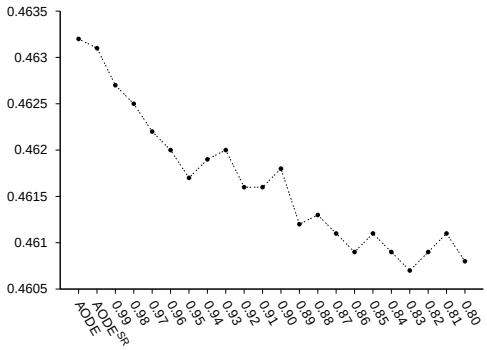
To examine the behaviors of NB_r^{SR} and AODE_r^{SR} with different lower bounds, we generate learning curves in which each point represents the error of NB_r^{SR} and AODE_r^{SR} corresponding to each r on the x-axis. As there are only small differences within intervals of less than 0.01, we present the results in the range of $r = 0.99$ to $r = 0.80$ with a decrement of 0.01. The two decimal numbers on the x-axis are the lower bounds of the near specialization–generalization relationship. The error of NB and NB^{SR} are also included in each graph for NB_r^{SR} and that of AODE and AODE^{SR} are included in each graph for AODE_r^{SR}.

Five main patterns are observed in our experiments when the error of NB with NSR and AODE with NSR are considered:

- Error reduces continuously with slight fluctuations. Examples include Abalone (Figures 6.4), Connect-4 Opening, Letter Recognition, Syncon and Waveform-5000 for both NB_r^{SR} and AODE_r^{SR}. On some data sets, such as Pen Digits, NB_r^{SR} has a largely downwards trend in error with decreasing values of r, while AODE_r^{SR} does not have error reduction. Similarly, on some other data sets, such as Auto Imports and Vowel, the error of NB_r^{SR} first decreases and then increases after $r = 0.88$, while that of AODE_r^{SR} continuously decreases with slight fluctuations.



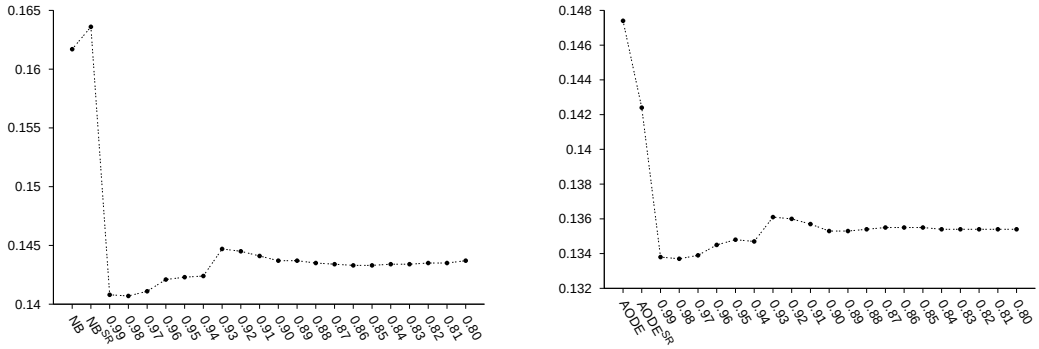
(a) NB, NB^{SR} and NB_r^{SR}



(b) AODE, AODE^{SR} and AODE_r^{SR}

Figure 6.4: Error on Abalone.

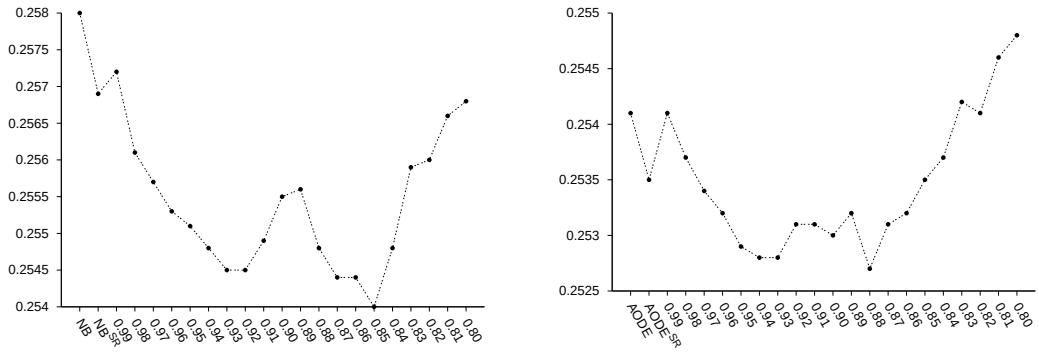
- Error reduces first and then increases, such as the pattern on Adult (Figure 6.5), Annealing, Audiology, Credit Screening, Horse Colic, King-rook-vs-king-pawn, Segment, Sign and Vehicle for both NB_r^{SR} and AODE_r^{SR}.



(a) NB, NB^{SR} and NB_r^{SR} (b) AODE, AODE^{SR} and AODE_r^{SR}

Figure 6.5: Error on Adult.

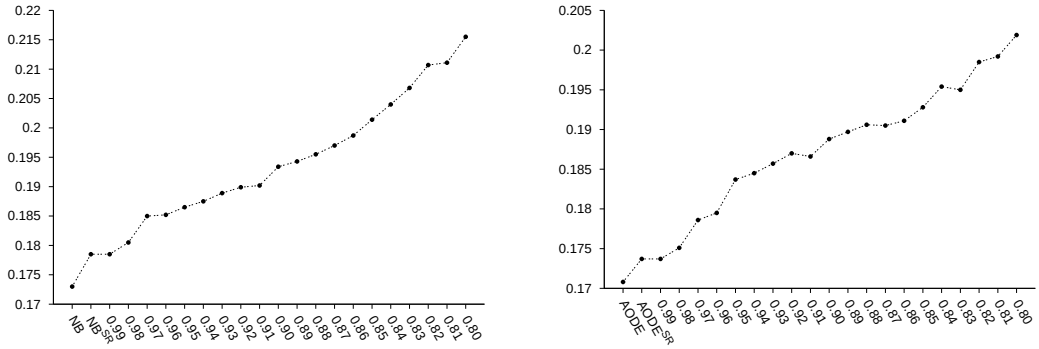
- Error increases initially and then reduces. It increases again when small r is used. Example includes Pima Indians Diabetes (Figure 6.6) for both NB_r^{SR} and AODE_r^{SR}.



(a) NB, NB^{SR} and NB_r^{SR} (b) AODE, AODE^{SR} and AODE_r^{SR}

Figure 6.6: Error on Pima Indians Diabetes.

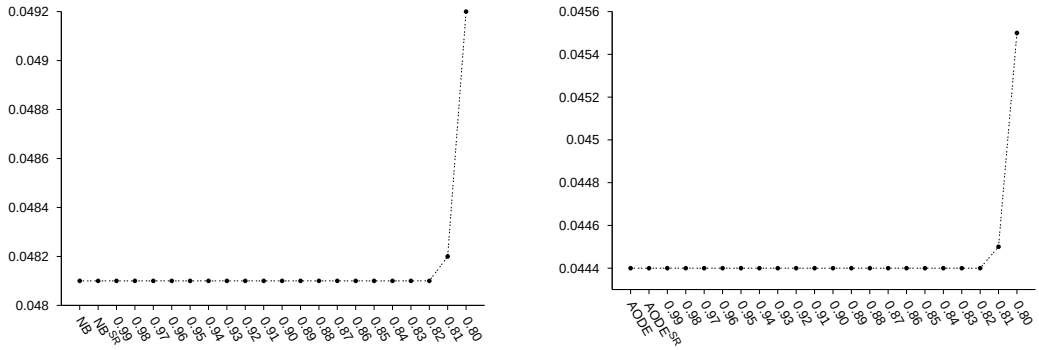
- Error increases continuously with slight fluctuations, such as the pattern on Hepatitis (Figures 6.7), House Votes 84, LED, Primary Tumor and Wine Recognition for NB_r^{SR} and AODE_r^{SR}.



(a) NB, NB^{SR} and NB_r^{SR} (b) AODE, AODE^{SR} and AODE_r^{SR}

Figure 6.7: Error on Hepatitis.

- Error does not change for most values of r, such as the pattern on Splice-junction Gene Sequences (Figure 6.8) and Nettalk.

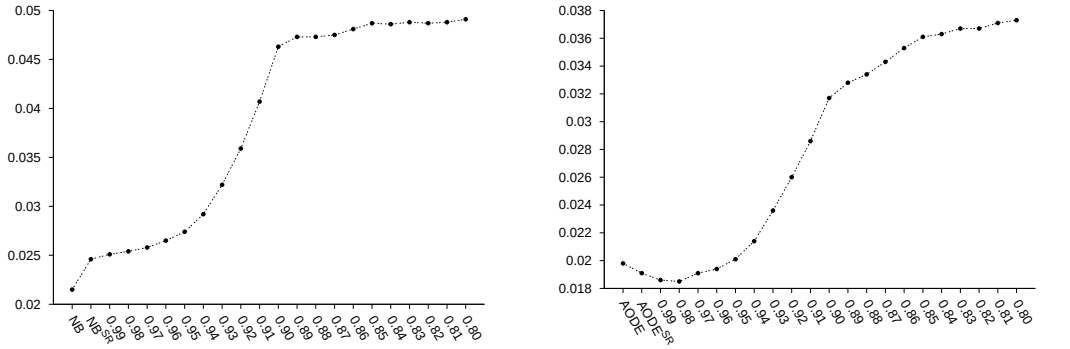


(a) NB, NB^{SR} and NB_r^{SR} (b) AODE, AODE^{SR} and AODE_r^{SR}

Figure 6.8: Error on Splice-junction Gene Sequences.

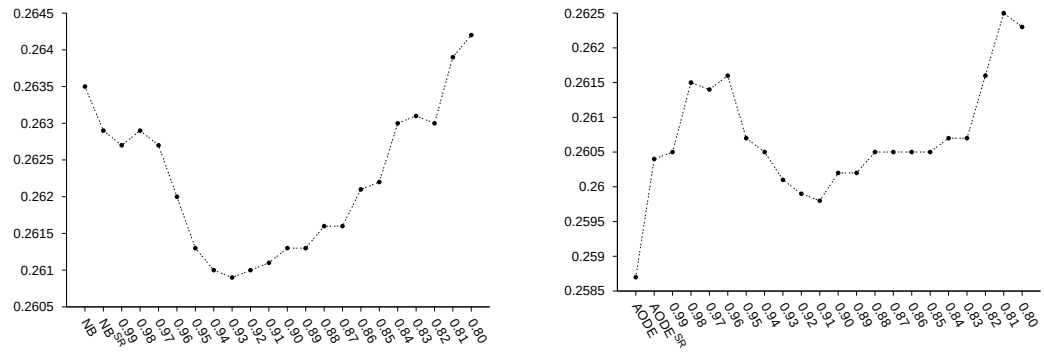
In some cases, NSR has a positive effect on AODE but a negative effect on NB. In other cases, NSR has a negative effect on AODE but a positive effect on NB. For example, on Hypothyroid, NB with NSR has higher error at all settings of r compared

to NB, while AODE with NSR using $0.96 \leq r \leq 1.00$ has lower error compared to AODE (Figure 6.9). NSR reduces the error of NB when $0.82 \leq r \leq 0.99$ on German, while NSR increases the error of AODE at all settings of r (Figure 6.10).



(a) NB, NB^{SR} and NB_r^{SR} (b) AODE, AODE^{SR} and AODE_r^{SR}

Figure 6.9: Error on Hypothyroid(Garavan) .



(a) NB, NB^{SR} and NB_r^{SR} (b) AODE, AODE^{SR} and AODE_r^{SR}

Figure 6.10: Error on German.

We obtain a horizontal learning curve on nine data sets (Car Evaluation, Contact-lenses, Labor negotiations, Liver Disorders, Lung Cancer, Nursery, Promoter Gene Sequences, Tic-Tac-Toe Endgame, Volcanoes). Among these data sets, the specialization–generalization relationship is detected on two data sets (Liver Disorders, Volcanoes).

6.5 Why Do We Delete Near-Generalizations?

The specialization–generalization relation is common in real world data, as shown in Figure 5.10. However, it is difficult to satisfy the exact relationship in many scenarios and hence it is more practical to relax the condition. For example, in a noisy data set in which a single male is indicated as pregnant, we can not infer $P(Gender{=}female \mid Pregnant{=}yes) = 1.0$. A simple solution is to weaken the condition by using a threshold less than $P(x_j \mid x_i) = 1.0$.

From the experimental results in Section 6.4 we can see that AODE_r^{SR} with $0.990 \leq r \leq 0.999$ except $r = 0.999$, $r = 0.998$ and $r = 0.992$ enjoy a significant advantage in error over AODE^{SR}. Nonetheless, NB_r^{SR} shares a similar level of error with NB^{SR} at all settings of r. It is not immediately obvious why those values of r should be desirable in the context of AODE. In this section, we explore the reasons deleting near-generalizations often proves advantageous based on three exemplar data sets (Adult, Abalone and Pima Indians Diabetes). We use $r = 0.990$ as the lower bound. Similar results can be obtained using $0.990 < r \leq 0.999$ on these three data sets.

6.5.1 Adult

The classification task of adult is to predict whether income exceeds fifty thousand US dollars a year. It has 14 attributes (6 continuous and 8 discrete) besides the class label. The attribute $Education{-}num$ (the 5th attribute) recodes the attribute *Education* (the 4th attribute) from a descriptive to a numeric format and hence $Education{-}num$ is a substitution of *Education* (refer to the Definition 2 in Section 5.1). Given *Education*, $Education{-}num$ is redundant. This redundancy can be detected by applying SR to NB and AODE. The errors of AODE with all attributes and all attributes except $Education{-}num$ are 0.1474 and 0.1439 respectively using the experimental method mentioned in Section 3.3.2. However, NB with all attributes has lower error (0.1617) than NB with all attributes except $Education{-}num$ (0.1674). The error of NB^{SR} and AODE^{SR} are 0.1636 and 0.1424 respectively. As has been discussed in Section 5.6.6, NB^{SR} and AODE^{SR} delete attributes depending upon which attribute values are instantiated in the object being classified, hence they may have different results from that of NB and AODE by deleting complete attributes. These

error results are presented in Table 6.5, where $\text{NB}_{\_EN}$ and $\text{AODE}_{\_EN}$ indicate NB and AODE with all attributes except $Education{-}num$ respectively.

Table 6.5: Errors on Adult

Error			
NB	$\text{NB}_{\_EN}$	NB^{SR}	$\text{NB}^{SR}_{0.99}$
0.1617	0.1674	0.1636	0.1408
Error			
AODE	$\text{AODE}_{\_EN}$	AODE^{SR}	$\text{AODE}^{SR}_{0.99}$
0.1474	0.1439	0.1424	0.1338

$\text{NB}_{\_EN}$: NB with all attributes except $Education{-}num$
$\text{AODE}_{\_EN}$: AODE with all attributes except $Education{-}num$

$\text{NB}^{SR}_{0.99}$ improves upon NB^{SR} substantially (the error reduces from 0.1636 to 0.1408). $\text{AODE}^{SR}_{0.99}$ also has a considerable error reduction on AODE^{SR} (from 0.1424 to 0.1338). We investigate the attributes that are deleted by NB^{SR} and $\text{NB}^{SR}_{0.99}$ (NB^{SR} deletes identical attributes to AODE^{SR}, and so does $\text{NB}^{SR}_{0.99}$ to $\text{AODE}^{SR}_{0.99}$). Our experiment reveals that both NB^{SR} and $\text{NB}^{SR}_{0.99}$ delete $Education{-}num$ for those test instances of which the frequencies in the training set of the value of $Education{-}num$ are larger than $l = 30$, and $\text{NB}^{SR}_{0.99}$ also deletes two other types of attributes. The first one is near-generalizations and the second one is attributes with noise.

6.5.1.1 Near-Generalization

Attribute $Marital{-}status$ (the 6th attribute) has 7 values: $married{-}civ{-}spouse$, $divorced$, $never{-}married$, $separated$, $widowed$, $married{-}spouse{-}absent$ and $married{-}AF{-}spouse$. It is closely associated with attribute $Relationship$ (the 8th attribute) which has 6 values: $wife$, $own{-}child$, $husband$, $not{-}in{-}family$, $other{-}relative$ and $unmarried$. If a person is classified as a wife (or husband), she (or he) must be a married person. However, we could not make the further judgement whether a married person is either a $married{-}civ{-}spouse$ or $married{-}AF{-}spouse$

due to two types of marriage being listed in the data set. There are 22379 instances of $Married{-}civ{-}spouse$ and 37 instances of $Married{-}AF{-}spouse$. It is obvious that civilian marriages account for the majority of marriages. Therefore, we can approximately infer that a married person belongs to a civilian marriage. That is, $Marital{-}status{=}married{-}civ{-}spouse$ is a near-generalization of $Relationship = husband$ and $Relationship{=}wife$.

To evaluate the effect of deleting $Marital{-}status{=}married{-}civ{-}spouse$ using $r = 0.99$, we apply NSR to NB and AODE but restrict its application to deleting only values of $Marital{-}status$. The error of $\text{NB}^{SR}_{0.99}$ is $0.1457 < 0.1617$ and that of $\text{AODE}^{SR}_{0.99}$ is $0.1383 < 0.1474$. These results suggest that the elimination of a near-generalization accounting for a large part of population to which near-specializations belong can be positive.

6.5.1.2 Attributes with Noise

Attribute Sex (the 10th attribute) has two values: $female$ and $male$. These values have a clear specialization–generalization relationship with the two values ($husband$ and $wife$) of attribute $Relationship$. That is, $Sex{=}female$ is a generalization of $Relationship{=}wife$ and $Sex{=}male$ is a generalization of $Relationship{=}husband$. However, due to noise in the data, the relation can not be detected by NB^{SR} and AODE^{SR}.

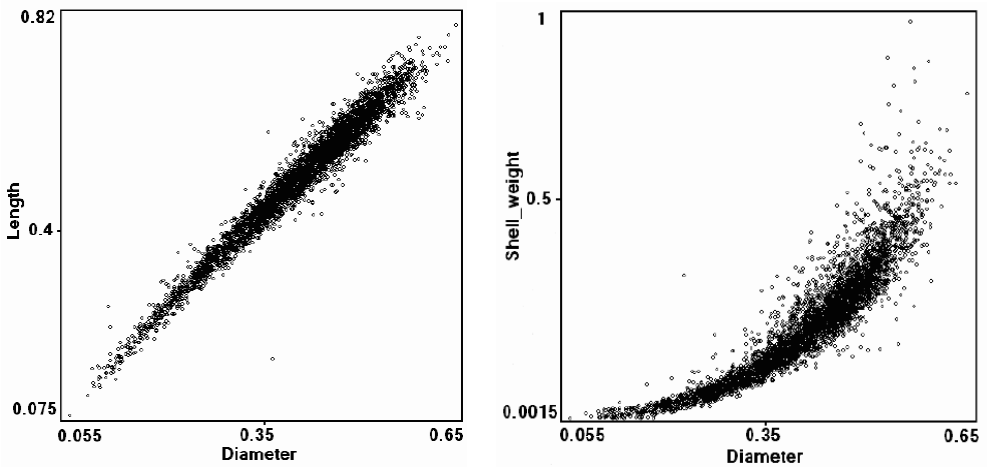
The values of $Relationship$ and Sex of the 7110th instance in Adult are $husband$ and $female$ respectively. Another two instances with noise are the 576th and 27142th instances in which $Relationship{=}wife$ and $Sex{=}male$. When we apply NSR to NB and AODE but restrict its application to deleting only values of Sex, $\text{NB}^{SR}_{0.99}$ and $\text{AODE}^{SR}_{0.99}$ have errors of $0.1551 (< 0.1617)$ and $0.1449 (< 0.1474)$ respectively. These results indicate that the near-generalization technique can be useful in at least some cases of noise.

6.5.2 Abalone

In Abalone, the classification task is to predict the age of an abalone from its physical measurements, many of which are closely correlated to one another. Since NB and

AODE cannot handle numeric classes, we select the only attribute (Sex) that has categorial values (M, F and I) as the class.

Figure 6.11 (a) presents the scatter graph that plots the values of *Length* versus the corresponding values of *Diameter* and Figure 6.11 (b) plots the values of *Shell_weight* versus the corresponding values of *Diameter*. While the data are discretized in our experiments, we do not show discretized values here as the discretization cut points will change from one cross-validation run to another and between cross-validation folds. It can be seen that the relationship between *Diameter* and *Length* is linear and that of *Diameter* and *Shell_weight* is roughly linear. Similar relationships are observed for other attributes (scatter graphs are not presented), specifically *Shucked_weight* and $Viscera\_weight$ are linearly related and $Whole\_weight$ and *Length* are roughly linearly related.



(a) *Length* vs. *Diameter* (b) *Shell_weight* vs. *Diameter*

Figure 6.11: Close correlation

As *Diameter* and *Length* are positively linearly related, it is logical to infer that an abalone with small diameter is shorter. However, there are some exceptional cases where abalones with small diameter measure longer than average. Due to these outliers, NB^{SR} and AODE^{SR} often cannot identify the relationship, while NB^{SR}_r and AODE^{SR}_r may detect it if a small value of r is used. For instance, NB^{SR} and AODE^{SR} find the relationship between *Diameter* and *Length* 9786 times in 50 runs of two-fold cross validation experiments, while $\text{NB}^{SR}_{0.99}$ and $\text{AODE}^{SR}_{0.99}$ detect the relationship 29357 times. Note that when the relationship is detectable, a deletion will only occur for

test cases that are not themselves outliers as, for example, a long abalone with small diameter will not be an instance of the detected near-generalization relationship.

For attributes, such as *Diameter* and *Shell_weight*, that are not perfectly correlated (but closely correlated), NB^{SR} and AODE^{SR} cannot find the relations most of the time. If highly associated attributes can be excluded, NB and AODE's accuracy might be further improved. As shown in Figure 6.4, NB$^{SR}_r$ and AODE$^{SR}_r$ have a largely downwards trend in error with decreasing values of r for this data.

6.5.3 Pima Indians Diabetes

All the 768 instances in Pima Indians Diabetes are females at least 21 years old. There is a strong relationship between age and the number of times that a female was pregnant. It is generally true that a young female will have had few pregnancies. To analyze the relationship between *Age* (the 8th attribute) and *Number_of_times_pregnant* (the 1st attribute), we use MDL discretization to discretize the continuous attributes. The resulting cut points for *Age* and *Number_of_times_pregnant* are 28.5 and 6.5 respectively.

Table 6.6 cross-tabulates the frequency of each range of values for each attribute. As we can see, there are only two outliers where females younger than 28.5 have been pregnant more than 6.5 times. $Age{>}28.5$ is a near-generalization of $Number\_of\_times\_pregnant{>}6.5$ and $Number\_of\_times\_pregnant{\leq}6.5$ is a near-generalization of $Age{\leq}28.5$. NB^{SR} and AODE^{SR} detect the relationship in few of the 50 runs in the experiment (those where both outliers are in the test set) and reduce NB and AODE's error from 0.2580 to 0.2569 and from 0.2541 to 0.2535 respectively.

Table 6.6: Frequency table for *Number_of_times_pregnant* and *Age*

	$Number\_of\_times\_pregnant \leq 6.5$	$Number\_of\_times\_pregnant > 6.5$
$Age \leq 28.5$	365	2
$Age > 28.5$	234	167

However, as shown in Figure 6.6, $\text{NB}^{SR}_{0.99}$ has a higher error than that of NB^{SR} and NB^{SR}_{r} ($0.80 \leq r \leq 0.98$). Similar results can be observed for $\text{AODE}^{SR}_{0.99}$. We examine those instances that are classified to different classes by NB^{SR} and $\text{NB}^{SR}_{0.99}$. The probability estimates of the two classes for these instances are close to 0.5. For example, NB^{SR} correctly classifies the 267th instance to the first class (the probability estimates for the two classes are 0.5659 and 0.4341), while $\text{NB}^{SR}_{0.99}$ incorrectly classifies the instance to the second class (the probability estimates for the two classes are 0.4947 and 0.5053). $\text{NB}^{SR}_{0.99}$ misclassifies 144 instances that are correctly classified by NB^{SR} and corrects 118 misclassifications of NB^{SR} in the 50 runs. Nonetheless, $\text{NB}^{SR}_{0.98}$ frequently and substantially improves probabilistic prediction of NB^{SR} and $\text{NB}^{SR}_{0.99}$. For example,NB^{SR} and $\text{NB}^{SR}_{0.99}$ misclassify the 503th instance to the second class (the probability estimates for the two classes are 0.4666 and 0.5334), and $\text{NB}^{SR}_{0.98}$ correctly classifies it to the first class (the probability estimates for the two classes are 0.6220 and 0.3780). $\text{NB}^{SR}_{0.98}$ corrects 480 misclassifications of NB^{SR} and misclassifies 418 instances that are correctly classified by NB^{SR} in the 50 runs. $\text{NB}^{SR}_{0.93}$ has the lowest error, correcting 710 misclassifications of NB^{SR} and misclassifying 502 instances that are correctly classified by NB^{SR} in the 50 runs. There does not appear to be any systematic reason for particular values of r performing better or worse with this data. Decreasing the value of r produces some advantageous attribute-value deletions together with some disadvantageous deletions, and it appears to be a matter of chance whether the advantageous or disadvantageous deletions dominate for any particular value of r.

6.6 Summary

In this chapter, we extend SR to NSR which detects near specialization-generalization relationship and deletes near-generalizations. We explore the reasons NSR proves profitable based on three exemplar data sets. When a near-generalization accounts for the majority of the population to which the corresponding near-specialization belongs, elimination of the near-generalization may excel. It might have an advantage when attributes are closely rather than perfectly associated. Furthermore, it may provide tolerance for noise.

Chapter 7

Conclusions and Future Work

This thesis has analyzed the strengths and weaknesses of previous semi-naive Bayesian techniques and proposed new algorithms that further improve the classification and conditional probability estimation accuracy of NB and AODE. We provide in the following sections a summary of its main results, some directions for future research and conclusions.

7.1 Summary of Results

In this section, we briefly summarize the results from this thesis.

Systematic Survey

- *A taxonomy of previous semi-naive Bayesian methods*

  This thesis provides a four category taxonomy of semi-naive Bayesian methods. The first group forms a new attribute set by deleting or joining attributes to remove harmful interdependencies and applies conventional NB to this attribute set. Examples include BSE, FSS and BSEJ. The second group directly addresses interdependencies between attributes by adding explicit arcs into NB's structure. Examples include TAN, SP-TAN, NBTree, LBR, AODE and MAPLMG. The third group accommodates violations of the attribute independence assumption by applying NB to a subset of training set. Examples include LWNB, NBTree and LBR. Therefore, the third group and the second group are not

mutually exclusive. The fourth group performs corrections to NB's probability estimates by making adjustments to the class or attribute conditional probabilities. Examples include APNB and IB. LBR and LWNB are lazy methods as they defer learning until classification time, and BSE, FSS, BSEJ, TAN, SP-TAN, NBTree, AODE, MAPLMG, APNB and IB are eager methods because they perform learning at training time.

- *Detailed time and space complexity analysis*

  The training time complexity of BSEJ and SP-TAN are $O\big(tkn^3\big)$, that of NBTree is $O\big(t^2kn^2/v\big)$ and that of APNB is $O\big(tn + t^2k^2\big)$. Hence, BSEJ and SP-TAN have very high training time if the number of attributes is large, NBTree if the number of instances and attributes are large and APNB if the number of instances and classes are large. These four algorithms have identical classification time complexity to NB. That is, classification time complexity is linear in the number of classes and attributes. Therefore, once models are formed, they can efficiently classify test instances. MAPLMG, BSE and FSS have relatively higher training time complexity than TAN, AODE and IB, whose training time complexities are moderate. The classification time complexity of LBR is $O\big(tkn^3\big)$. This hampers LBR's application when large numbers of examples are to be classified, although its low training time complexity makes it highly efficient when few classifications are required. LWNB is also a lazy method, but its classification time complexity, $O\big(tn + kn\big)$, is substantially lower than that of LBR. The classification time complexity of AODE and MAPLMG is $O\big(kn^2\big)$, which is higher than all the other methods except LBR and LWNB and considerably lower than that of LWNB when t is substantially greater than n and k (a usual case in the real world) and that of LBR. BSEJ has very high training space complexity of $O\big(tn + kv^n\big)$.

- *Empirical comparison of twelve semi-naive Bayesian methods on sixty data sets using the Friedman and Nemenyi tests*

  - *Error*

    Five semi-naive Bayesian methods, MAPLMG, LBR, AODE, LWNB and SP-TAN, significantly outperform NB. MAPLMG, LBR and AODE

achieve the lowest, second lowest and third lowest mean error ranks. The error of MAPLMG is not significantly lower than that of LBR, AODE, LWNB, SP-TAN and BSEJ.

- *Bias*

  All semi-naive Bayesian methods, except APNB and IB, have significantly lower mean bias ranks than NB. NBTree and LWNB have the lowest and second lowest mean bias ranks. The bias differences between NBTree and LWNB, BSEJ and LBR are not statistically significant.

- *Variance*

  All semi-naive Bayesian methods, except AODE, MAPLMG, APNB and IB, significantly increase the variance of NB. NBTree has the highest mean variance rank, which is higher, but not significantly higher, than the mean variance ranks of FSS, TAN, LWNB and BSEJ.

- *RMSE*

  Four semi-naive Bayesian methods, MAPLMG, AODE, LBR and SP-TAN, significantly reduce the RMSE of NB. MAPLMG achieves the lowest mean RMSE rank, AODE comes next and LBR obtains the third lowest mean RMSE rank.

- *Computing time*

  Computing time results are provided as an adjunct to time complexity analysis. These results should be treated with caution as various implementations may result in differences in efficiency. In our data collection, NBTree, MAPLMG, SP-TAN and BSEJ have high training time and LWNB has high classification time. LBR was excluded from time comparison as it was executed on a different machine. It is substantially slower than all the other methods at classification time.

- *Empirical comparison of NB, MAPLMG, LBR, AODE, LWNB, SP-TAN, logistic regression and LibSVM*

  LibSVM has the lowest mean error rank and significantly outperforms logistic regression and NB. MAPLMG has slightly higher mean error rank but substantially lower training time than LibSVM. All methods, but NB, have lower mean

error ranks than logistic regression, which has higher variance and significantly lower bias than NB.

- *Recommendations for selection between semi-naive Bayesian methods*

  These recommendations are based on the observations that low bias algorithms appear to have an advantage in error with large training sets, while low variance algorithms appear to have an advantage with small training sets. For extremely small data NB may prove best and for large data NBTree, LWNB, BSEJ and LBR may have an advantage if their computational profiles are appropriate to the task. AODE may prove advantageous for many classification tasks as it achieves very low variance and RMSE, intermediate bias, low training time complexity and modest classification time complexity. MAPLMG further reduces AODE's error and RMSE with substantially increased training time, therefore, it may excel for many classification problems if its computational cost can fall within the computational constraints of the given application.

Child Elimination for AODE

- *Development of new elimination strategies: Parent and Child Elimination, Child Elimination and Parent or Child Elimination*

- *Two explanations for why straightforward application of BSE to AODE proves ineffective*

  - AODE has higher variance compared with NB, and hence appropriate variance management is required.
  - Child selection appears to have greater effect than parent selection, as ODEs are combined using a linear function but attributes within an ODE are combined using a multiplicative function.

- *Effective child elimination strategies for AODE*

  Our extensive experiments on sixty data sets suggest that the types of attribute elimination that remove child attributes from within the constituent ODEs can significantly reduce bias and error, but only if a statistical test is employed to

provide variance management. In contrast, elimination of complete constituent ODEs does not consistently provide error reduction.

Subsumption Resolution

- *Identification and analysis of extreme but common relationships: generalization, substitution and duplication relationships*

  A duplication relationship is a special case of a substitution relationship, which in turn is a special case of generalization relationship. These relationships are common in the real world. In our data collection, the generalization relationship is detected on 49 out of 60 data sets.

- *Prove that deletion of generalizations is theoretically correct*

- *Development of a new learning technique, SR, for efficiently identifying occurrences of the generalization relationship and removing generalizations at classification time*

- *SR can in practice significantly improve both classification accuracy and the precision of conditional probability estimates.*

  SR efficiently identifies and deletes generalizations at classification time. When applied to NB, it significantly improves both classification and conditional probability estimation accuracy with increasing time complexity. When applied to AODE, it significantly improves both classification and conditional probability estimation accuracy with identical time complexity. Compared to BSE, it has significant advantages in probabilistic prediction and computation efficiency in the context of AODE.

- *Incrementality*

  SR only uses a two-dimensional table of probability estimates indexed by attribute values generated at training time to detect and delete generalizations. This probability estimates table can be updated incrementally. In consequence, SR does not interfere with NB and AODE's capacity for incremental learning.

- *The application of SR to AODE competes favorably with state-of-the-art semi-naive Bayesian methods.*

The Nemenyi test indicates that AODE^{SR} has lower mean error and RMSE ranks than AODE, LBR, LWNB and SP-TAN and higher than MAPLMG. This test does not differentiate AODE^{SR} from AODE as the power of the test is low when a large number of methods are compared. AODE^{SR} has considerably lower training time complexity relative to MAPLMG and SP-TAN, classification time complexity relative to LWNB in most cases and LBR and identical time complexity relative to AODE. We believe AODE^{SR} provides a reasonable trade-off between classification accuracy and computational efficiency and appears to be a promising approach for a wide range of classification problems.

Semi-Supervised Subsumption Resolution

- *Development of a new learning technique, SSSR, for using both labeled and unlabeled data to identify occurrences of the generalization relationship and remove generalizations at classification time*

  SSSR substantially reduces the variance of SR in the context of NB and AODE.

Near-Subsumption Resolution

- *Development of a new learning technique, NSR, for efficiently identifying occurrences of the near-generalization relationship and removing near-generalizations at classification time*

- *Theoretical and experimental analysis of the circumstances in which elimination of near-generalizations proves profitable*

  NSR proves profitable when:

  - A near-generalization accounts for the majority of the population to which the corresponding near-specialization belongs.
  - Attributes are closely rather than perfectly associated.
  - Data contains noise that may prevent detection of generalization relationships.

7.2 Future Work

This section discusses several interesting research topics that would allow us to further explore and expand the findings of this thesis.

- To avoid the problems that result from zero frequencies and zero probabilities, Laplace estimation, in keeping with Weka's default probability estimation method, was employed to estimate the base probabilities of semi-naive Bayesian methods. However, in our preliminary experiments, m-estimation ($m = 1$) often appears to lead to more accurate probabilities than Laplace estimation for NB and its variants we tested. It would be interesting to examine the relative performance of semi-naive Bayesian methods using m-estimation. Kohavi, Becker and Sommerfield (1997) reported that the classification accuracy of NB using $m < 1$ is usually higher than $m = 1$. Nonetheless, the effect of different values of m on NB still remains unclear. A promising research direction would seem to be detailed investigation into the effect of different values of m on NB and its variants and the selection of an appropriate m for them.

- The largest data set sizes employed in this thesis are modest relative to many data mining applications. It would be interesting to observe behaviors of semi-naive Bayesian algorithms on relatively large data sets.

- Since SR detects and deletes generalizations at classification time, it might be directly applied to lazy methods, such as k-nearest neighbors. However, the effect of elimination of generalizations on distance functions is not immediately clear. For example, assume that there are two attributes $Pregnant$ and $Gender$, the test instance is $\langle Gender{=}female, Pregnant{=}yes\rangle$ and the distance between two instances is defined as the number of attributes that have different values. The distance between the test instance and $\langle Gender{=}female, Pregnant{=}no\rangle$ is one and that of the test instance and $\langle Gender{=}male, Pregnant{=}no\rangle$ is two. If the generalization $Gender{=}female$ is deleted, both distances are one. In such case, elimination of generalizations may have negative effect on distances. In other cases, such as the case with presence of perfectly correlated attributes, elimination of generalizations may have positive effect. Investigation of the effect of SR on lazy methods would appear to be another interesting research

topic. In addition, SR may also be applied to methods that can deal with missing values as generalizations can be treated as missing values.

- This thesis explored the reasons Near-Subsumption Resolution proves profitable based on three exemplar data sets. However, the investigation does not provide any a priori guidelines for identifying in advance when Near-Subsumption Resolution is likely to be advantageous and this remains an open issue for future research.

- SR can provide a theoretically correct adjustment for the duplication, substitution and generalization relationships, however, it does not exploit other interdependence relationships. It would be useful to explore new techniques for repairing other forms of harmful interdependencies. A challenge in this work is how to efficiently identify interdependence relationships between attributes and remove them.

7.3 Concluding Remarks

The elegant simplicity, computational efficiency and classification efficacy of NB fosters ongoing interest in exploring semi-naive Bayesian algorithms that improve NB's accuracy by alleviating the attribute interdependence problem. This thesis analyzes twelve key semi-naive Bayesian techniques and provides insight into the strengths and weaknesses of them. Armed with such insight and an extensive comparative study, it offers general suggestions for selection between these techniques.

The comparative study conducted in this thesis supports previous findings of strong performance from AODE, which significantly improves the accuracy of NB with modest training and classification time. Motivated by the desire to further improve classification and conditional probability estimation accuracy of AODE, this thesis develops several novel and effective semi-naive Bayesian techniques. The first three techniques eliminate child attributes from within the constituent ODEs, thereby significantly improving AODE's prediction accuracy. We provide theoretical explanations for why child elimination techniques might be effective, and empirical evidence, as observed in real world data sets.

This thesis also proposes, from a fresh perspective, a new technique SR to efficiently identify a frequently observed interdependence between two attribute values such that one is a generalization of the other and remove the interdependence by deleting generalizations at classification time. We prove the theorem that elimination of generalizations is theoretically correct and demonstrate experimentally that it can in practice significantly improve both classification accuracy and the precision of conditional probability estimates. When applied to AODE, SR achieves a desirable balance between classification accuracy and computational efficiency, competing favorably with state-of-the-art semi-naive Bayesian methods without undue time complexity. In addition, SR is suited to incremental and semi-supervised learning. This thesis also explores circumstances under which elimination of near-generalizations proves beneficial.

The success of our child elimination techniques lends support to the approach we followed of seeking to improve a learner's classification accuracy by seeking an appropriate balance between bias and variance. The success of our Subsumption Resolution technique suggests that it might be profitable to explore further approaches to efficiently identify and remove other forms of interdependencies.

Appendix A

Error, Bias, Variance and RMSE Results

This appendix presents the detailed results for Error (Table A.1), Bias (Table A.2), Variance (Table A.3) and RMSE (Table A.4). The data sets are in the number sequence of Table 3.4. Experimental methodology is described in Section 3.3.2. Since we have not obtained the results of LibSVM on Adult and Connect-4 Opening, they are not presented in each table.

Table A.1: Error (the data sets are in the number sequence of Table 3.4)

No.	NB	BSE	FSS	BSEJ	TAN	SP-TAN	NBTree	LBR	AODE	MAPLMG	LWNB	APNB	IB	NB^{SR}	$AODE^{SR}$	Logistic	LibSVM
1	0.4822	0.4796	0.4723	0.4813	0.4717	0.4774	0.4799	0.4737	0.4632	0.4626	0.4698	0.4776	0.4820	0.4821	0.4631	0.4445	0.4458
2	0.1617	0.1399	0.1394	0.1347	0.1415	0.1415	0.1390	0.1342	0.1474	0.1378	0.1474	0.1398	0.1345	0.1636	0.1424	0.1491	
3	0.1109	0.1080	0.1095	0.1071	0.0877	0.0973	0.1056	0.1033	0.1120	0.1105	0.0830	0.1007	0.1240	0.0959	0.0928	0.1362	0.1121
4	0.3864	0.3888	0.3365	0.3947	0.3290	0.3797	0.3321	0.3719	0.3863	0.3861	0.2952	0.3777	0.6060	0.3204	0.3126	0.3059	0.2942
5	0.4056	0.3973	0.3969	0.3756	0.3339	0.3585	0.3448	0.3737	0.3502	0.3454	0.2860	0.4025	0.5159	0.3915	0.3343	0.4003	0.3860
6	0.2459	0.2486	0.2485	0.2476	0.2509	0.2476	0.2483	0.2461	0.2494	0.2498	0.2498	0.2459	0.2455	0.2459	0.2494	0.1096	0.0225
7	0.0275	0.0292	0.0338	0.0304	0.0550	0.0283	0.0304	0.0275	0.0336	0.0332	0.0333	0.0275	0.0279	0.0275	0.0336	0.0819	0.0366
8	0.1598	0.1595	0.1647	0.0825	0.1123	0.0769	0.0860	0.0973	0.1125	0.0987	0.1066	0.1431	0.1849	0.1598	0.1125	0.0728	0.0261
9	0.2794	0.2732	0.2745	0.2138	0.2386	0.2392	0.2752	0.2266	0.2454	0.2415	0.2036	0.2686	0.2506	0.2742	0.2395	0.2425	
10	0.3083	0.3000	0.3450	0.3100	0.4317	0.3167	0.3083	0.3083	0.3150	0.3017	0.3242	0.3083	0.3083	0.3083	0.3150	0.3475	0.3600
11	0.4992	0.5017	0.5033	0.5027	0.5054	0.5000	0.5012	0.5001	0.4964	0.4960	0.5175	0.5009	0.4521	0.4992	0.4963	0.4923	0.4863
12	0.1429	0.1455	0.1504	0.1451	0.1540	0.1436	0.1485	0.1427	0.1392	0.1370	0.1516	0.1429	0.1397	0.1381	0.1380	0.1608	0.1452
13	0.2615	0.2614	0.2777	0.2681	0.2939	0.2455	0.2734	0.2612	0.2518	0.2467	0.2467	0.2621	0.2615	0.2548	0.2526	0.2678	0.2791
14	0.0262	0.0261	0.0333	0.0258	0.0337	0.0255	0.0395	0.0262	0.0260	0.0258	0.0231	0.0266	0.0262	0.0257	0.0261	0.0340	0.0263
15	0.3450	0.3441	0.3447	0.3418	0.3437	0.3424	0.3437	0.3431	0.3441	0.3441	0.3427	0.3423	0.3400	0.3450	0.3438	0.2811	0.3179
16	0.2635	0.2651	0.2739	0.2702	0.2798	0.2662	0.2742	0.2648	0.2587	0.2596	0.2743	0.2640	0.2629	0.2629	0.2604	0.2569	0.2503
17	0.3035	0.3030	0.3064	0.2981	0.3005	0.2975	0.3034	0.3010	0.3003	0.3005	0.2936	0.3039	0.3293	0.3031	0.2967	0.3194	0.2547
18	0.2778	0.2796	0.2794	0.2793	0.2839	0.2807	0.2829	0.2792	0.2822	0.2820	0.2839	0.2813	0.2788	0.2778	0.2833	0.2667	0.2712
19	0.1743	0.1776	0.2036	0.1809	0.1852	0.1754	0.1912	0.1745	0.1719	0.1723	0.1983	0.1746	0.1752	0.1743	0.1719	0.1859	0.1870
20	0.1730	0.1794	0.1894	0.1745	0.1675	0.1730	0.1877	0.1732	0.1708	0.1717	0.1747	0.1741	0.1745	0.1785	0.1737	0.2234	0.1859
21	0.1915	0.1852	0.1701	0.1867	0.2028	0.1885	0.1928	0.1859	0.1826	0.1806	0.2014	0.1902	0.1713	0.1914	0.1810	0.2455	0.1827
22	0.0991	0.0878	0.0524	0.0885	0.0731	0.0894	0.0609	0.0679	0.0588	0.0535	0.0500	0.0997	0.0954	0.0990	0.0588	0.0856	0.0499
23	0.1633	0.1741	0.1866	0.1726	0.1676	0.1682	0.1763	0.1644	0.1613	0.1612	0.1745	0.1632	0.1725	0.1637	0.1601	0.1791	0.1805
24	0.0215	0.0154	0.0131	0.0124	0.0123	0.0124	0.0125	0.0142	0.0198	0.0172	0.0178	0.0197	0.0222	0.0246	0.0191	0.0342	0.0307
25	0.1063	0.1044	0.1039	0.1042	0.0892	0.1056	0.1067	0.1053	0.0909	0.0901	0.0867	0.1064	0.1064	0.1123	0.0860	0.1608	0.0660
26	0.0601	0.0608	0.0601	0.0604	0.0668	0.0613	0.0623	0.0601	0.0605	0.0613	0.0616	0.0601	0.0603	0.0599	0.0603	0.0505	0.0445
27	0.1276	0.0710	0.0552	0.0354	0.0675	0.0563	0.0287	0.0421	0.0946	0.0598	0.0355	0.1255	0.0675	0.1248	0.0802	0.0286	0.0109
28	0.1305	0.1309	0.1484	0.1354	0.1425	0.1312	0.1305	0.1305	0.1298	0.1298	0.1225	0.1305	0.1333	0.1305	0.1298	0.0972	0.1340
29	0.2614	0.2626	0.2626	0.2638	0.2676	0.2653	0.2660	0.2629	0.2622	0.2626	0.2727	0.2636	0.2602	0.2614	0.2622	0.2691	0.2714

continued on next page

Error (continued from previous page)

No.	NB	BSE	FSS	BSEJ	TAN	SP-TAN	NBTree	LBR	AODE	MAPLMG	LWNB	APNB	IB	NB^{SR}	$AODE^{SR}$	Logistic	LibSVM
30	0.2679	0.2608	0.2611	0.2023	0.1774	0.1585	0.1858	0.1755	0.1365	0.1256	0.0797	0.2615	0.2679	0.2673	0.1345	0.2281	0.0313
31	0.4222	0.4222	0.4222	0.4222	0.4221	0.4222	0.4222	0.4222	0.4222	0.4222	0.4222	0.4222	0.4210	0.4222	0.4222	0.3268	0.2987
32	0.5519	0.5519	0.5813	0.5506	0.5469	0.5513	0.5519	0.5500	0.5581	0.5581	0.5681	0.5500	0.5544	0.5519	0.5581	0.5525	0.5869
33	0.1665	0.1701	0.2150	0.1668	0.1951	0.1645	0.1926	0.1654	0.1569	0.1573	0.1700	0.1664	0.1665	0.1742	0.1564	0.2530	0.2104
34	0.2275	0.1905	0.1893	0.1717	0.1726	0.1723	0.1659	0.1658	0.1839	0.1749	0.1676	0.2010	0.1765	0.2227	0.1805	0.2091	0.1451
35	0.0519	0.0108	0.0063	0.0011	0.0043	0.0009	0.0001	0.0008	0.0006	0.0005	0.0000	0.0235	0.0431	0.0208	0.0006	0.0016	0.0000
36	0.3132	0.2946	0.2764	0.2796	0.3216	0.2358	0.2279	0.2503	0.2732	0.2373	0.2556	0.3100	0.3132	0.2902	0.2537	0.2940	0.3914
37	0.0496	0.0535	0.0580	0.0530	0.0645	0.0514	0.0574	0.0496	0.0549	0.0553	0.0613	0.0507	0.0498	0.0489	0.0547	0.0499	0.0490
38	0.0979	0.0980	0.1014	0.0492	0.0753	0.0559	0.0436	0.0451	0.0745	0.0787	0.0349	0.0863	0.0929	0.0979	0.0745	0.0751	0.0019
39	0.0815	0.0788	0.0825	0.0665	0.0469	0.0685	0.0861	0.0649	0.0349	0.0348	0.0287	0.0813	0.0764	0.0814	0.0346	0.0621	0.0107
40	0.0677	0.0480	0.0459	0.0422	0.0484	0.0446	0.0374	0.0385	0.0347	0.0346	0.0340	0.0532	0.0585	0.0673	0.0347	0.0359	0.0338
41	0.1272	0.1206	0.1200	0.0685	0.0565	0.0386	0.0567	0.0512	0.0288	0.0275	0.0221	0.1128	0.1216	0.1272	0.0288	0.0467	0.0046
42	0.2580	0.2537	0.2525	0.2532	0.2527	0.2532	0.2555	0.2570	0.2541	0.2533	0.2555	0.2570	0.2501	0.2569	0.2535	0.2327	0.2361
43	0.3393	0.3273	0.3073	0.3460	0.3864	0.3569	0.3604	0.3391	0.3487	0.3498	0.3516	0.3313	0.3224	0.3400	0.3491	0.4342	0.3013
44	0.5680	0.5735	0.6195	0.5791	0.5617	0.5696	0.6000	0.5683	0.5608	0.5611	0.5824	0.5673	0.5680	0.5663	0.5609	0.6865	0.5858
45	0.1298	0.1298	0.2100	0.1353	0.2470	0.1340	0.1900	0.1298	0.1630	0.1655	0.1513	0.1306	0.1415	0.1298	0.1630	0.1277	0.2794
46	0.1006	0.0843	0.0731	0.0626	0.0548	0.0653	0.0682	0.0739	0.0596	0.0581	0.0522	0.0977	0.0996	0.0952	0.0565	0.0511	0.0380
47	0.0303	0.0264	0.0254	0.0260	0.0266	0.0266	0.0250	0.0277	0.0283	0.0256	0.0272	0.0298	0.0259	0.0296	0.0271	0.0338	0.0338
48	0.3607	0.3598	0.3621	0.2649	0.2859	0.2829	0.2552	0.2545	0.2927	0.2841	0.2295	0.3550	0.3335	0.3606	0.2887	0.4078	0.1635
49	0.1873	0.1812	0.1778	0.1716	0.1635	0.1716	0.1704	0.1716	0.1630	0.1620	0.1667	0.1849	0.4997	0.1865	0.1627	0.1617	0.1639
50	0.2595	0.2655	0.2848	0.2632	0.2776	0.2607	0.2769	0.2611	0.2581	0.2578	0.2786	0.2589	0.2604	0.2613	0.2630	0.3053	0.1652
51	0.1027	0.0833	0.0780	0.0751	0.0831	0.0889	0.0758	0.0709	0.0730	0.0645	0.0564	0.1025	0.0934	0.1020	0.0728	0.0777	0.0835
52	0.0481	0.0469	0.0528	0.0491	0.0594	0.0473	0.0484	0.0480	0.0444	0.0445	0.0656	0.0482	0.0437	0.0481	0.0444	0.1324	0.1498
53	0.0590	0.0575	0.0543	0.0577	0.0253	0.0578	0.0724	0.0546	0.0349	0.0337	0.0204	0.0590	0.0619	0.0581	0.0318	0.2497	0.0102
54	0.2909	0.2866	0.2900	0.2164	0.2657	0.2801	0.1899	0.2282	0.2509	0.2493	0.0916	0.2745	0.2603	0.2909	0.2509	0.0244	0.0170
55	0.4165	0.4133	0.3942	0.3761	0.3509	0.3786	0.3464	0.3444	0.3378	0.3360	0.3264	0.4084	0.4042	0.4108	0.3362	0.2217	0.1786
56	0.3338	0.3338	0.3338	0.3338	0.3338	0.3338	0.3338	0.3338	0.3338	0.3338	0.3338	0.3338	0.3337	0.3338	0.3338	0.3401	0.3377
57	0.4940	0.4503	0.4573	0.4198	0.4378	0.3762	0.3709	0.4011	0.3865	0.3597	0.2522	0.4949	0.4904	0.4852	0.3674	0.2841	0.0687
58	0.2009	0.1917	0.1945	0.1742	0.1843	0.1780	0.1886	0.1742	0.1560	0.1559	0.1824	0.1696	0.2009	0.2009	0.1536	0.1370	0.1387
59	0.0274	0.0329	0.0438	0.0303	0.0436	0.0278	0.0364	0.0274	0.0265	0.0264	0.0328	0.0274	0.0275	0.0279	0.0272	0.0393	0.0256
60	0.0901	0.0891	0.1206	0.0877	0.1030	0.0877	0.0891	0.0903	0.0792	0.0792	0.0713	0.0901	0.0901	0.0966	0.0844	0.0846	0.0743

Table A.2: Bias (the data sets are in the number sequence of Table 3.4)

No.	NB	BSE	FSS	BSEJ	TAN	SP-TAN	NBTree	LBR	AODE	MAPLMG	LWNB	APNB	IB	NB^{SR}	$AODE^{SR}$	Logistic	LibSVM
1	0.4146	0.3947	0.3420	0.3371	0.3044	0.3787	0.3507	0.3422	0.3166	0.3154	0.2870	0.3679	0.4028	0.4143	0.3160	0.3849	0.3327
2	0.1516	0.1130	0.1164	0.1218	0.1250	0.1152	0.1219	0.1199	0.1362	0.1264	0.1127	0.1306	0.1280	0.1525	0.1318	0.1408	
3	0.0659	0.0602	0.0558	0.0597	0.0585	0.0543	0.0555	0.0592	0.0648	0.0641	0.0503	0.0601	0.0660	0.0667	0.0610	0.0931	0.0724
4	0.3009	0.2989	0.2221	0.2976	0.1828	0.2906	0.2108	0.2639	0.3002	0.3000	0.1938	0.2776	0.2778	0.2163	0.2109	0.1482	0.1778
5	0.2763	0.2668	0.2230	0.2449	0.1798	0.2268	0.1877	0.2243	0.2251	0.2202	0.1522	0.2703	0.2830	0.2642	0.2061	0.1733	0.2016
6	0.1275	0.1295	0.1289	0.1293	0.1315	0.1289	0.1287	0.1280	0.1295	0.1302	0.1303	0.1275	0.1273	0.1275	0.1295	0.0689	0.0068
7	0.0263	0.0258	0.0255	0.0251	0.0312	0.0257	0.0254	0.0263	0.0284	0.0275	0.0276	0.0263	0.0265	0.0263	0.0284	0.0338	0.0289
8	0.1099	0.1093	0.1081	0.0579	0.0653	0.0491	0.0367	0.0462	0.0662	0.0549	0.0489	0.0841	0.1355	0.1099	0.0662	0.0512	0.0083
9	0.2649	0.2569	0.2545	0.1610	0.2250	0.1981	0.2508	0.1484	0.2287	0.2235	0.1617	0.2539	0.2413	0.2609	0.2218	0.2347	
10	0.1656	0.1583	0.1956	0.1666	0.2941	0.1718	0.1656	0.1656	0.1734	0.1687	0.1939	0.1656	0.1656	0.1656	0.1734	0.1709	0.2137
11	0.4315	0.3866	0.3554	0.3686	0.3605	0.4028	0.4086	0.4083	0.4071	0.3973	0.3502	0.4057	0.3355	0.4316	0.4070	0.4019	0.3338
12	0.1199	0.1144	0.1067	0.1147	0.1102	0.1172	0.1027	0.1179	0.1157	0.1112	0.1100	0.1194	0.1127	0.1143	0.1116	0.1146	0.1102
13	0.1904	0.1830	0.1621	0.1747	0.1918	0.1427	0.1600	0.1844	0.1859	0.1786	0.1556	0.1895	0.1904	0.1848	0.1772	0.1416	0.1664
14	0.0146	0.0122	0.0112	0.0112	0.0129	0.0137	0.0130	0.0146	0.0140	0.0138	0.0099	0.0147	0.0146	0.0144	0.0140	0.0169	0.0157
15	0.2280	0.2302	0.2314	0.2357	0.2349	0.2321	0.2317	0.2314	0.2295	0.2295	0.2359	0.2308	0.2438	0.2280	0.2321	0.2247	0.2241
16	0.2059	0.1986	0.1929	0.1939	0.1857	0.1978	0.1842	0.2021	0.2000	0.1998	0.1968	0.2054	0.2019	0.2046	0.1996	0.1924	0.1972
17	0.1907	0.1842	0.1812	0.1812	0.1787	0.1816	0.1771	0.1867	0.1831	0.1821	0.1695	0.1911	0.1940	0.1899	0.1796	0.2242	0.1513
18	0.2144	0.2130	0.2140	0.2155	0.2243	0.2176	0.2166	0.2139	0.2145	0.2142	0.2217	0.2142	0.2210	0.2144	0.2172	0.2232	0.2171
19	0.1413	0.1366	0.1252	0.1391	0.1319	0.1386	0.1282	0.1409	0.1380	0.1371	0.1389	0.1403	0.1365	0.1412	0.1365	0.1308	0.1407
20	0.1326	0.1296	0.1209	0.1264	0.1146	0.1301	0.1189	0.1322	0.1277	0.1279	0.1219	0.1289	0.1190	0.1351	0.1258	0.1143	0.1271
21	0.1662	0.1541	0.1284	0.1484	0.1464	0.1570	0.1343	0.1541	0.1537	0.1512	0.1458	0.1615	0.1421	0.1661	0.1503	0.1385	0.1370
22	0.0930	0.0736	0.0386	0.0736	0.0535	0.0787	0.0334	0.0499	0.0512	0.0450	0.0366	0.0912	0.0890	0.0929	0.0512	0.0338	0.0348
23	0.1498	0.1405	0.1382	0.1405	0.1183	0.1437	0.1375	0.1492	0.1473	0.1471	0.1278	0.1496	0.1531	0.1502	0.1455	0.1292	0.1236
24	0.0149	0.0095	0.0082	0.0072	0.0075	0.0073	0.0061	0.0089	0.0132	0.0113	0.0095	0.0139	0.0144	0.0166	0.0119	0.0240	0.0186
25	0.0865	0.0840	0.0629	0.0850	0.0677	0.0859	0.0630	0.0843	0.0721	0.0710	0.0668	0.0867	0.0854	0.0876	0.0666	0.0816	0.0417
26	0.0388	0.0386	0.0382	0.0381	0.0424	0.0391	0.0390	0.0388	0.0392	0.0392	0.0389	0.0388	0.0385	0.0385	0.0389	0.0270	0.0293
27	0.1076	0.0453	0.0450	0.0220	0.0550	0.0374	0.0096	0.0238	0.0745	0.0475	0.0179	0.1035	0.0606	0.1059	0.0617	0.0199	0.0043
28	0.0553	0.0542	0.0656	0.0581	0.0556	0.0561	0.0553	0.0553	0.0565	0.0564	0.0499	0.0553	0.0573	0.0553	0.0565	0.0358	0.0622
29	0.2262	0.2261	0.2261	0.2243	0.2300	0.2240	0.2239	0.2255	0.2267	0.2267	0.2266	0.2261	0.2296	0.2262	0.2267	0.2270	0.2251

continued on next page

Bias (continued from previous page)

No.	NB	BSE	FSS	BSEJ	TAN	SP-TAN	NBTree	LBR	AODE	MAPLMG	LWNB	APNB	IB	NB^{SR}	$AODE^{SR}$	Logistic	LibSVM
30	0.2244	0.2183	0.2154	0.1120	0.1085	0.0805	0.0909	0.0761	0.0967	0.0861	0.0396	0.2132	0.2244	0.2237	0.0948	0.1981	0.0152
31	0.3553	0.3553	0.3553	0.3553	0.3552	0.3553	0.3553	0.3553	0.3553	0.3553	0.3553	0.3553	0.3664	0.3553	0.3553	0.2614	0.2121
32	0.3589	0.3618	0.3186	0.3552	0.3349	0.3584	0.3589	0.3539	0.3577	0.3581	0.3632	0.3497	0.3535	0.3589	0.3577	0.3604	0.3514
33	0.1261	0.1217	0.1257	0.1163	0.1195	0.1204	0.0992	0.1246	0.1201	0.1202	0.1194	0.1260	0.1261	0.1325	0.1172	0.1227	0.1308
34	0.2090	0.1553	0.1475	0.1279	0.1277	0.1262	0.1206	0.1189	0.1539	0.1413	0.1167	0.1618	0.1418	0.2006	0.1507	0.2045	0.1194
35	0.0488	0.0103	0.0027	0.0005	0.0032	0.0004	0.0000	0.0002	0.0005	0.0003	0.0000	0.0214	0.0337	0.0182	0.0005	0.0001	0.0000
36	0.2315	0.2116	0.2058	0.2035	0.1926	0.1368	0.1516	0.1561	0.1925	0.1708	0.1483	0.2223	0.2315	0.2092	0.1731	0.1332	0.2375
37	0.0277	0.0269	0.0266	0.0273	0.0238	0.0264	0.0284	0.0277	0.0308	0.0304	0.0341	0.0278	0.0279	0.0252	0.0294	0.0219	0.0260
38	0.0903	0.0903	0.0898	0.0334	0.0624	0.0416	0.0188	0.0254	0.0658	0.0671	0.0136	0.0748	0.0844	0.0903	0.0658	0.0686	0.0004
39	0.0684	0.0612	0.0510	0.0478	0.0303	0.0503	0.0291	0.0283	0.0241	0.0238	0.0172	0.0669	0.0620	0.0681	0.0238	0.0236	0.0065
40	0.0521	0.0316	0.0316	0.0273	0.0331	0.0276	0.0226	0.0247	0.0245	0.0236	0.0229	0.0419	0.0401	0.0510	0.0243	0.0276	0.0254
41	0.1113	0.0972	0.0951	0.0396	0.0365	0.0176	0.0218	0.0202	0.0194	0.0176	0.0117	0.0928	0.1011	0.1113	0.0194	0.0361	0.0027
42	0.1801	0.1784	0.1801	0.1767	0.1765	0.1776	0.1782	0.1797	0.1794	0.1786	0.1755	0.1799	0.1818	0.1794	0.1784	0.2064	0.1953
43	0.2703	0.2681	0.2740	0.2730	0.2798	0.2701	0.2733	0.2703	0.2843	0.2850	0.2894	0.2717	0.2830	0.2706	0.2846	0.2590	0.2847
44	0.4105	0.3981	0.3981	0.3983	0.3835	0.3993	0.3671	0.4081	0.4055	0.4039	0.3983	0.4071	0.4105	0.4079	0.4035	0.3809	0.3854
45	0.0637	0.0638	0.0737	0.0651	0.0910	0.0643	0.0708	0.0637	0.0833	0.0843	0.0724	0.0639	0.0696	0.0637	0.0833	0.0625	0.1474
46	0.0743	0.0554	0.0400	0.0374	0.0299	0.0367	0.0286	0.0352	0.0382	0.0369	0.0299	0.0710	0.0728	0.0691	0.0361	0.0331	0.0208
47	0.0263	0.0224	0.0221	0.0211	0.0222	0.0220	0.0185	0.0220	0.0247	0.0223	0.0207	0.0254	0.0232	0.0255	0.0236	0.0286	0.0238
48	0.3304	0.3151	0.3113	0.1883	0.2461	0.2214	0.1768	0.1871	0.2564	0.2441	0.1640	0.3148	0.2945	0.3303	0.2504	0.3904	0.1171
49	0.1769	0.1596	0.1571	0.1490	0.1492	0.1484	0.1450	0.1508	0.1525	0.1517	0.1466	0.1648	0.4300	0.1761	0.1523	0.1513	0.1492
50	0.1754	0.1697	0.1603	0.1667	0.1606	0.1731	0.1535	0.1738	0.1716	0.1712	0.1515	0.1742	0.1732	0.1771	0.1664	0.1547	0.0892
51	0.0943	0.0664	0.0539	0.0587	0.0693	0.0757	0.0405	0.0475	0.0633	0.0547	0.0362	0.0931	0.0840	0.0933	0.0633	0.0612	0.0577
52	0.0394	0.0364	0.0350	0.0359	0.0375	0.0369	0.0392	0.0370	0.0342	0.0340	0.0330	0.0391	0.0333	0.0394	0.0342	0.0441	0.0999
53	0.0408	0.0394	0.0253	0.0395	0.0106	0.0398	0.0194	0.0331	0.0207	0.0196	0.0115	0.0407	0.0426	0.0402	0.0187	0.0884	0.0045
54	0.2502	0.2350	0.2196	0.0876	0.1759	0.1912	0.0612	0.1176	0.2000	0.1951	0.0422	0.2350	0.2217	0.2502	0.2000	0.0168	0.0162
55	0.2966	0.2885	0.2485	0.2411	0.2022	0.2475	0.1885	0.2053	0.2095	0.2063	0.1894	0.2855	0.2832	0.2870	0.2049	0.1513	0.1001
56	0.3293	0.3293	0.3293	0.3293	0.3293	0.3293	0.3293	0.3293	0.3293	0.3293	0.3293	0.3293	0.3294	0.3293	0.3293	0.3232	0.3085
57	0.2487	0.2332	0.2393	0.1717	0.1859	0.1390	0.1253	0.1512	0.1598	0.1445	0.0768	0.2461	0.2369	0.2459	0.1496	0.1204	0.0118
58	0.1762	0.1440	0.1371	0.1244	0.1211	0.1336	0.1016	0.1110	0.1176	0.1144	0.1033	0.1328	0.1762	0.1762	0.1141	0.1143	0.1120
59	0.0146	0.0159	0.0156	0.0153	0.0193	0.0146	0.0146	0.0146	0.0151	0.0151	0.0150	0.0146	0.0149	0.0142	0.0148	0.0203	0.0116
60	0.0538	0.0493	0.0526	0.0495	0.0580	0.0504	0.0416	0.0539	0.0446	0.0446	0.0393	0.0538	0.0538	0.0574	0.0475	0.0295	0.0348

Table A.3: Variance (the data sets are in the number sequence of Table 3.4)

No.	NB	BSE	FSS	BSEJ	TAN	SP-TAN	NBTree	LBR	AODE	MAPLMG	LWNB	APNB	IB	NB^{SR}	$AODE^{SR}$	Logistic	LibSVM
1	0.0676	0.0850	0.1304	0.1442	0.1673	0.0987	0.1292	0.1315	0.1466	0.1472	0.1828	0.1097	0.0792	0.0678	0.1470	0.0596	0.1131
2	0.0101	0.0269	0.0230	0.0129	0.0165	0.0262	0.0171	0.0143	0.0112	0.0114	0.0347	0.0092	0.0065	0.0112	0.0105	0.0083	
3	0.0450	0.0478	0.0538	0.0474	0.0291	0.0431	0.0501	0.0441	0.0472	0.0464	0.0327	0.0405	0.0580	0.0292	0.0318	0.0431	0.0397
4	0.0855	0.0900	0.1144	0.0970	0.1462	0.0891	0.1214	0.1079	0.0861	0.0862	0.1015	0.1001	0.3282	0.1040	0.1017	0.1577	0.1164
5	0.1292	0.1305	0.1739	0.1307	0.1540	0.1317	0.1571	0.1493	0.1251	0.1252	0.1338	0.1323	0.2329	0.1274	0.1283	0.2270	0.1844
6	0.1184	0.1191	0.1196	0.1183	0.1194	0.1187	0.1196	0.1182	0.1198	0.1196	0.1194	0.1184	0.1182	0.1184	0.1198	0.0407	0.0157
7	0.0012	0.0033	0.0084	0.0054	0.0238	0.0026	0.0050	0.0012	0.0052	0.0057	0.0057	0.0012	0.0014	0.0012	0.0052	0.0481	0.0077
8	0.0499	0.0503	0.0566	0.0246	0.0470	0.0278	0.0492	0.0511	0.0462	0.0438	0.0577	0.0591	0.0494	0.0499	0.0462	0.0216	0.0179
9	0.0145	0.0164	0.0200	0.0528	0.0136	0.0412	0.0244	0.0781	0.0167	0.0180	0.0420	0.0147	0.0093	0.0132	0.0177	0.0078	
10	0.1427	0.1417	0.1494	0.1434	0.1375	0.1449	0.1427	0.1427	0.1416	0.1330	0.1302	0.1427	0.1427	0.1427	0.1416	0.1766	0.1463
11	0.0677	0.1151	0.1480	0.1341	0.1450	0.0972	0.0925	0.0919	0.0893	0.0986	0.1673	0.0951	0.1166	0.0677	0.0894	0.0904	0.1525
12	0.0230	0.0312	0.0437	0.0304	0.0438	0.0264	0.0458	0.0248	0.0236	0.0258	0.0415	0.0234	0.0270	0.0239	0.0264	0.0461	0.0349
13	0.0711	0.0784	0.1157	0.0934	0.1021	0.1028	0.1134	0.0767	0.0659	0.0681	0.0911	0.0726	0.0711	0.0700	0.0754	0.1263	0.1127
14	0.0116	0.0139	0.0221	0.0146	0.0207	0.0118	0.0264	0.0116	0.0120	0.0120	0.0132	0.0119	0.0116	0.0112	0.0121	0.0171	0.0106
15	0.1170	0.1139	0.1134	0.1061	0.1087	0.1104	0.1119	0.1117	0.1146	0.1146	0.1069	0.1115	0.0962	0.1170	0.1117	0.0563	0.0937
16	0.0576	0.0666	0.0810	0.0764	0.0942	0.0685	0.0900	0.0628	0.0587	0.0599	0.0775	0.0585	0.0610	0.0583	0.0608	0.0645	0.0531
17	0.1128	0.1188	0.1252	0.1169	0.1217	0.1159	0.1263	0.1143	0.1172	0.1184	0.1241	0.1128	0.1353	0.1132	0.1171	0.0952	0.1034
18	0.0635	0.0666	0.0654	0.0638	0.0595	0.0631	0.0663	0.0652	0.0676	0.0677	0.0621	0.0671	0.0578	0.0635	0.0661	0.0434	0.0541
19	0.0330	0.0410	0.0784	0.0418	0.0533	0.0368	0.0630	0.0336	0.0339	0.0352	0.0593	0.0342	0.0387	0.0331	0.0353	0.0550	0.0463
20	0.0405	0.0498	0.0686	0.0481	0.0529	0.0429	0.0689	0.0410	0.0431	0.0439	0.0528	0.0451	0.0554	0.0434	0.0479	0.1090	0.0588
21	0.0253	0.0311	0.0417	0.0383	0.0564	0.0314	0.0585	0.0318	0.0289	0.0294	0.0556	0.0287	0.0292	0.0253	0.0307	0.1070	0.0456
22	0.0061	0.0142	0.0138	0.0149	0.0197	0.0108	0.0275	0.0180	0.0076	0.0085	0.0134	0.0085	0.0065	0.0061	0.0076	0.0517	0.0151
23	0.0135	0.0337	0.0484	0.0321	0.0493	0.0244	0.0388	0.0152	0.0140	0.0140	0.0467	0.0136	0.0194	0.0135	0.0147	0.0499	0.0569
24	0.0066	0.0060	0.0049	0.0052	0.0048	0.0051	0.0064	0.0053	0.0065	0.0059	0.0082	0.0058	0.0078	0.0081	0.0072	0.0102	0.0121
25	0.0198	0.0204	0.0411	0.0192	0.0215	0.0198	0.0437	0.0210	0.0188	0.0191	0.0199	0.0197	0.0210	0.0247	0.0194	0.0792	0.0243
26	0.0214	0.0222	0.0219	0.0223	0.0244	0.0222	0.0233	0.0214	0.0213	0.0222	0.0227	0.0214	0.0217	0.0213	0.0213	0.0235	0.0152
27	0.0199	0.0258	0.0102	0.0134	0.0125	0.0189	0.0192	0.0183	0.0202	0.0123	0.0176	0.0220	0.0070	0.0190	0.0186	0.0087	0.0066
28	0.0752	0.0767	0.0828	0.0773	0.0869	0.0751	0.0752	0.0752	0.0733	0.0734	0.0725	0.0752	0.0760	0.0752	0.0733	0.0614	0.0719
29	0.0352	0.0365	0.0365	0.0395	0.0376	0.0413	0.0421	0.0375	0.0354	0.0359	0.0461	0.0375	0.0306	0.0352	0.0354	0.0421	0.0464

continued on next page

Variance (continued from previous page)

No.	NB	BSE	FSS	BSEJ	TAN	SP-TAN	NBTree	LBR	AODE	MAPLMG	LWNB	APNB	IB	NB^{SR}	$AODE^{SR}$	Logistic	LibSVM
30	0.0436	0.0425	0.0457	0.0903	0.0689	0.0780	0.0949	0.0993	0.0398	0.0394	0.0401	0.0484	0.0436	0.0436	0.0397	0.0300	0.0161
31	0.0669	0.0669	0.0669	0.0669	0.0669	0.0669	0.0669	0.0669	0.0669	0.0669	0.0669	0.0669	0.0547	0.0669	0.0669	0.0654	0.0866
32	0.1929	0.1900	0.2626	0.1954	0.2120	0.1928	0.1929	0.1961	0.2004	0.2000	0.2049	0.2003	0.2008	0.1929	0.2004	0.1921	0.2355
33	0.0404	0.0485	0.0893	0.0505	0.0756	0.0441	0.0934	0.0408	0.0368	0.0371	0.0506	0.0404	0.0404	0.0417	0.0391	0.1303	0.0796
34	0.0185	0.0351	0.0418	0.0438	0.0449	0.0462	0.0453	0.0469	0.0299	0.0337	0.0509	0.0392	0.0347	0.0221	0.0298	0.0046	0.0256
35	0.0031	0.0005	0.0036	0.0006	0.0011	0.0004	0.0001	0.0006	0.0001	0.0001	0.0000	0.0021	0.0094	0.0026	0.0001	0.0015	0.0000
36	0.0816	0.0830	0.0706	0.0761	0.1290	0.0990	0.0763	0.0943	0.0807	0.0665	0.1073	0.0877	0.0816	0.0810	0.0806	0.1608	0.1539
37	0.0219	0.0266	0.0314	0.0258	0.0407	0.0251	0.0289	0.0219	0.0240	0.0248	0.0272	0.0229	0.0219	0.0237	0.0253	0.0279	0.0230
38	0.0077	0.0077	0.0116	0.0158	0.0129	0.0144	0.0248	0.0197	0.0087	0.0116	0.0214	0.0115	0.0085	0.0077	0.0087	0.0065	0.0015
39	0.0131	0.0177	0.0314	0.0187	0.0165	0.0182	0.0571	0.0366	0.0107	0.0109	0.0115	0.0144	0.0144	0.0133	0.0108	0.0385	0.0041
40	0.0155	0.0164	0.0143	0.0149	0.0154	0.0170	0.0148	0.0138	0.0102	0.0110	0.0111	0.0113	0.0184	0.0164	0.0104	0.0083	0.0084
41	0.0159	0.0233	0.0249	0.0289	0.0200	0.0210	0.0350	0.0310	0.0094	0.0099	0.0104	0.0200	0.0205	0.0159	0.0094	0.0106	0.0019
42	0.0779	0.0753	0.0724	0.0765	0.0762	0.0755	0.0773	0.0773	0.0747	0.0747	0.0800	0.0771	0.0683	0.0775	0.0751	0.0263	0.0408
43	0.0690	0.0593	0.0333	0.0730	0.1067	0.0868	0.0872	0.0688	0.0644	0.0648	0.0622	0.0596	0.0395	0.0694	0.0645	0.1752	0.0167
44	0.1575	0.1754	0.2215	0.1808	0.1781	0.1704	0.2329	0.1603	0.1553	0.1571	0.1841	0.1603	0.1575	0.1584	0.1575	0.3056	0.2003
45	0.0661	0.0660	0.1363	0.0702	0.1560	0.0696	0.1192	0.0661	0.0797	0.0811	0.0790	0.0666	0.0719	0.0661	0.0797	0.0653	0.1320
46	0.0264	0.0289	0.0331	0.0251	0.0248	0.0286	0.0396	0.0387	0.0214	0.0212	0.0223	0.0267	0.0268	0.0262	0.0204	0.0180	0.0172
47	0.0040	0.0040	0.0033	0.0050	0.0044	0.0046	0.0065	0.0056	0.0036	0.0033	0.0065	0.0044	0.0027	0.0041	0.0035	0.0052	0.0100
48	0.0303	0.0448	0.0508	0.0767	0.0398	0.0615	0.0784	0.0674	0.0363	0.0400	0.0655	0.0403	0.0390	0.0303	0.0384	0.0174	0.0465
49	0.0104	0.0217	0.0207	0.0226	0.0143	0.0232	0.0254	0.0208	0.0105	0.0103	0.0201	0.0201	0.0697	0.0104	0.0104	0.0104	0.0147
50	0.0841	0.0957	0.1245	0.0964	0.1170	0.0875	0.1234	0.0873	0.0865	0.0866	0.1271	0.0847	0.0872	0.0842	0.0966	0.1506	0.0760
51	0.0084	0.0169	0.0241	0.0164	0.0137	0.0132	0.0353	0.0234	0.0096	0.0098	0.0202	0.0094	0.0094	0.0087	0.0095	0.0165	0.0259
52	0.0087	0.0106	0.0178	0.0132	0.0218	0.0104	0.0092	0.0110	0.0103	0.0104	0.0325	0.0091	0.0104	0.0087	0.0103	0.0883	0.0499
53	0.0182	0.0181	0.0290	0.0182	0.0147	0.0179	0.0530	0.0215	0.0142	0.0141	0.0089	0.0183	0.0193	0.0179	0.0131	0.1613	0.0057
54	0.0407	0.0516	0.0704	0.1288	0.0898	0.0889	0.1287	0.1106	0.0509	0.0542	0.0494	0.0395	0.0386	0.0407	0.0509	0.0076	0.0008
55	0.1199	0.1248	0.1457	0.1350	0.1487	0.1311	0.1579	0.1391	0.1283	0.1298	0.1369	0.1230	0.1210	0.1238	0.1313	0.0704	0.0785
56	0.0045	0.0045	0.0045	0.0045	0.0045	0.0045	0.0045	0.0045	0.0045	0.0045	0.0045	0.0045	0.0043	0.0045	0.0045	0.0169	0.0292
57	0.2453	0.2171	0.2180	0.2481	0.2519	0.2372	0.2457	0.2499	0.2267	0.2152	0.1754	0.2489	0.2535	0.2393	0.2178	0.1637	0.0569
58	0.0247	0.0477	0.0573	0.0498	0.0632	0.0444	0.0870	0.0632	0.0385	0.0416	0.0790	0.0368	0.0247	0.0247	0.0394	0.0227	0.0267
59	0.0128	0.0171	0.0283	0.0150	0.0243	0.0131	0.0218	0.0128	0.0114	0.0113	0.0178	0.0128	0.0126	0.0137	0.0124	0.0190	0.0140
60	0.0363	0.0398	0.0680	0.0383	0.0450	0.0374	0.0475	0.0364	0.0346	0.0346	0.0320	0.0363	0.0363	0.0393	0.0368	0.0550	0.0395

Table A.4: RMSE (the data sets are in the number sequence of Table 3.4)

No.	NB	BSE	FSS	BSEJ	TAN	SP-TAN	NBTree	LBR	AODE	MAPLMG	LWNB	APNB	IB	NB^{SR}	$AODE^{SR}$	Logistic
1	0.4645	0.4548	0.4352	0.4487	0.4268	0.4527	0.4591	0.4507	0.4221	0.4216	0.4482	0.4649	0.4648	0.4640	0.4220	0.4176
2	0.3426	0.3136	0.3126	0.3120	0.3099	0.3185	0.3223	0.3138	0.3224	0.3137	0.3456	0.3265	0.3264	0.3417	0.3146	0.3207
3	0.1588	0.1593	0.1646	0.1595	0.1427	0.1520	0.1666	0.1558	0.1594	0.1584	0.1459	0.1572	0.1744	0.1506	0.1474	0.1856
4	0.1616	0.1617	0.1426	0.1624	0.1382	0.1598	0.1594	0.1603	0.1614	0.1614	0.1435	0.1597	0.2153	0.1385	0.1364	0.1551
5	0.3054	0.3007	0.2852	0.2920	0.2716	0.2879	0.2854	0.2927	0.2781	0.2757	0.2585	0.3055	0.3548	0.2975	0.2678	0.3329
6	0.3402	0.3473	0.3471	0.3419	0.3384	0.3399	0.3403	0.3401	0.3375	0.3377	0.3383	0.3410	0.3377	0.3402	0.3375	0.2124
7	0.1602	0.1637	0.1709	0.1662	0.2070	0.1617	0.1665	0.1602	0.1645	0.1640	0.1738	0.1602	0.1607	0.1602	0.1644	0.2805
8	0.2311	0.2311	0.2339	0.1957	0.2004	0.1871	0.1852	0.1971	0.2137	0.2023	0.1938	0.2279	0.2570	0.2311	0.2137	0.1643
9	0.3592	0.3570	0.3578	0.3183	0.3326	0.3355	0.3571	0.3290	0.3388	0.3357	0.3195	0.3586	0.3548	0.3568	0.3355	0.3361
10	0.3721	0.3645	0.3920	0.3739	0.4372	0.3745	0.3721	0.3721	0.3917	0.3862	0.3818	0.3721	0.3721	0.3721	0.3917	0.4744
11	0.4497	0.4444	0.4405	0.4465	0.4442	0.4487	0.4517	0.4494	0.4404	0.4397	0.4691	0.4505	0.2500	0.4497	0.4404	0.4419
12	0.3386	0.3385	0.3356	0.3383	0.3425	0.3387	0.3441	0.3375	0.3296	0.3253	0.3440	0.3388	0.3347	0.3291	0.3246	0.3494
13	0.4392	0.4336	0.4299	0.4364	0.4696	0.4357	0.4627	0.4387	0.4282	0.4216	0.4396	0.4396	0.4392	0.4331	0.4233	0.5049
14	0.0783	0.0782	0.0929	0.0782	0.0921	0.0778	0.1037	0.0783	0.0784	0.0783	0.0768	0.0787	0.0783	0.0779	0.0796	0.1042
15	0.4779	0.4784	0.4783	0.4784	0.4752	0.4775	0.4791	0.4782	0.4771	0.4772	0.4828	0.4889	0.5405	0.4779	0.4772	0.4365
16	0.4238	0.4238	0.4288	0.4288	0.4436	0.4279	0.4432	0.4254	0.4183	0.4187	0.4552	0.4242	0.4484	0.4234	0.4188	0.4214
17	0.3799	0.3799	0.3812	0.3791	0.3736	0.3774	0.3830	0.3789	0.3724	0.3722	0.3816	0.3809	0.4099	0.3797	0.3706	0.3981
18	0.4449	0.4428	0.4431	0.4444	0.4506	0.4494	0.4516	0.4456	0.4467	0.4463	0.4650	0.4509	0.4846	0.4449	0.4492	0.4414
19	0.3684	0.3702	0.3871	0.3719	0.3685	0.3699	0.3827	0.3683	0.3597	0.3592	0.3786	0.3693	0.3707	0.3683	0.3588	0.3763
20	0.3743	0.3768	0.3794	0.3747	0.3570	0.3747	0.3826	0.3745	0.3616	0.3612	0.3680	0.3758	0.3794	0.3766	0.3611	0.4635
21	0.3995	0.3909	0.3687	0.3928	0.4003	0.3973	0.4021	0.3927	0.3830	0.3806	0.4017	0.3978	0.3786	0.3994	0.3793	0.4610
22	0.3002	0.2785	0.2103	0.2777	0.2404	0.2830	0.2251	0.2367	0.2157	0.2045	0.2041	0.3011	0.2941	0.2996	0.2155	0.2898
23	0.3591	0.3659	0.3779	0.3631	0.3419	0.3599	0.3659	0.3595	0.3526	0.3523	0.3595	0.3591	0.3767	0.3591	0.3498	0.3675
24	0.0895	0.0814	0.0785	0.0723	0.0706	0.0722	0.0744	0.0759	0.0851	0.0815	0.0849	0.0883	0.0915	0.0945	0.0840	0.1158
25	0.3144	0.3124	0.2976	0.3107	0.2819	0.3139	0.3130	0.3124	0.2833	0.2822	0.2829	0.3145	0.3146	0.3202	0.2721	0.3990
26	0.1722	0.1798	0.1828	0.1772	0.1816	0.1736	0.1790	0.1722	0.1691	0.1696	0.1744	0.1722	0.1736	0.1724	0.1693	0.1773
27	0.3054	0.2772	0.2912	0.1826	0.2333	0.2252	0.1575	0.1915	0.2748	0.2362	0.1719	0.3052	0.2277	0.3012	0.2649	0.1544
28	0.3068	0.3101	0.3323	0.3117	0.3199	0.3065	0.3068	0.3068	0.3077	0.3079	0.3019	0.3068	0.3134	0.3068	0.3077	0.2992
29	0.1982	0.1987	0.1987	0.1997	0.2030	0.2000	0.2003	0.1988	0.1990	0.1991	0.2082	0.1995	0.2029	0.1982	0.1990	0.2023

continued on next page

RMSE (continued from previous page)

No.	NB	BSE	FSS	BSEJ	TAN	SP-TAN	NBTree	LBR	AODE	MAPLMG	LWNB	APNB	IB	NB^{SR}	$AODE^{SR}$	Logistic
30	0.1208	0.1189	0.1190	0.1037	0.0990	0.0938	0.1043	0.0981	0.0857	0.0825	0.0701	0.1198	0.1208	0.1207	0.0851	0.1132
31	0.4980	0.4980	0.4980	0.4980	0.4978	0.4980	0.4980	0.4980	0.4980	0.4980	0.4982	0.4980	0.6111	0.4980	0.4980	0.4664
32	0.5741	0.5728	0.5283	0.5727	0.5175	0.5738	0.5741	0.5736	0.5752	0.5748	0.5834	0.5756	0.5712	0.5741	0.5752	0.5967
33	0.2529	0.2534	0.2822	0.2496	0.2710	0.2518	0.2723	0.2528	0.2449	0.2452	0.2528	0.2530	0.2529	0.2580	0.2453	0.3478
34	0.4002	0.3712	0.3717	0.3554	0.3509	0.3554	0.3541	0.3497	0.3609	0.3546	0.3623	0.3836	0.3674	0.3948	0.3591	0.3840
35	0.1986	0.1091	0.0914	0.0353	0.0565	0.0305	0.0107	0.0265	0.0226	0.0186	0.0063	0.1417	0.1829	0.1221	0.0261	0.0359
36	0.0940	0.0912	0.0886	0.0891	0.0960	0.0849	0.0829	0.0851	0.0890	0.0822	0.0908	0.0936	0.0940	0.0907	0.0874	0.1069
37	0.1610	0.1686	0.1767	0.1682	0.1813	0.1632	0.1730	0.1610	0.1667	0.1668	0.1833	0.1629	0.1617	0.1592	0.1663	0.1789
38	0.1773	0.1773	0.1802	0.1466	0.1455	0.1484	0.1206	0.1385	0.1593	0.1488	0.1162	0.1744	0.1815	0.1773	0.1593	0.1468
39	0.1191	0.1167	0.1150	0.1069	0.0885	0.1088	0.1213	0.1046	0.0755	0.0755	0.0705	0.1189	0.1153	0.1190	0.0751	0.1095
40	0.1491	0.1253	0.1233	0.1162	0.1261	0.1220	0.1125	0.1122	0.1042	0.1035	0.1065	0.1360	0.1392	0.1482	0.1036	0.1054
41	0.1468	0.1414	0.1405	0.1038	0.0950	0.0783	0.0955	0.0892	0.0663	0.0649	0.0606	0.1385	0.1436	0.1468	0.0663	0.0857
42	0.4231	0.4210	0.4199	0.4210	0.4156	0.4210	0.4212	0.4219	0.4182	0.4174	0.4231	0.4235	0.4434	0.4225	0.4162	0.3996
43	0.4112	0.4063	0.3970	0.4145	0.4246	0.4152	0.4249	0.4115	0.4151	0.4157	0.4257	0.4103	0.4319	0.4112	0.4152	0.4841
44	0.1840	0.1842	0.1887	0.1851	0.1812	0.1838	0.1956	0.1842	0.1829	0.1828	0.1948	0.1842	0.1840	0.1837	0.1827	0.2345
45	0.3207	0.3209	0.3906	0.3273	0.4463	0.3241	0.3933	0.3207	0.3483	0.3488	0.3444	0.3216	0.3322	0.3207	0.3483	0.3542
46	0.1551	0.1390	0.1278	0.1212	0.1122	0.1243	0.1289	0.1316	0.1169	0.1153	0.1119	0.1535	0.1543	0.1502	0.1134	0.1075
47	0.1637	0.1542	0.1500	0.1521	0.1532	0.1536	0.1471	0.1543	0.1559	0.1486	0.1509	0.1619	0.1495	0.1602	0.1523	0.1671
48	0.3995	0.3996	0.4006	0.3474	0.3570	0.3595	0.3457	0.3437	0.3582	0.3550	0.3313	0.4036	0.3967	0.3995	0.3566	0.4211
49	0.3867	0.3818	0.3750	0.3742	0.3649	0.3730	0.3760	0.3745	0.3637	0.3634	0.3776	0.3931	0.4497	0.3848	0.3624	0.3626
50	0.4636	0.4656	0.4632	0.4625	0.4489	0.4629	0.4662	0.4637	0.4562	0.4560	0.4539	0.4638	0.4674	0.4629	0.4445	0.5469
51	0.3015	0.2669	0.2521	0.2535	0.2628	0.2773	0.2563	0.2420	0.2445	0.2308	0.2172	0.3015	0.2866	0.3004	0.2439	0.2462
52	0.1560	0.1544	0.1678	0.1576	0.1735	0.1549	0.1565	0.1560	0.1504	0.1501	0.1882	0.1561	0.1505	0.1560	0.1504	0.2946
53	0.1321	0.1297	0.1227	0.1301	0.0838	0.1307	0.1414	0.1263	0.0980	0.0966	0.0762	0.1320	0.1350	0.1312	0.0927	0.2811
54	0.4334	0.4325	0.4399	0.3844	0.4196	0.4292	0.3762	0.3999	0.4044	0.4029	0.2818	0.4366	0.4551	0.4334	0.4044	0.1381
55	0.3985	0.3938	0.3665	0.3695	0.3389	0.3747	0.3646	0.3533	0.3323	0.3304	0.3395	0.3987	0.3964	0.3862	0.3286	0.2835
56	0.3275	0.3275	0.3275	0.3275	0.3276	0.3275	0.3275	0.3275	0.3275	0.3275	0.3276	0.3275	0.3351	0.3275	0.3275	0.3277
57	0.2409	0.2306	0.2331	0.2257	0.2298	0.2154	0.2223	0.2225	0.2164	0.2109	0.1828	0.2418	0.2410	0.2387	0.2121	0.2235
58	0.3359	0.3173	0.3157	0.2970	0.2951	0.3051	0.3087	0.2935	0.2703	0.2684	0.3008	0.2955	0.3359	0.3359	0.2672	0.2554
59	0.1208	0.1322	0.1503	0.1271	0.1483	0.1223	0.1373	0.1208	0.1231	0.1230	0.1293	0.1208	0.1215	0.1220	0.1255	0.1575
60	0.1330	0.1333	0.1545	0.1329	0.1435	0.1323	0.1363	0.1332	0.1260	0.1259	0.1208	0.1330	0.1330	0.1360	0.1282	0.1483

References

Acid, S., Campos, L. M. D. and Castellano, J. G. (2005). Learning Bayesian network classifiers: Searching in a space of partially directed acyclic graphs, *Machine Learning* **59**(3): 213–235.

Almuallim, H. and Dietterich, T. G. (1991). Learning with many irrelevant features, *Proceedings of the Ninth National Conference on Artificial Intelligence*, Vol. 2, AAAI Press, pp. 547–552.

Amaldi, E. and Kann, V. (1998). On the approximability of minimizing nonzero variables or unsatisfied relations in linear systems, *Theoretical Computer Science* **209**(1–2): 237–260.

Atkeson, C., Moore, A. and Schaal, S. (1997). Locally weighted learning, *Artificial Intelligence Review* **11**(1-5): 11–73.

Avriel, M. (2003). *Nonlinear Programming: Analysis and Methods*, Courier Dover Publications.

Bachmann, P. (1894). *Die Analytische Zahlentheorie*, Leipzig: B. G. Teubner.

Blum, A. and Langley, P. (1997). Selection of relevant features and examples in machine learning, *Artificial Intelligence* **97**(1-2): 245–271.

Bluman, A. G. (1997). *Elementary Statistics: A Step by Step Approach*, Irwin Professional Publishing.

Boser, B. E., Guyon, I. and Vapnik, V. (1992). A training algorithm for optimal margin classifiers, *Proceedings of the Fifth Annual Workshop on Computational Learing Theory*, ACM Press, pp. 144–152.

Brain, D. and Webb, G. I. (2002). The need for low bias algorithms in classification learning from large data sets, *Proceedings of the Sixteenth European Conference on Principles of Data Mining and Knowledge Discovery*, Berlin:Springer-Verlag, pp. 62–73.

Breiman, L. (1996). Bias, variance, and arcing classifiers, *Technical Report 460*, Statistics Department, University of California, Berkeley, CA.

Cardie, C. (1993). Using decision trees to improve case–based learning, *Proceedings of the Tenth International Conference on Machine Learning*, Morgan Kaufmann, pp. 25–32.

Caruana, R. and Freitag, D. (1994). Greedy attribute selection, *Proceedings of the Eleventh International Conference on Machine Learning*, pp. 28–36.

Castillo, G. and Gama, J. (2006). An adaptive prequential learning framework for bayesian network classifiers., *Proceedings of the Seventeenth European Conference on Machine Learning*, Springer Berlin / Heidelberg, pp. 67–78.

Catlett, J. (1991). On changing continuous attributes into ordered discrete attributes, *Proceedings of the European working session on learning on Machine learning*, Springer-Verlag, pp. 164–178.

Cerquides, J. and de Màntaras, R. L. (1997). Proposal and empirical comparison of a parallelizable distance-based discretization method, *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining*, pp. 139–142.

Cerquides, J. and Mántaras, R. L. D. (2005a). Robust Bayesian linear classifier ensembles, *Proceedings of the Sixteenth European Conference on Machine Learning*, pp. 70–81.

Cerquides, J. and Mántaras, R. L. D. (2005b). TAN classifiers based on decomposable distributions, *Machine Learning* **59**(3): 323–354.

Cestnik, B. (1990). Estimating probabilities: A crucial task in machine learning, *Proceedings of the Ninth European Conference on Artificial Intelligence*, London: Pitman, pp. 147–149.

Chow, C. and Liu, C. (1968). Approximating discrete probability distributions with dependence trees, *IEEE Transactions on Information Theory* **14**: 462–467.

Cleveland, W. S. (1979). Robust locally weighted regression and smoothing scatterplots, *Journal of the American Statistical Association* **74**: 859–836.

Cormen, T. H., Leiserson, C. E., Rivest, R. L. and Stein, C. (2001). *Introduction to Algorithms*, MIT Press and McGraw-Hill.

Cortes, C. and Vapnik, V. (1995). Support-vector networks, *Machine Learning* **20**(3): 273–297.

Das, S. (2001). Filters, wrappers and a boosting-based hybrid for feature selection, *Proceedings of the Eighteenth International Conference on Machine Learning*, Morgan Kaufmann, pp. 74–81.

Demšar, J. (2006). Statistical comparisons of classifiers over multiple data sets, *Journal of Machine Learning Research* **7**: 1–30.

Doak, J. (1992). An evaluation of feature selection methods and their application to computer security, *Technical report*, Davis, CA: University of California, Department of Computer Science.

Domingos, P. (2000). A unified bias-variance decomposition and its applications, *Proceedings of the Seventeenth International Conference on Machine Learning*, Morgan Kaufmann, San Francisco, CA, pp. 231–238.

Domingos, P. and Pazzani, M. J. (1996). Beyond independence: Conditions for the optimality of the simple Bayesian classifier, *Proceedings of the Thirteenth International Conference on Machine Learning*, Morgan Kaufmann, pp. 105–112.

Domingos, P. and Pazzani, M. J. (1997). On the optimality of the simple Bayesian classifier under zero-one loss, *Machine Learning* **29**(2-3): 103–130.

Dougherty, J., Kohavi, R. and Sahami, M. (1995). Supervised and unsupervised discretization of continuous features, *Proceedings of the Twelfth International Conference on Machine Learning*, pp. 194–202.

Duda, R. O. and Hart, P. E. (1973). *Pattern Classification and Scene Analysis*, John Wiley and Sons, New York.

Fayyad, U. M. and Irani, K. B. (1993). Multi-interval discretization of continuous-valued attributes for classification learning, *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence*, Morgan Kaufmann, pp. 1022–1029.

Forman, G. (2003). An extensive empirical study of feature selection metrics for text classification, *Journal of Machine Learning Research* **3**: 1289–1305.

Frank, E., Hall, M. and Pfahringer, B. (2003). Locally weighted naive Bayes, *Proceedings of the Nineteenth Conference in Uncertainty in Artificial Intelligence*, Morgan Kaufmann, pp. 249–256.

Frank, E. and Witten, I. H. (1999). Making better use of global discretization, *Proceedings of the Sixteenth International Conference on Machine Learning*, Morgan Kaufmann, pp. 115–123.

Friedman, J. H. (1997). On bias, variance, 0/1-loss, and the curse-of-dimensionality, *Data Mining and Knowledge Discovery* **1**(1): 55–77.

Friedman, M. (1937). The use of ranks to avoid the assumption of normality implicit in the analysis of variance, *the American Statistical Association* **32**(200): 675–701.

Friedman, M. (1940). A comparison of alternative tests of significance for the problem of m rankings, *the American Statistical Association* **11**(1): 86–92.

Friedman, N., Geiger, D. and Goldszmidt, M. (1997). Bayesian network classifiers, *Machine Learning* **29**(2): 131–163.

Gama, J. (2003). Iterative Bayes, *Theoretical Computer Science* **292**(2): 417–430.

Geurts, P. (2002). *Contributions to Decision Tree Induction: Bias/Variance Tradeoff and Time Series Classification*, PhD thesis, University of Liège, Belgium. `http://www.montefiore.ulg.ac.be/services/stochastic/pubs/2002/Geu02`

Greiner, R., Su, X., Shen, B. and Zhou, W. (2005). Structural extensions to logistic regression: Discriminative parameter learning of belief net classifiers, *Machine Learning* **59**(3): 297–322.

Grünwald, P., Pitt, M. A. and Myung, I. J. (2005). *Advances in Minimum Description Length: Theory and Applications*, M.I.T. Press.

Guyon, I. and Elisseeff, A. (2003). An introduction to variable and feature selection, *Journal of Machine Learning Research* **3**: 1157–1182.

Hall, M. A. (2000). Correlation-based feature selection for discrete and numeric class machine learning, *Proceedings of the Seventeenth International Conference on Machine Learning*, Morgan Kaufmann, San Francisco, CA, pp. 359–366.

Hastie, T., Tibshirani, R. and Friedman, J. (2001). *Elements of Statistical Learning: Data Mining, Inference and Prediction*, Springer, New York.

Hearst, M. A., Schölkopf, B., Dumais, S., Osuna, E. and Platt, J. (1998). Trends & controversies: Support vector machines, *IEEE Intelligent Systems* **13**(4): 18–28.

Heskes, T. (1998). Bias/variance decompositions for likelihood-based estimators, *Neural Computation* **10**(6): 1425–1433.

Hilario, M. and Kalousis, A. (1999). Characterizing learning models and algorithms for classification, *Technical Report UNIGE-AI-TR-99-1*, University of Geneva.

Holte, R. C., Acker, L. and Porter, B. W. (1989). Concept learning and the problem of small disjuncts, *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, San Mateo, CA: Morgan Kaufmann, pp. 813–818.

Hope, L. R. and Korb, K. B. (2004). A Bayesian metric for evaluating machine learning algorithms, *Proceedings of the Seventeenth Australian Joint Conference on Advances in Artificial Intelligence*, Springer-Verlag, pp. 991–997.

Hsu, C.-N., Huang, H.-J. and Wong, T.-T. (2000). Why discretization works for naive Bayesian classifiers, *Proceedings of the Seventeenth International Conference on Machine Learning*, Morgan Kaufmann, pp. 399–406.

Hsu, C.-N., Huang, H.-J. and Wong, T.-T. (2003). Implications of the dirichlet assumption for discretization of continuous variables in naive Bayesian classifiers, *Machine Learning* **53**(3): 235–263.

Hsu, C.-W., Chang, C.-C. and Lin, C.-J. (2007). A practical guide to support vector classification.
`http://www.csie.ntu.edu.tw/ cjlin/papers/guide/guide.pdf`

Hsu, C.-W. and Lin, C.-J. (2002). A comparison of methods for multi-class support vector machines, *IEEE Transactions on Neural Networks* **13**(2): 415–425.

Iman, R. L. and Davenport, J. M. (1980). Approximations of the critical region of the Friedman statistic, *Communications in Statistics* pp. 571–595.

Jakulin, A. (2005). *Machine Learning Based on Attribute Interactions*, PhD thesis, University of Ljubljana.

James, G. M. (2003). Variance and bias for general loss functions, *Machine Learning* **51**(2): 115–135.

Keogh, E. J. and Pazzani, M. J. (1999). Learning augmented Bayesian classifers: A comparison of distribution-based and classification-based approaches, *Proceedings of the International Workshop on Artificial Intelligence and Statistics*, pp. 225–230.

Kerber, R. (1992). Chimerge: Discretization of numeric attributes, *Proceedings of the Tenth National Conference on Artificial Intelligence*, The AAAI Press, pp. 123–128.

Kira, K. and Rendell, L. A. (1992). The feature selection problem: Traditional methods and a new algorithm, *Proceedings of the Ninth National Conference on Artificial Intelligence*, pp. 129–134.

Kittler, J. (1986). Feature selection and extraction, *in* T. Y. Young and K.-S. Fu (eds), *Handbook of Pattern Recognition and Image Processing*, Academic Press, New York.

Kohavi, R. (1995). *Wrappers for Performance Enhancement and Oblivious Decision Graphs*, PhD thesis, Department of Computer Science, Stanford University, CA.

Kohavi, R. (1996). Scaling up the accuracy of naive-Bayes classifiers: a decision-tree hybrid, *Proceedings of the Second ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 202–207.

Kohavi, R. and John, G. H. (1997). Wrappers for feature subset selection, *Artificial Intelligence* **97**(1-2): 273–324.

Kohavi, R. and Wolpert, D. (1996). Bias plus variance decomposition for zero-one loss functions, *Proceedings of the Thirteenth International Conference on Machine Learning*, San Francisco: Morgan Kaufmann, pp. 275–283.

Koller, D. and Sahami, M. (1996). Toward optimal feature selection, *Proceedings of the Thirteenth International Conference on Machine Learning*, pp. 284–292.

Kong, E. B. and Dietterich, T. G. (1995). Error-correcting output coding corrects bias and variance, *Proceedings of the Twelfth International Conference on Machine Learning*, pp. 313–321.

Kononenko, I. (1990). Comparison of inductive and naive Bayesian learning approaches to automatic knowledge acquisition, *in* B. Wielinga, J. Boose, B.Gaines, G. Schreiber and M. van Someren (eds), *Current Trends in Knowledge Acquisition*, Amsterdam: IOS Press.

Kononenko, I. (1991). Semi-naive Bayesian classifier, *Proceedings of the Sixth European Working Session on Machine learning*, Berlin: Springer-Verlag, pp. 206–219.

Langley, P. (1993). Induction of recursive Bayesian classifiers, *Proceedings of the 1993 European Conference on Machine Learning*, Berlin: Springer-Verlag, pp. 153–164.

Langley, P. (1994). Selection of relevant features in machine learning, *Proceedings of AAAI Fall Symposium on Relevance*, pp. 140–144.

Langley, P., Iba, W. and Thompson, K. (1992). An analysis of Bayesian classifiers, *Proceedings of the Tenth National Conference on Artificial Intelligence*, AAAI Press and MIT Press, pp. 223–228.

Langley, P. and Sage, S. (1994). Induction of selective Bayesian classifiers, *Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence*, Morgan Kaufmann, pp. 399–406.

Langseth, H. and Nielsen, T. D. (2006). Classification using hierarchical naive Bayes models, *Machine Learning* **63**(2): 135 – 159.

Lewis, D. D. (1998). Naive Bayes at forty: The independence assumption in information retrieval, *Proceedings of the Tenth European Conference on Machine Learning*, Springer, Berlin, pp. 4–15.

Liu, H., Hussain, F., Tan, C. L. and Dash, M. (2002). Discretization: An enabling technique, *Data Mining and Knowledge Discovery* **6**(4): 393–423.

Liu, H. and Yu, L. (2005). Toward integrating feature selection algorithms for classification and clustering, *IEEE Transactions on Knowledge and Data Engineering* **17**(4): 491–502.

Loader, C. (1999). *Local Regression and Likelihood*, Springer.

Mclachlan, G. J. (1992). *Discriminant Analysis and Statistical Pattern Recognition*, John Wiley & Sons, New York.

Mitchell, T. (1997). *Machine Learning*, McGraw Hill.

Mitchell, T. (2005). *Generative and Discriminative Classifiers: Naive Bayes and Logistic Regression*, chapter 1, pp. 1–17.
`http://www.cs.cmu.edu/%7Etom/NewChapters.html`

Modrzejewski, M. (1993). Feature selection using rough sets theory, *Proceedings of the Sixth European Conference on Machine Learning*, pp. 213–226.

Mucciardi, A. N. and Gose, E. E. (1971). A comparison of seven techniques for choosing subsets of pattern recognition properties, *IEEE Transactions on Computers* **20**(9): 1023–1031.

Narendra, P. and Fukunaga, K. (1977). A branch and bound algorithm for feature selection, *IEEE Transactions on Computers* **26**(9): 917–922.

Newman, D., Hettich, S., Blake, C. and Merz, C. (1998). UCI repository of machine learning databases. Irvine, CA: University of California, Department of Information and Computer Science.
`http://www.ics.uci.edu/~mlearn/MLRepository.html`

Ng, A. Y. and Jordan, M. I. (2001). On discriminative vs. generative classifiers: A comparison of logistic regression and naive Bayes, *Advances in Neural Information Processing Systems*, MIT Press, pp. 841–848.

Oliveira, A. L. and Sangiovanni-Vincentelli, A. L. (1992). Constructive induction using a non-greedy strategy for feature selection, *Proceedings of the Ninth International Workshop on Machine Learning*, Morgan Kaufmann, pp. 355–360.

Pazzani, M. J. (1996). Constructive induction of Cartesian product attributes, *ISIS: Information, Statistics and Induction in Science* pp. 66–77.

Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*, Morgan Kaufmann.

Pearl, J. and Russell, S. (2000). Bayesian networks, *Technical Report R-277*, University of California.

Pedregal, P. (2004). *Introduction to Optimization. Number 46 in Texts in Applied Mathematics*, Springer.

Platt, J. C. (1999). Probabilistic outputs for support vector machines and comparison to regularized likelihood methods, *Advances in Large Margin Classifiers*, MIT Press.

Rish, I. (2001). An empirical study of the naive Bayes classifier, *Proceedings of IJCAI-01 workshop on Empirical Methods in AI*, pp. 41–46.

Rissanen, J. (1978). Modeling by shortest data description, *Automatica* **14**: 465–471.

Roos, T., Wettig, H., Grünwald, P., Myllymäki, P. and Tirri, H. (2005). On discriminative Bayesian network classifiers and logistic regression, *Machine Learning* **59**(3): 267–296.

Rubinstein, Y. D. and Hastie, T. (1997). Discriminative vs informative learning, *Proceedings of the Third International Conference Knowledge Discovery and Data Mining*, AAAI Press, pp. 49–53.

Sahami, M. (1996). Learning limited dependence Bayesian classifiers, *Proceedings of the Second International Conference on Knowledge Discovery in Databases*, Menlo Park, CA: AAAI Press, pp. 334–338.

Schlimmer, J. C. (1993). Efficiently inducing determinations: A complete and systematic search algorithm that uses optimal pruning, *Proceedings of the Tenth International Conference on Machine Learning*, pp. 284–290.

Sheinvald, J., Dom, B. and Niblack, W. (1990). A modelling approach to feature selection, *Proceedings of the Tenth International Conference on Pattern Recognition*, pp. 35–539.

Singh, M. and Provan, G. M. (1996). Efficient learning of selective Bayesian network classifiers, *Proceedings of the Thirteenth International Conference on Machine Learning*, Morgan Kaufmann, pp. 453–461.

Sipser, M. (1997). *Introduction to the Theory of Computation*, PWS Publishing.

Thomson ISI (2008). Web of science.
`http://scientific.thomson.com/products/wos/`

Webb, G. I. (2000). Multiboosting: A technique for combining boosting and wagging, *Machine Learning* **40**(2): 159–196.

Webb, G. I. (2001). Candidate elimination criteria for lazy Bayesian rules, *Proceedings of the Fourteenth Australian Joint Conference on Artificial Intelligence*, Vol. 2256, Berlin:Springer, pp. 545–556.

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

Webb, G. I. and Pazzani, M. J. (1998). Adjusted probability naive Bayesian induction, *Proceedings of the Eleventh Australian Joint Conference on Artificial Intelligence*, Berlin:Springer, pp. 285–295.

Witten, I. H. and Frank, E. (2005). *Data Mining: Practical Machine Learning Tools and Techniques*, Morgan Kaufmann.

Wu, T.-F., Lin, C.-J. and Weng, R. C. (2004). Probability estimates for multi-class classification by pairwise coupling, *Journal of Machine Learning Research* **5**: 975–1005.

Xie, Z., Hsu, W., Liu, Z. and Lee, M. L. (2002). Snnb: A selective neighborhood based naive Bayes for lazy learning, *Advances in Knowledge Discovery and Data Mining, Proceedings of the Pacific-Asia Conference*, Berlin:Springer, pp. 104–114.

Xing, E. P., Jordan, M. I. and Karp, R. M. (2001). Feature selection for high-dimensional genomic microarray data, *Proceedings of the Eighteenth International Conference on Machine Learning*, Morgan Kaufmann, pp. 601–608.

Yang, Y. (2003). *Discretization for Naive-Bayes Learning*, PhD thesis, School of Computer Science and Software Engineering, Monash University.

Yang, Y., Webb, G., Cerquides, J., Korb, K., Boughton, J. and Ting, K. M. (2006). To select or to weigh: A comparative study of model selection and model weighing for SPODE ensembles, *Proceedings of the Seventeenth European Conference on Machine Learning*, Springer, pp. 533–544.

Zadrozny, B. and Elkan, C. (2001). Obtaining calibrated probability estimates from decision trees and naive Bayesian classifiers, *Proceedings of the Eighteenth International Conference on Machine Learning*, Morgan Kaufmann, San Francisco, CA, pp. 609–616.

Zhang, H., Jiang, L. and Su, J. (2005). Hidden naive Bayes, *Proceedings of the Twentieth National Conference on Artificial Intelligence*, AAAI Press, pp. 919–924.

Zheng, F. and Webb, G. I. (2005). A comparative study of semi-naive Bayes methods in classification learning, *Proceedings of the Fourth Australasian Data Mining Conference*, pp. 141–156.

Zheng, F. and Webb, G. I. (2006). Efficient lazy elimination for averaged-one dependence estimators, *Proceedings of the Twenty-third International Conference on Machine Learning*, ACM Press, pp. 1113–1120.

Zheng, F. and Webb, G. I. (2007). Finding the right family: Parent and child selection for averaged one-dependence estimators, *Proceedings of the Eighteenth European Conference on Machine Learning*, Springer Berlin / Heidelberg, pp. 490–501.

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

Zheng, Z., Webb, G. I. and Ting, K. M. (1999). Lazy Bayesian rules: A lazy semi-naive Bayesian learning technique competitive to boosting decision trees, *Proceedings of the Sixteenth International Conference on Machine Learning*, Morgan Kaufmann, pp. 493–502.