2001

Beyond boosting: detecting noisy and inverse observations

Virginia Wheway
University of Wollongong

Beyond Boosting - Detecting Noisy and Inverse Observations

A thesis submitted in fulfillment of the

requirements for the award of the degree of

Doctor of Philosophy

from

The University of Wollongong

by

Virginia Wheway, BMath.(Hons)

School of Mathematics and Applied Statistics

School of Information Technology and Computer Science

2001

This thesis is submitted to the University of Wollongong, and has not been submitted for a degree to any other University or Institution.

Virginia Whéway

November, 2001

Acknowledgments

The majority of this reseach was funded through a UPA scholarship from the University of Wollongong. Earlier sections were completed through a School of Computer Science and Engineering scholarship at the University of New South Wales.

Firstly, I would like to thank my versatile and patient supervisors, Pam Davy and John Fulcher. To Pam for her availability, persistence and attention to detail. To John for his 'big picture' approach and contributing significantly in keeping my spirits above zero. Thank you also to my friends and colleagues at The University of Wollongong who welcomed me back with open arms (and scholarships!).
To my supportive family who knew when not to ask about the "T" word and were always interested in my journey rather than my destination.

There have been many angels cross my path during the past 5 years. Too many to mention individually but to all who have listened, danced, and adventured with me, you have contributed to this Thesis by refreshing me for another day in a rich life which just happened to include PhD study.

And finally to Douglas, my newfound best friend and kindred spirit. Thank

you for always being 'sensational' and encouraging me to do likewise.

As I close this Chapter of my life, I look forward, with new energy and a heightened awareness of the wonderful opportunites that life can present.

Table of Contents

List of Tables

List of Figures

Abstract

Noisy data is inherent in many real-life and industrial modelling situations. If prior knowledge of such data were available, it would be a simple process to remove or account for noise and improve model robustness. Unfortunately, in the majority of learning situations, the presence of underlying noise is suspected but difficult to detect.

Ensemble classification techniques such as *bagging*, [4] ,*boosting* [20] and *arcing* algorithms [5] have received much attention in the recent literature. Such techniques have been shown to lead to reduced classification error on unseen cases, and this Thesis demonstrates that they may also be employed as noise detectors. Recently defined diagnostics such as *edge* and *margin* [6, 20, 41] have been used to explain the improvements made in generalisation error when ensemble classifiers are built. The distributions of these measures are key in the noise detection process introduced in this research.

This Thesis presents some empirical and theoretical results on edge distributions that confirm existing theories on boosting's tendency to 'balance' error rates. The results are then extended to introduce a methodology whereby boosting may be used to identify noise in training data by examining the changes in edge and

margin distributions as boosting proceeds.

Further enrichment can be made by detecting clusters of observations behaving differently to the main 'core' of data, as opposed to detecting single, unique noisy observations. These clusters may form boundaries, and in extreme cases form inverse models, which undetected lead to overestimated generalisation errors. Using edge distributions to perform this task is trivial in comparison to the significant effort required to undertake this task in large multi-dimensional datasets using visual methods or sorting algorithms. Partitioning datasets according to clusters leads to significant improvements on generalisation error for each partition, along with the benefit of simpler classifiers on each partition.

However, this process requires a new technique for estimating generalisation error as classifiers are trained on subsets of the original dataset and generalisation error is not representative. A classifier trained and tested using a biased subset of the data will be overly optimistic and give no indication of the proportion of data for which this error estimate applies.

This notion leads to the key result of an analyst being able to partition generalisation error into components pertaining to incoming data noise and model noise. Depending on the magnitude of each component, analysts can directly concentrate on the process step having the majority contribution to the generalisation error.

Chapter 1

Background : Knowledge Discovery in Databases and Data Mining

1.1 Background to Data Mining

The emerging field of data mining is receiving much attention in both the machine learning and statistical communities, both of which share common goals in:

1. extracting meaning from data [25]

2. learning from data [17]

Data mining is most often considered to be a field of research which deals with vast databases containing large numbers of observations and variables. Research into data mining aims to develop tools by which to extract meaningful information from such databases. The field has been defined as "computer automated exploratory data analysis of (usually) large complex data sets" [22] and "any algorithm that enumerates patterns from, or fits models to, data" [19]. Data mining is a single, yet central step in the Knowledge Discovery in Databases (KDD)

process [19]. KDD refers to the process of extracting useful information from data and encompasses data warehousing, targeted data selection, pre-processing, transformation, model selection, evaluation and interpretation. Data mining is the process step in which pattern extraction takes place. Interest has been propelled by recent advances in computing technology and the ability to collect and store vast amounts of data.

The need for analysis tools in a world which is becoming increasingly data overloaded is paramount. Many industries now have the capacity to collect and store massive amounts of data, with data being accumulated at an extraordinary rate. As databases move from megabytes to gigabytes to terabytes, demand for tools which can quickly exploit information available in existing databases is increasing. In making better use of existing data, companies can gain significant competitive advantage when data is converted into information for decision making. For example, in the manufacturing sector, databases may contain untapped optimisation opportunities and keys to improving processes.

The concept of data mining has had negative connotations in statistical literature [19, 30] and has been referred to as data dredging and data archaeology. Such criticisms may have arisen from the notion of naive data mining, whereby data is 'crunched' through a black box software tool. Blind searches with poor input data inevitably lead to misleading and spurious conclusions. Like all data analysis methods, data mining is open to abuse where, under continual probing, false patterns may be revealed. However, as ideas from the two fields merge,

negative suggestions are being addressed and dispelled. Huber [27] claims that advancement of data mining knowledge by the statistics, database and artificial intelligence (AI) communities are now showing signs of union. It is now widely recognised that advances in data mining will come about with the merging of ideas from the computer science and statistics disciplines. There needs to be a growth in methodologies which have come about via the pooling of skills in computer science and statistics. It is perhaps with this is mind that the statistician Peter Huber , quoted in [18] makes the following skeptical comment:

> I do not think that I am doing injustice to the present situation by contending that data mining is still a nearly empty hull, held in place by hot air, and serving as place-holder for more substantive contents to come.

Fayyad [18] argues as follows that there may be a misunderstanding of the aims of data mining.

> Data mining is not about automating data analysis. Data mining is about making analysis more convenient, scaling analysis algorithms to large databases, and providing data owners with easy-to-use tools to help them navigate,visualize, summarize, and model data.
>
> I personally look forward to the proper balance that will emerge from the mixing of computational algorithm-oriented approaches characterizing the database and computer science communities with the powerful mathematical theories and methods for estimation developed in

statistics [18].

Data mining is not considered a subset of statistics due to the scalability issues and statistics methodologies not having been designed to deal with massive datasets. Dietterich [12] lists scalability of existing algorithms as an important direction in current machine learning research. However, the statistical literature has a wealth of technical procedures and results to offer any attempt to extract information from data. Much commercial data mining activity uses relatively conventional statistical methods. Yet, with the increase in data collection ability comes problems in the scalability of existing data analysis methods and algorithms. Complex structure is a recurring issue in massive datasets, often caused by the nature of data collection. Such structure may cause traditional statistical methods of data summary and subsampling to fail.

To date, the statistics literature has been primarily focused on hypothesis verification as the primary mode of analysis and many statistical methods do not provide searches over models and parameters.

Many analyses require significant data manipulation, often merging several datasets and deciding on a consistent format. Required is a data analysis strategy able to step from massaging the initial dataset to a targetted search for patterns. Along with this need arises the issue of computational efficiency and interpretability versus statistical consistency and model accuracy [19]. With the size and complexity of such analysis arises a complexity versus accuracy trade-off. Effective means for dimensionality reduction should form part of the pre-processing steps

for data analysis.

Authors from the statistics, database and artificial intelligence communities stress the need for good data [25, 27, 38]. Key understandings are based on the notion that resulting inferences are only as good as the data allow and mixing highly accurate methodologies with unreliable data will make the resulting database only as good as the inaccurate data [30]. Data can be collected via designed experiments or by exploiting observational studies where little control is exercised over the data generating process. Branches of statistics such as observational studies and experimental design are very applicable in the initial stages of data mining. The massive datasets usually associated with data mining rarely come from designed experiments and are often collected as a matter of course as part of an industrial, financial or medical process, to name a few. Some applications, however, are specific to their individual fields. Yet observational data analysis is often teeming with statistical pitfalls and a good measure of healthy skepticism by the analyst will go a long way in avoiding spurious information and models from data.

Glymour et. al. [25] list three themes of modern statistics that are of key importance to data miners. These are:

1. clarity of goals

2. appropriate reliability assessment

3. sufficient accounting for sources of uncertainty

An analyst should be able to pose appropriate questions before undertaking analysis. Directed searches, as opposed to blind searches, can reduce computational complexity significantly. However, the formulation of goals requires human cognition and many authors have highlighted the human interface necessary for successful data mining. A debate appears to be continuing as to the need for human interaction versus shielding the users from underlying algorithms. Tukey [44], a defender of the exploratory data analysis school, expressed the desire for the 'information extraction' process to be human-centred, arguing that theory should adapt to the scientific method. A human's ability to inspect vast databases is limited not only by computational limitations but simply by the human optical system [45].

If experimental design is so important, why the push for data mining - does statistics now have to "loosen" its assumptions? Is data collection driven by an understanding of what is worth collecting? Methods applicable to Exploratory Data Analysis (EDA) have a large overlap with the methods and tools of data mining. There is now wide agreement that while observational databases have their uses e.g. in drawing attention to side effects, they are unreliable and potentially misleading sources of information for decision making. Causal inference is a difficult and challenging area in which to work. A more exploratory form of data mining applies a search process to a dataset and looks for interesting associations. Data may have been collected for a different purpose, to answer different questions. There is an expectation that further searching may provide interesting

and potentially useful information. Data collection is often undertaken to answer a particular question, and there is an expectation that something "more interesting" may lie in the database and the ability to extract this bonus information is paramount. Statisticians may deem interesting or unusual features outliers and this has been the focus of much research in data mining.

With the explosion in interest comes an explosion in commercially available data mining tools. A study has been made on the leading data mining tools by [16]. Tools such as neural networks, decision trees and association rules appear in the document. Data mining packages are marketed as time and cost saving methods allowing one to make better use of the data one already has. Marketing strategies for such software claim that one will be disadvantaged by not exploiting information available and remaining less informed than competitors who are making use of data mining technologies.

Collaboration between the statistics and machine learning communities can also serve to advance existing methodologies. Researchers from both fields share common goals in extracting meaning and learning from data. The vigorous growth in new methodologies and a glut of data provides abundant opportunities for research and application development.

1.2 The General Learning Problem

Concept learning or concept formation is the process by which

> information is generalized into categories for efficient storage and communication as a basis for further exploration in the environment [10].

Representation may differ, but all learners have the common goal of accurately predicting unseen future values given input attributes for the new data. A learner is trained using data from a *training* set and future performance is measured on accuracy from unseen instances in a *test* set. In the statistics literature, performance on a test set is often referred to as out-of-sample performance.

A *training* set consists of attribute-value pairs of the form, $\{(\mathbf{x}_i, y_i), i = 1 \ldots, N\}$. The y_i's are measured responses to the input vectors $\mathbf{x}_i$. Using this training set, procedures capable of forming hypotheses $h(\mathbf{x})$ are sought. Such hypotheses are estimates of the unknown relationship $y = f(\mathbf{x})$ and allow future values of y_i to be predicted given the input values of $\mathbf{x}_i$.

Many hypotheses predict well on the training set but yield a significant drop in accuracy over unseen cases. The accuracy of the learner on unseen cases gives a truer indication of the learner's future performance. Error measured on unseen cases (*test* set) is referred to in the literature as *generalisation error* [37], with the best learning methods having the smallest generalisation error. A learner with low error over the training set and high generalisation error is often said to be *overfitted*, with overfitted models tending to be highly complex. Overfitting occurs whenever there is a large number of plausible hypotheses and it is possible for learners to find meaningless relationships in the data. The term *overfitted* has a similar meaning in statistics referring to models that are highly parameterised

in relation to the number of data observations available. Controlling overfitting is paramount in successful data analysis, with the accuracy versus readability trade-off being a key issue. Several authors have related this notion to Occam's razor[1]:

> *The most likely hypothesis is the simplest one that is consistent with all observations.* [37]

1.2.1 Data Mining Tools

In many data analysis and modelling pursuits, the focus is moving towards simple and interpretable models such as rules, trees and graphs. Primary tasks of data mining are considered to be classification, regression, clustering, dependency modelling and change and deviation detection. Data mining methodology can be grouped under the following headings:

- decision tree induction (tree based regression eg. *c*4.5 [33], CART [8])
- rule induction
- association rules
- clustering analysis
- hot spot analysis (identifying subgroups of cases which show especially significant patterns.)

[1]Sometimes spelled "Ockham"

Decision trees, neural networks and similar methods seem to be favoured by software developers. Hybrid methods combine the two. CART, originally a decision tree system, now incorporates logistic regression. A neural network is essentially a non-linear mathematical model for a learning process, good at pattern recognition and solving problems where algorithms suffer from computational complexity. There is a close connection between neural networks and some statistical methods but neural networks are often criticised because of their "black box" method of operation. Neural networks have received significant attention from engineers, computer scientists, biologists and neuro-physicists. They require the same assumptions as standard statistical techniques, in particular assumptions placed on the error structure. Decision tree s and neural networks are most effective with large datasets. They seem to make the best trade-off between sampling variance and model bias. Larger datasets may be more susceptible to model bias. Neither of the latter tools are an everyday tool for statisticians yet.

1.2.2 Types of Learning Problems

Learning problems can be classified into many groups [37]:

- **supervised** learning - this requires prior identification of relevant model inputs and correct outputs. This type of learner cannot operate without a human trainer. e.g. neural networks, decision trees

- **unsupervised** learning - the learner is given no indication as to the correct value of the output. e.g. clustering

- **reinforcement** learning - the learning system receives occasional positive or negative rewards, rather than being told the correct action.
- **passive** learning - the learner watches the world go by and tries to learn the utility of being in various states.
- **active** learning - the learner must act on learned information and uses its problem generator to suggest explorations of the unknown environment at every instance.
- **incremental** learning - the old hypothesis is updated whenever a new example arrives.
- **batch** learning - the learner is provided with all examples at once and forms its hypothesis using all data.

The performance of a learner can be assessed by plotting a learning curve as follows [37]:

- collect a large set of data (examples)
- divide the data into a *training* set and a *test* set
- apply the learner to the *training* set to generate the hypothesis $h(\mathbf{x})$
- measure the error of $h(\mathbf{x})$ based on data in the *test* set

The above steps are repeated for randomly varying training sets and varying training set size. Average prediction accuracy can then be plotted as a function

of training set size. Another method of assessing a learner's performance is that of cross-validation. k-fold cross-validation is applied by splitting the training set into k smaller groups. One of these groups is set aside as the test set and the learner is trained on the remaining $k-1$ groups. Error on the test set is measured and the process is repeated, with each of the k subsets of the original data becoming the test set. Accuracy results are averaged for all k subsets to give an average error rate and a variation in error rate. This process is very similar to the jackknife method commonly used in statistics [14].

1.3 Classification - a Specific Learning Problem

Classification is an induction task relevant to a wide variety of domains. Several methods have been developed in both the statistical and machine learning communities to solve such a problem. The general classification problem is posed formally as follows:

- A learner is presented with a series of N labelled training examples

  $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_N, y_N)$

- y_i is the class label associated with the vector of input attributes $\mathbf{x}_i$

- $\mathbf{x}_i = \{x_{1,i}, x_{2,i}, \ldots, x_{L,i}\}$, where $x_{j,i}$ can assume real or discrete values

- $y_i = \{1, \ldots, k\}$ is a k-class discrete variable, which represents the response given input $\mathbf{x}_i$

In the learning theory literature, y_i is not restricted to take only discrete values (for example regression allows y_i to be a real number). However, this Thesis is restricted to discussion of the classification problem, whereby y_i can only assume discrete class values. For example, in an industrial setting, the $x_{j,i}$'s may refer to temperature, incoming speed, colour or material type and y_i may refer to the severity of a particular defect or a breakdown condition. In the medical field, the $x_{j,i}$ may refer to a patient's blood pressure, age, sex, blood chemistry and y_i to the risk level of say, heart disease or the diagnosis of one disease over another.

This Thesis is concerned with the classification problem, whereby a learner is presented with a training set comprising a series of N labelled training examples of the form $(x_1, y_1), \ldots, (x_N, y_N)$, with $y_i \in (1, \ldots, k)$. The learner's task is to use these training examples to produce an hypothesis, $h(\mathbf{x})$, which is an estimate of the unknown relationship $y = f(\mathbf{x})$. This 'hypothesis' then allows future prediction of y_i given new input values of $\mathbf{x}_i$.

The objective of such a learning system is to find a set of rules that covers the observations of one particular class without covering data belonging to another class. Geometrically, a learning system fits boundaries in space to distinguish between classes. The ability to cover all of one class whilst omitting another class rarely happens in practice due to presence of noise, missing values and measuring inappropriate input variables.

Examples of existing classification methods include Fisher's linear discriminant, clustering, decision trees ($c4.5$ [33], CART [8]), rule based systems, neural

networks and instance based learning. These methods are discussed in more detail in Section 1.4.

1.4 Classification Tools

Weiss and Kulikowski [46] give a good overview of the most popular classification techniques:

- Neural networks
- Decision rules and trees
- Linear discriminant
- Clustering

1.4.1 Neural Networks

Neural networks are a form of non linear mathematical model which have links to biological nervous systems. It is believed that neural networks loosely model the operation of the brain, with interest fuelled by mankind's interest in modelling the functioning of the human brain. Even the naming of neural network components match nomenclature for sections of the brain. Because of the biological similies, it is often thought that the most potential for neural network application lies in speech recognition or vision [32, 46].

The basic component of a neural network is a node. Essentially a neural network comprises a series of interconnected nodes, where each node outputs a non linear function of the input. Because of the network's connectedness, inputs may feed

from a previous node, or straight from the input data source. In its entirety a neural network is highly complex set of interdependencies. Output from one node is input into another node so that information is disseminated through further nodes. Networks can be made as complex as befits the problem at hand. Outputs from one network may be fed as inputs into a new network resulting in highly complex and non linear structure.

Neural networks are very useful tools for complex functions with continuous outputs and many noisy inputs. However, the computational speed and parallel ability of neural networks has sparked significant interest and investigations but interpretability is still an issue. Networks can be highly accurate but the underlying learned concepts are not always evident to the analyst.

1.4.2 Decision Trees and Rules

A common problem in many model building techniques is that of model interpretability. In order to predict unseen cases, a human user often desires the ability to track the classification process. Mathematically inferenced models require the simplification of models which is a daunting prospect for the non-inclined. This problem led researchers to develop new methods capable of representing solutions in a manner that is easily understood and more aligned with human reasoning [46]. One representation that is easily understood by humans is that of logical association rules such as :

if $X_1 > 10$ *and* $X_2 =$ *False then Y=Class 1.*

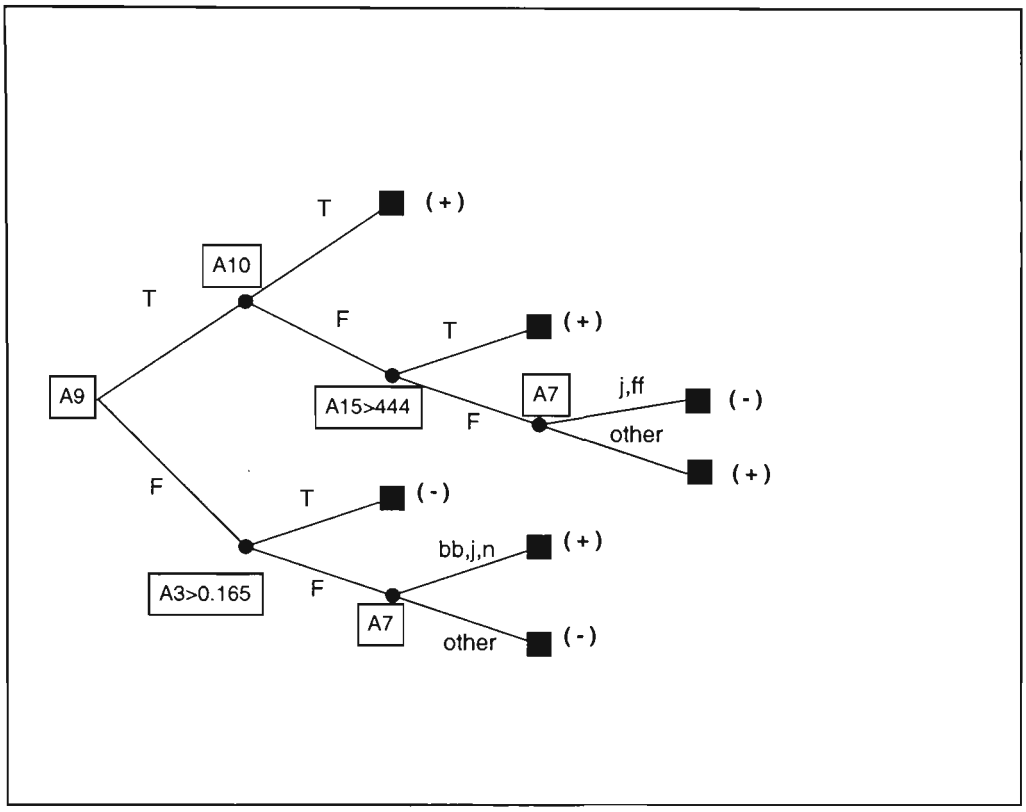

Figure 1.1: Example of a binary decision tree.

Decision trees and rules are built upon a series of such propositions and are evaluated to associate input data to a single class. Rules that are presented in a logical form are reassuring to humans.

Yet does this increase in interpretability lead to a decline in accuracy? Empirical studies to date have shown that techniques in logical rule representation are comparable in accuracy performance to other existing methods [32].

Decision trees consist of nodes and branches: an example of a simple *binary* decision tree is given in Figure 1.1. Each node represents a single decision step whose result decides along which branch to next proceed. Terminal nodes or leaves are responsible for the final allocation of class. Rule induction is used to build the

optimal tree with several algorithms currently existing to perform this task. Popular decision tree algorithms that currently exist are CART [8], *c*4.5 [33] and CHAID. Rule based systems include Induct, Ripple Down Rules and *c*4.5's rule formation capacity.

Tree based algorithms often differ in splitting criterion, with popular splitting criterion including information gain, Gini criterion and residual least squares. The amount of pruning applied to tree growth is another powerful option in decision tree algorithms. Forward pruning involves building a decision tree one split at a time whereas backward pruning is based on building a tree that is too large and pruned up the branches. The latter is the basis of pruning for the *c*4.5 and CART algorithms.

Chapter 2

Recent Advances in Classification

2.1 Ensembles of Classifiers

The area of ensemble classification is a very active area of research in the machine learning community. Recall from Section 1.2.2, a learning algorithm is presented with training examples of the form $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_N, y_N)$, and outputs an hypothesis $h(\mathbf{x})$ which is an estimate of the unknown relationship $y = f(\mathbf{x})$. Ensemble classifiers comprise of several $h(\mathbf{x})$'s whose individual predictions are combined in some fashion to produce a single classifier. Methods of combination vary, but weighted or unweighted voting currently appears to be the most popular.

Research into these methods is currently very active in the machine learning and statistical communities. Several authors have demonstrated improved generalisation error by employing ensemble classifiers to a variety of classification problems [4, 12, 34, 41, 42]. These empirical studies have demonstrated that the final accuracy of a combined classifier is often better than the accuracy of individual classifiers making up the ensemble. Several authors have attempted

to address the reasons for improved classifier accuracy when ensembles of classifiers are grown. Ensembles have been shown to lead to improved performance on unstable learning methods. i.e. methods where small changes in the input data lead to large changes in the learned classifier. Methods such as decision trees and neural networks are considered unstable. An ensemble can be more accurate than individual classifiers only if there is a conflict in classification between the individual classifiers [12, 26]. Essentially, this equates to errors in the $h(\mathbf{x})'s$ being uncorrelated, which can be explained intuitively as follows.

Assume we have hypotheses $h_1(\mathbf{x}), \ldots, h_M(\mathbf{x})$, all of which output the same prediction for y_i. If the prediction for y_i is incorrect, no amount of voting or reweighting will produce the correct classification. However, if the class predictions for the $h_i(\mathbf{x})$'s differ, appropriate combination of these predictions should lead to a more accurate classifier overall. Dietterich [12] simulates error rates for L hypotheses, all of which have error rates, $\epsilon < 1/2$. If errors are independent, the probability that the majority vote is incorrect is equal to the area under a binomial distribution where more than $L/2$ hypotheses are wrong. This is true since

$$Pr(h(\mathbf{x}_i) \neq y_i) \quad = \quad p$$

The probability that more than $\frac{L}{2}$ hypothesis will be incorrect is given by:

$$\sum_{i=[\frac{L}{2}+1]}^{N} p^i (1-p)^{N-i}$$

which is the area under a binomial distribution.

Dietterich [12] simulated this result for 21 hypotheses, each with an error rate of 0.3. The probability that more than 11 hypotheses would be simultaneously wrong is equal to 0.026, which is clearly much less than the individual error rate of 0.3. However, if the individual error rate exceeds 0.5, voting can increase the combined classification error. Vital to successful voting methods is ensuring individual classifiers have error rates less than 50%, with minimal correlation between error rates.

Ensemble generating methods can be classified into the four areas listed in Section 2.2. Several methods also exist for combining ensembles of classifiers once the individual classifiers have been built.

2.2 Methods for Generating Ensembles

Dietterich [12] groups existing ensemble growing methods into 4 distinct groups:

1. Subsampling training examples (or reweighting training examples)

2. Manipulating Input Features

3. Manipulating Output Targets

4. Injecting Randomness

2.2.1 Subsampling Training Examples

A series of classifiers can be built by training a learning algorithm on different subsamples of the original training set. The notion of subsampling to improve estimation has been applied extensively in the statistics community. Quenoille [14] first introduced the jackknife method as a means of eliminating bias in statistical parameter estimation. Tukey [44] suggested that the technique may be used to estimate variances and since then the jackknife has been applied and researched broadly [14]. The jackknife procedure applies a leave one out methodology of subsampling and estimation, similar to cross-validation.

The bootstrap is another non-parametric method for estimating the variance and bias of a population parameter. Bootstrapping was first introduced by Efron in 1979 [15] and has since proved a popular tool within the statistics community. Bootstrapping is a resampling procedure which is based on observations being selected from a parent sample in order to form several subsamples. A bootstrap replicate of the original training set of N elements is taken by selecting a random sample of size N with replacement from the original training set (i.e. each element has probability $1/N$ of selection). On average, 63.2% of the original training set is included in the bootstrap replicate, with some original training examples appearing more than once. (The figure 63.2% is asymptotic in N since $lim_{N\to\infty}(1-\frac{1}{N})^N = e^{-1}$)

Ensemble generation methods based on the bootstrap and a weighted form of the bootstrap have been developed by both computer scientists and statisticians.

These methods are described in more detail in Section 2.3 and form the backbone of the analysis in this Thesis.

2.2.2 Manipulating Input Features

This method of ensemble generation changes the form of input data available to the learning algorithm. Sammut et. al. applied this method to the problem of learning to fly a Cessna on a flight simulator [39]. The aim of the study was to learn the correct mapping from a series of input parameters to 90,000 examples, each with 20 state variables. The decision tree algorithm $c4.5$ [33] was used and fed back into the control system, resulting in the program learning to fly. Similar statistical approaches include principal component analysis and factor analysis for dimensionality reduction. Manipulating the input features works well when some of the input features are redundant.

2.2.3 Manipulating Output Features

Many methods (for example, neural networks) are well suited for the problem of learning binary tasks. However, in many applications, we are faced with a multi-class target variable. The method of *Error Correcting Output Coding (ECOC)* accommodates for this by formulating a series of new binary classification problems, which may then be combined to predict the value of a multi-class variable. Dietterich and Bakhiri [11] first introduced the method of *ECOC*, with its aim being the reformulation of a single k-class problem into k 2-class problems. This technique is particularly effective as k, the number of classes increases.

2.2.4 Injecting Randomness

Studies have been carried out on neural networks by injecting randomness into the initial weights. The same examples are used for modelling but they have varying initial weights. Decision trees such as *c*4.5 can also accommodate the injection of randomness. *c*4.5 makes split decisions based on an information gain criterion to rank possible contenders for a branch split. Dietterich [13] et. al. devised a version of *c*4.5 which chooses randomly with equal probability among the top 20 best tests. Results of their empirical study show that injecting randomness performs more favourably than bagging and *c*4.5 applied with no randomisation. Ali and Pazzani [1] published work based on the FOIL learning system which showed error reduction can take place through learning multiple descriptions. The authors ranked all candidate conditions that scored within the top 80% of information gain and applied a random choice algorithm using these weights. Work has also been done on randomisation in the neural network field. Much of this work is closely related to the statistical technique of Markov Chain Monte Carlo simulation.

2.3 Ensembles Built via Resampling

Resampling techniques have proven to be successful in statistical estimation problems when high variance was encountered in the parameters to be estimated. Similarly, [12] indicates that ensembles grown from repeatedly applying a learning algorithm over different subsets of the training data, improve generalisation error

for unstable learning algorithms (i.e. algorithms with high variance whose output classifications vary significantly with small perturbations in the training data). Decision trees, neural networks and rule learning methods are all considered to be unstable, whereas linear regression, nearest neighbour and linear threshold techniques are usually stable [5, 12].

The first subsampling method for ensemble building was introduced by Breiman in 1996 [3]. *Bagging* or *b*ootstrap *agg*regating sees the learning algorithm applied on several training sets which are bootstrap replicates of the original training set. Bagging applied to **CART**[1] showed significant decreases in test set error. Bagging is a variance reduction technique and since trees are high in variance, bagging produces good results. Brieman also noted that the stable methods of nearest neighbour and linear discriminants did not yield significant improvement when resampling procedures were applied.

The term *arcing* (*a*daptive *r*esampling and *c*ombining) was coined by Breiman in 1996 and refers to the procedure of reweighting the data in the training set, building a new classifier based on the reweighted data, and combining the resulting classifiers [4].

In 1996, Freund and Schapire [20] introduced an extremely successful *arcing* procedure, *AdaBoost*. The *AdaBoost* algorithm is based on repeated resampling from a probability distribution D_m, where D_m is adjusted multiplicatively at each iteration so that observations which are predicted incorrectly in the previous iteration

[1] **CART** : **C**lassification and **R**egression **T**rees [8]

are given more weight in the next round. The system 'boosts' a weak learner and forces it to form predictors from 'harder' sections of the space. ('harder' refers to those sections of the space which are difficult to predict). Boosting is presented in more detail in Section 2.3.1.

Brieman [5] claims that the main effect of *bagging* and *arcing* is to reduce variance, where reduction comes from the adaptive resampling and not the specific form of the *arcing* algorithm. Brieman also comments that it is not clear what the optimal arcing algorithm will look like and defines another arcing algorithm *arc-x4* [5] as follows:

- let $m(n)$ be the number of misclassifications of the nth case by $h_1, h_2, \ldots, h_m$
- $w_i^{t+1} = \frac{1+m(n)^4}{\sum(1+m(n)^4)}$ (w_i^{t+1} is the $t+1$ step weighting for observation i)
- after m iterations, the $h_1, h_2, \ldots, h_m$ are combined by unweighted voting

This algorithm has similar empirical advantages of reducing training error and was developed ad-hoc to prove that the concept of adaptive resampling is what really reduces error. Breiman [5] claims that arcing is more successful than bagging in variance reduction. It is widely recognised that

> additional research aimed at understanding the workings of this class of algorithms will have a high pay-off [5].

2.3.1 Boosting

Introduced by Freund and Schapire [20] in 1995, boosting is recognised as being one of the most significant recent advances in classification. The algorithm

'boosts' a weak learner to yield a combined classifier with improved generalisation error. Since its introduction, boosting has been the subject of many theoretical and empirical studies [20, 34, 41, 42]. An advantage of this method is that it does not require any background knowledge about the performance of the underlying weak learning algorithm.

Boosting works on the premise of maintaining a distribution, D_m, of weights for each observation. At the m-th iteration, a sample is taken from the original training set according to D_m. Weights are adjusted multiplicatively at each iteration according to whether a particular training example was classified correctly by the most recent classifier. Those examples which were classified incorrectly have their weight increased so that the total weight on incorrect observations is $\frac{1}{2}$. Hence the learning algorithm will be given more opportunity to explore areas of the training set which are more difficult to classify.

Hypotheses generated from these difficult parts of the space make fewer mistakes on these sections and can play an important role in prediction when all hypotheses are combined via weighted voting. The weighted error rate of the classifier at each iteration is calculated, stored and used in the final voting weight when individual classifiers are combined to form the ensemble. Accuracy of the final hypothesis depends on the accuracy of *all* the hypotheses returned at each iteration and exploits hypotheses that predict well in more difficult parts of the

instance space.

The *AdaBoost* algorithm is first presented for the two class problem where $y_i \in \{0, 1\}$ and is extended to include the multi-class problem where $y_i \in \{1, 2, \ldots, k\}$ (**AdaBoost.M1**). A further enhancement involves using pseudo-loss to adjust weights as opposed to weighted error at each iteration. The **AdaBoost.M1** algorithm is outlined in Table 2.1. At each iteration $(1, \ldots, M)$, a sample of N observations is taken from the initial training set according to a distribution D_m. Initially $D_1(i) = \frac{1}{N}\ \forall i$. **WeakLearn** is called on the sample resulting from D_m to produce a *weak* hypothesis, h_m. The weighted error from h_m is calculated as $\epsilon_m = Pr_{i \sim D_m}[h_m(\mathbf{x}_i) \neq y_i)]$, and h_m should have low classification error on the observations with high weights according to D_m . Based on the value of ϵ_m, D_m is updated multiplicatively in such a way that weight is increased on examples which were classified incorrectly. The resulting weights reflect the relative importance of each example for the next round. Classifiers are then combined by weighted voting involving the ϵ_m's.

Table 2.1: AdaBoost:M1

AdaBoost:M1

Input: N training instances $\mathbf{x}_i$ with labels y_i. Maximum trials, M. Base learner, M.

Initialization: All training instances begin with weight $w_i^0 = 1/N$.

Repeat for M trials:

- Induce classifier, $h_m(\mathbf{x})$, using weighted training data and H.
- ϵ_m = weighted error for $h_m(\mathbf{x})$ on the training data. If $\epsilon_m > 1/2$, discard h_m and stop boosting. (If $\epsilon_m = 0$, then h_m gets infinite weight.)
- Classifier weight, $\beta_m = log\frac{1-\epsilon_m}{\epsilon_m}$
- Re-weight training instances:
  if $h_m(x_i) \neq y_i$ then, $w_i^{m+1} = w_i^m/(2\epsilon_m)$
  else, $w_i^{m+1} = w_i^m/2(1-\epsilon_m)$

Unseen instances are classified by voting the ensemble of classifiers h_m with weights β_m.

Note the weight update rule is multiplicative since $w_i^{m+1} = w_i^m \beta_m^{\|1-h_m(\mathbf{x}_i)\neq y_i\|}$ For the 2-class problem, Freund and Schapire [20] prove upper bounds on the error of the final hypothesis with respect to the training set. It can be shown that this error is bounded by $exp(-2\sum_{m=1}^{M}\gamma_m^2)$, where $\epsilon_m = 1/2 - \gamma_m$ is the error of the m-th hypothesis. These bounds have been shown to be overly generous by several authors, and empirical studies have demonstrated that boosting achieves *far better* accuracy than the derived theoretical bounds. It remains a research challenge to develop tighter bounds on this error. Freund and Schapire explain

> γ_m measures the accuracy of the m-th hypothesis relative to random guessing (which has expected error 1/2).

Hence, if a weak learner can consistently produce hypotheses with error better than 1/2, then the error of the final hypothesis drops exponentially fast. They prove upper and lower bounds on the $\sum w_i^{m+1}$ and then imply lower bounds on the loss. In this case, the loss function is a form of the indicator function. i.e. $l_i^m = 1 - \|h_m(\mathbf{x}_i) - y_i\|$ and is smallest when h_m produces an incorrect classification. Therefore, the exponent of β_m in the weight update rule is decreased when a correct classification is made. Decreasing β_m also increases $\log(1/\beta_m)$ which is allocated to h_m in the final classification decision. Therefore more accurate hypotheses result in larger changes in D_m and have more influence on the outcome of the final hypothesis.

Recall that final error is defined as $\epsilon = Pr_{i\sim D}[h_m(\mathbf{x}_i) \neq y_i]$. Freund and Schapire [40] prove that the error is bounded above by $\epsilon \leq 2^M \prod_{m=1}^{M} \sqrt{\epsilon_m(1-\epsilon_m)}$. This

implies that the final error depends on errors of all input hypotheses [20]. An important theoretical property to be noted is that if the error of individual classifiers is less than $\frac{1}{2}$, the final error drops exponentially fast to zero. For binary classification this implies that individual classifiers need only be slightly better than random guessing in order for boosting to yield improvements. Now while the training error drops exponentially fast, this says nothing about generalisation or test set error.

The resampling techniques described above fall into the category of *'perturbing and combining'*, whereby multiple versions of the classifier are generated by perturbing the training set and then combining these classifiers to yield a final predictor.

Authors such as [24, 40] have presented the **AdaBoost.M1** algorithm in terms of $y_i \in \{-1, 1\}$ with varying update rules. The proof below demonstrates the consistency of both notations.

Redefine Freund and Schapire's *AdaBoost* in terms of $y_i \in \{-1, 1\}$

$$\begin{aligned} \epsilon_m &= \sum_{i=1}^{N} p_i^m \| h_m(\mathbf{x}_i) - y_i \| \\ &= \sum_{i=1}^{N} p_i^m 1_{(h_m(\mathbf{x}_i) \neq y_i)} \\ &= E_w[1_{(h_m(\mathbf{x}_i) \neq y_i)}] \\ &= E_{D_m}[1_{(h_m(\mathbf{x}_i) \neq y_i)}] \end{aligned}$$

ϵ_m is the same as e as defined in Friedman et. al. [22].

Let $\beta_m = (\frac{\epsilon_m}{1-\epsilon_m})$

Now, in Freund/Schapire [20]

$$\begin{aligned} p_i^{m+1} &= p_i^m \beta_m^{1-\|h_m(\mathbf{x}_i)-y_i\|} \\ &= p_i^m \left(\frac{\epsilon_m}{1-\epsilon_m}\right)^{1-\|h_m(\mathbf{x}_i)-y_i\|} \\ &= p_i^m \exp\left(\log\left(\frac{\epsilon_m}{1-\epsilon_m}\right)^{1-\|h_m(\mathbf{x}_i)-y_i\|}\right) \\ &= p_i^m \exp\left(1-|h_m(\mathbf{x}_i)-y_i|\right)\log\left(\frac{\epsilon_m}{1-\epsilon_m}\right) \end{aligned}$$

Now,

$$\begin{aligned} 1-\|h_m(\mathbf{x}_i)-y_i\| &= 1 \quad if \mathrm{h}_m = y \\ &= 0 \quad if \mathrm{h}_m \neq y \end{aligned}$$

Therefore $1-\|h_m(\mathbf{x}_i)-y_i\| = 1_{(h_m(\mathbf{x}_i)=y_i)}$

$$p_i^{m+1} = p_i^m \exp[\log\left(\frac{\epsilon_m}{1-\epsilon_m}\right)1_{(h_m(\mathbf{x}_i)=y_i)}]$$

Vote with weight $\log(\frac{1}{\beta_m}) = \log(\frac{1-\epsilon_m}{\epsilon_m})$

So the weight is decreased if $h_m(\mathbf{x}_i) = y_i$. i.e. $\exp(\log(\frac{\epsilon_m}{1-\epsilon_m})) = (\frac{\epsilon_m}{1-\epsilon_m}) < 1$, and if $h_m(\mathbf{x}_i) \neq y_i$, then $w_i^{m+1} = w_i^m$.

Comparing this updated weight rule to that of Friedman et. al. [24], Freund and Schapire actively *decrease* weight on correct estimation. Friedman et. al. actively *increase* weight on incorrect estimation so that $w(y|x) \leftarrow w(y|x) \times \exp\left(\log(\frac{1-\epsilon_m}{\epsilon_m})1_{(y\neq\hat{f})}\right)$.

Now $\log(\frac{1-\epsilon_m}{\epsilon_m}) = \log(\frac{1}{\beta_m})$ and $H(x) = sign(\sum c_m f_m)$ where $c_m = \log(\frac{1}{\beta_m})$

Using the value for c_m derived in Friedman et. al.

$$\begin{aligned} sign(\sum c_m f_m) &= sign\left(\frac{1}{2}\sum \log(\frac{1-\epsilon_m}{\epsilon_m})f_m\right) \\ &= sign\left(\frac{1}{2}\sum \log(\frac{1}{\beta_m})f_m\right) \\ &= sign\left(\sum \log(\frac{1}{\beta_m})f_m\right) \end{aligned}$$

Hence the results derived from Freund and Schapire's *AdaBoost* [20] paper match Friedman et al. [24].

2.3.2 Results from Existing Empirical Studies on Ensembles Built via Resampling

Several empirical studies using a selection of representative datasets have been carried out on boosting and bagging. The common conclusion of studies is that a weak learner's performance can be considerably improved by forming ensembles via resampling. Based on the results published to date, boosting appears to have an advantage over bagging. Quinlan [34] conducted a study on the performance of boosting and bagging when applied to the decision tree learner, $c4.5$. The study presented results of applying boosting and bagging with $c4.5$ as the underlying learner to datasets from the UCI[2] [2] repository for $M = 10$ rounds of boosting. Quinlan remarked that in most cases, the additional computational complexity required to form ensembles could be justified in light of the marked improvement in classification accuracy. Overall, boosting appears to be more

[2]URL = http://www.ics.uci.edu/~mlearn/MLRepository.html

effective than bagging in error reduction when applied to *c*4.5. It was also noted that the variance of the bagged ensembles was lower than boosting.

Because boosting's performance is not consistently better, an open question still exists. Quinlan suggests that weights may be adjusted separately within each class without adjusting overall class weights and also comments on whether the weight distribution might have something to do with why boosting may sometimes fail. Freund and Schapire [20] discuss results from experiments with a boosting algorithm. Their study used $M = 100$ as opposed to Quinlan's $M = 10$ and may explain the slight discrepancies in their results. Non-randomness was removed by Quinlan by retaining weights as opposed to resampling performed by Freund and Schapire. This may also explain some of the varying results. Schapire also looked at combining *ECOC* [11] with boosting to yield significant improvements [40].

Freund and Schapire designed *AdaBoost* to drive training error rapidly to zero. But how does it perform on test set error? Breiman [5] tried running *bagging* and *AdaBoost*, exiting when the training error became zero and keeping track of both the number of iterations to exit and the resulting test set error. Breiman concluded that *AdaBoost* reaches zero training error very quickly but the accompanying test set error is higher than that of bagging which takes longer to reach zero training set error. To produce optimum reductions in test set error, *AdaBoost* must be run far past the point of zero training set error. Brieman concludes by commenting that

> Arcing algorithms have a rich probabilistic structure and it is a challenging problem to connect this structure with their error reduction properties. [5]

Breiman indicates that steady state probability structure is important based on the outcome of his experiments but how this is so is not clear. What is known is that arcing derives most of its power from the use of adaptive resampling to reduce variance, as demonstrated by the ad-hoc $arc - x4$. In other words, it is not so much the exact form of the arcing algorithm but rather the concept of adaptive resampling which does the job by focussing on those cases that are hard to classify. It is not understood why *AdaBoost* degenerates below the performance of bagging, particularly on smaller datasets. A possible explanation may be the presence of outliers - an outlier will be persistently misclassified so its probability of being resampled increases to the point where it is sampled several times in the perturbed training set and distorts the classifier.

Most recent empirical studies boost by reweighting where D_m can be supplied directly to the learner. Boosting by resampling involves using D_m to actively take a subsample of the original training set and this new subset is presented to the learner. Resampling adds further randomness to the process and results in a slight decrease in performance.

Freund and Schapire [20] show that boosting can improve the performance of a *weak* learner when the performance of the weak learner is only slightly better than random guessing. Boosting was also compared to bagging in order to discover

the effect of the underlying sampling distribution . The following weak learners were used :

1. Non-randomness is removed by retaining weights as opposed to resampling.

2. A rule learner which searches for simple prediction rules based tests on a single attribute

3. An algorithm that searches for a single good decision rule, testing on conjunction of attribute tests similar to RIPPER [9] and IREP [?]

4. The $c4.5$ algorithm [33]

Datasets used were a representative set of 27 from the UCI [2] Machine Learning Repository. It was concluded that boosting consistently performs better than bagging when weak learners produce simple classifiers 1 and 2. With $c4.5$ and nearest neighbour methods, boosting still performs better but the results are not as convincing. Freund and Schapire [20] also introduce a new boosting algorithm **AdaBoost.M2** whereby a set of *plausible* classifications are output as opposed to a single y_i. Freund and Schapire explain that they experiment with this new algorithm to test if boosting using pseudo-loss as opposed to error is worthwhile. Their results indicate that pseudo-loss boosting outperforms standard error boosting as applied in **AdaBoost.M1**. Non-binary problems perform better if pseudo-loss is used. Friedman [23] has recently introduced a gradient boosting algorithm which is demonstrated to perform comparably to existing

boosting algorithms.

2.4 Why Does Boosting Work?

The area of ensemble classifiers and the reasons for their improved generalisation performance remains a fertile ground for further research, with significant gains still possible if such reasons are addressed. In theory, as the combined classifier complexity increases, the gap between training and test set error should increase. However, this is not reflected in empirical studies. There is strong empirical support for the view that overfitting is less of a problem (or perhaps a different problem) when boosting and/or arcing methods are used to improve a learner. Some authors have addressed this issue via bias and variance decompositions, which have been applied in an attempt to understand the stability of a learner [3, 21]. Although Breiman's and Friedman's definitions differ mathematically, their underlying concepts of bias and variance are the same.

Bias refers to the repeated error of the learning algorithm, which is the error that would be observed when an infinite number of independently trained classifiers were trained. The variance term refers to the wavering that can occur when a single classifier is trained. The idea of combining is to average over many classifiers to reduce the variance term and hence the resulting expected error. Friedman [21] gives the following explanation:

> The variance reflects the sensitivity of the function estimate to the

> training sample. Less sensitivity means that the estimate will be more stable against changes (sampling variation) in the data and thus be less variable under repeated sampling. ... The bias reflects sensitivity to the target function $f(\mathbf{x})$. It represents how closely on average the estimate is able to approximate the target.

Boosting is capable of bias and variance reduction and thus differs fundamentally from bagging. Schapire et. al. [41] claim that a weighted algorithm performs better than weighted resampling at each iteration as it removes the randomisation.

According to Breiman's [5] definition of bias, trees have high variance and low bias, whereas linear regression has low variance and high bias. Authors have developed bias-variance decompositions for classification problems which may address the problem of overfitting as statistics deals with overfitting via analysis of bias and variance. Breiman [5] claims that the main effect of *bagging* and *arcing* is to reduce variance, where reduction comes from the adaptive resampling and not the specific form of the *arcing* algorithm.

Further ideas as to why boosting works include a lack of overlap of datasets at each iteration through resampling or reweighting, resulting in uncorrelated errors. Another theory posted as to the success of ensemble classifiers is the non-overlap of errors. i.e. observations which are classified successfully by one hypothesis, are classified incorrectly by others and vice versa.

In theory, as the combined classifier complexity increases, the gap between training and test set error should increase but this is not seen in empirical studies. Empirical studies such as [5, 34] and [41] prove that large variance of the base classifier is *not* a requirement for boosting to be effective. In some cases, boosting increases the variance while reducing overall generalisation error. The idea of combining classifiers is to average over many classifiers to reduce the variance term and hence the resulting expected error. Studies have demonstrated that large variance of the base classifier is not a requirement for boosting to be effective [3, 34, 41]. In some cases, boosting increases the variance while reducing the overall generalisation error.

It is conjectured in existing studies that boosting helps algorithms with the following properties:

1. Observed examples have varying degrees of *hardness*.

2. The learning algorithm must be sensitive to changes in training examples so significantly different classifiers can be built.

Freund and Schapire [20] indicate that the greatest improvement in performance on a complex algorithm will be obtained when large amounts of data are available. Boosting is similar to bagging [4] in that it is best when underlying classifiers are unstable. Boosting changes the distribution over training examples more dramatically than bagging. Two reasons are conjectured as to why boosting may work [20]:

1. It generates hypotheses whose error on the training set is small by combining many hypotheses whose individual error may be large.

2. Variance reduction occurs by taking weighted majority of hypotheses, random variation is reduced.

Boosting, however, reduces bias more than bagging [5, 29].

If it is possible to build ensembles that improve classification accuracy, why is it not possible to build a single classifier in the first step which outperforms all other individual classifiers? Dietterich [12] answers this question by explaining that machine learning algorithms search a space of possible hypotheses for the most accurate hypothesis $h(\mathbf{x})$ which best fits the given training data, not necessarily the test data. Dietterich also makes the following observations:

- If the hypothesis space is large, we need a good sized training set in order to find a good approximation to $y = f(\mathbf{x})$. In a 2-class problem we need $O(\log |h|)$ (where h is a measure of the size of the hypothesis space) examples to select a unique classifier from the space. Therefore the given training data may not provide sufficient information for choosing a single best hypothesis. All hypotheses may have similar accuracy with respect to the training data as the training space is not representative of the entire environment.

- A learning algorithm may not be capable of solving difficult search problems. For example, some tree search problems are NP-Hard and have been implemented using a greedy search heuristic.

- The hypothesis space may not contain the true value of $f(\mathbf{x})$ but may only include approximations to f. By weighting different versions of approximations to f, we can achieve an estimate of f outside the available space.

Ensembles overcome representational inadequacies in the hypothesis space. The following comments and conclusions on boosting have been made to date in the literature:

- More recently, Breiman's position is *equalising*. Equalising refers to the tendency for the proportion of misclassifications for each observation to become more uniform as the number of boosting trials increases. [7]

- Successful ensemble classification is due to the non-overlap of errors [12] i.e. observations which are classified correctly by one hypothesis are classified incorrectly by others and vice versa. This notion is similar to that of non-overlapping subsamples and negatively correlated errors.

- If the final hypothesis is given by $H(x) = sign(\sum \alpha_m h_m \mathbf{x})$ where α_m is chosen to minimise a normalisation factor, then with respect to distribution D_{m+1}, $h_m(\mathbf{x_i})$ is exactly non-correlated with y_i. [20, 42]

- If the final voted classifier is denoted as $F(x)$, then the following corollary holds: "At the optimal $F(x)$ the weighted conditional mean of y is zero." ie $E_w(y|x) = 0$, with y being the response or class variable. [20, 24] This is result is similar to that described in the item above.

- The original intuitive idea behind *AdaBoost* is to concentrate weight on instances most likely to be misclassified. Breiman [5] developed *arc-x4* which aggressively concentrates weight on hard-to-classify examples and yielded comparable generalisation error to *AdaBoost.*

- Margin and edge analysis are recent explanations. More detail on these measures is provided in later sections of this Thesis (Sections 2.5 and 2.5.1).

- Schapire et. al. [41] proved results involving two terms for the upper bound of generalisation error: one term a function of the margin and the other being a complexity term. The complexity term is very loose for rich function classes such as decision trees.

- Breiman's recent study [7] introduces a new algorithm which whilst successful in minimising a criteria other than misclassification at iteration j, results in a concentration on many examples that would have not been considered difficult to classify. This new algorithm still gives lower generalisation error and this evidence seems to refute the notion of generalisation error improving by concentrating on difficult to classify examples. However, I am grateful to the examiner for pointing out a counterexample. This example follows:

  Let Algorithm A find a large margin combination from some small subset of a class F and Algorithm B find a large margin combination from all of F. Schapire et. al's bound still applies since both algorithms use class F.

If, for instance Algorithm B finds a better margin distribution but results in more complex functions in F then it would not be surprising to find higher error for Algorithm B's combination. This example does not contradict the bounds of Schapire et. al [41], nor their qualitative conclusions.

2.5 Edge and Margin Analysis

A repeatable phenomenon of boosting experiments is a non-increasing of the generalisation error, even as boosted ensembles become very large and perhaps overfitted. Recent explanations as to the success of boosting algorithms have their foundations in margin and edge analysis. These two measures are defined for the i-th training observation at trial m under voting vector $\mathbf{c} = (c_1 \dots c_m)$ as follows:

- $edge_i(m, \mathbf{c})$ = total weight assigned to all incorrect classes. The edge is defined formally in [3]. Assume the ensemble comprises a combination of base learners, each of which produce $h_m(\mathbf{x})$ at the m-th iteration. From $h_m(\mathbf{x})$, an error indicator function $I_m(\mathbf{x}_i) = I(h_m(\mathbf{x}_i) \neq y_i)$ may be determined. Let c_m represent the unnormalised vote for the m-th hypothesis. Then, for the i-th observation

$$edge_i(m, \mathbf{c}) = \frac{\sum_{j=1}^{m} c_j I_j(i)}{\sum_{j=1}^{m} c_j} \tag{2.5.1}$$

- $margin_i(m, \mathbf{c})$ = total weight assigned to the correct class minus the maximal weight assigned to any incorrect class.

For the 2-class case $margin_i(m, \mathbf{c}) = 1 - 2\ edge_i(m, \mathbf{c})$ and in general $margin_i(m, \mathbf{c}) \geq 1 - 2\ edge_i(m, \mathbf{c})$ [3]. For each observation, the edge is essentially the proportion of votes assigned to incorrect classes.

Whilst more difficult to compute, the value of the margin is relatively simple to interpret. Margin values will always fall in the range $[-1, 1]$, with high margins indicating confidence of correct classification. An example is classified incorrectly if it has a negative margin. The edge on the other hand cannot be used as an indicator variable for correct classification (except in the 2-class case). Whilst the margin is a useful measure due to its interpretability, mathematically it is perhaps not as robust and tractable as the edge. These two diagnostics have been the subject of several recent empirical and theoretical studies [6, 7, 41].

Schapire et. al. [41] claim that

> Boosting is particularly good at finding classifiers with large margins in that it concentrates on those examples whose margins are small (or negative) and forces the base learning algorithm to generate good classification for those examples.

i.e. boosting results in a higher 'distribution' of margins with boosting giving an overall higher margin distribution than bagging.

Breiman's position is made clear in [6], where he claims that the margin distribution does not determine generalisation error. This is demonstrated empirically

by introducing a new algorithm, *arc-gv*, which actively minimises the value of $\max[edge_i(m, \mathbf{c})]$. This is equivalent to tracking the proportion of training examples with margin less than some threshold, which is the explanation for boosting's success given in the Schapire et.al. paper [41]. Empirical results show that *arc-gv* results in lower edge values and hence higher margin values than *AdaBoost*. However, this is not reflected in a lower generalisation error. Brieman concludes that the current margin explanation is incomplete and the "*explanation must run deeper*". [6]

In their paper, Schapire et. al [41] derive bounds for the generalisation error based on VC-dimension and the margin distribution on the training set. These bounds are loose, and whilst correct in direction, do not give a true indication of the expected magnitude of generalisation error. Bounds show for a fixed VC-dimension, the larger the margin, the smaller the generalisation error.

To check this empirically, Breiman performed some empirical studies on *AdaBoost* versus *arc-gv* with VC dimension fixed. Recall *arc-gv* actively reduces $top_m(\mathbf{C})$ to its minimum value, equivalent to maximising the margins. Whilst *AdaBoost* does not actively pursue this strategy, Schapire et. al. claim that its power lies in its ability to increase margins. We would therefore expect *arc-gv* to outperform *AdaBoost* in terms of generalisation error as it is more aggressive in its margin maximisation. Empirically, this is not the case : test set errors are close, with *AdaBoost* generally performing better, but the values of $top_m(\mathbf{C})$ are significantly smaller for *arc-gv*. These results are almost the reverse of what would be expected

theoretically if margins are the key for improved generalization error. Examples show that vastly different margin distributions with fixed VC-dimension have little effect on generalisation error. In some cases the opposite margin effect was seen. Results so far provide grounds for further investigation leading to work in this Thesis grounded in analysis of different aspects of the edge and margin distributions. Breiman [6] derives different bounds based on VC-dimension and a measure, $top_m(\mathbf{C})$. When all training examples are given zero margin cost, Brieman's bounds are approximately the square root of the Schapire et. al. bounds and imply a similar trend - i.e. with increasing margins, lower generalisation error results. Although Breiman's bounds do not always give non-trivial results, they result in similar implications. If the Schapire et. al. explanation is correct, the VC based bounds are giving the correct qualitative picture.

In recent work, Breiman [7] generalises ensemble methods to *Random Forests* whereby an ensemble is considered to be a "forest" of trees. Each tree is built on a sample drawn using random vectors with the same distribution for each tree. New measures of *strength* and *correlation* as average margin and correlation between ensemble errors respectively. Both of these measures are found to be factors in determining the error rate of the entire forest. Breiman also tests random splits for node determination. (refer also to Section 2.2.4 for a summary of other methods which rely on injecting randomness).

To improve accuracy, Breiman conjectures that correlation needs to be minimised while maintaining strength. Randomness is injected via sampling and the goal is

to optimise strength via random sampling. Again, this is a similar conclusion to that presented by Schapire et. al. [41]. Breiman also proves upper bounds on the generalisation error with correlation and strength terms. Although loose, the bounds lie in the same direction as other bounds developed in the literature - in particular the VC-type bounds. The similarity lies in the two key constituents in the bound being strength and correlation expressed in terms of margin functions. Sections 3.1 and 3.3 in this Thesis examine the variance of the edge values versus the number of boosting trials performed. Methodology for this study is discussed in detail in Section 3.1 of the next Chapter.

Shown below is the proof for the binary classification problem that:

$$E[edge_i(m, \mathbf{c})] = 1 - \frac{1}{2}E[margin_i(m, \mathbf{c})] \tag{2.5.2}$$

And

$$Var[edge_i(m, \mathbf{c})] = \frac{1}{4}Var[margin_i(m, \mathbf{c})] \tag{2.5.3}$$

Therefore any overall trends and results presented in this Thesis pertaining to edge variance will hold true for margin variance and mean edge will be inverse to mean margin. Recall for the 2-class case $margin_i(m, \mathbf{c}) = 1 - 2edge_i(m, \mathbf{c})$. Hence:

$$\begin{aligned} E[margin_i(m, \mathbf{c})] &= E[1 - 2edge_i(m, \mathbf{c})] \\ &= 1 - 2E[edge_i(m, \mathbf{c})] \\ \therefore E[edge_i(m, \mathbf{c})] &= 1 - \frac{1}{2}E[margin_i(m, \mathbf{c})] \end{aligned}$$

And:

$$\begin{aligned} Var[margin_i(m, \mathbf{c})] &= Var[1 - 2edge_i(m, \mathbf{c})] \\ &= 4Var[edge_i(m, \mathbf{c})] \\ \therefore Var[edge_i(m, \mathbf{c})] &= \frac{1}{4}Var[margin_i(m, \mathbf{c})] \end{aligned}$$

2.5.1 Current Margin and Edge Hypotheses

The current explanation for boosting's effectiveness is the margin hypothesis. It is widely believed that boosting performs well because it forces low margins to become large. Breiman describes this as 'equalising', whereby high margins (low edges) are sacrificed at the expense of low margins (high edges) to form an improved classifier. Breiman, however, also claims this margin explanation is incomplete. The frontier paper on margin analysis is that by Schapire, Freund, Bartlett and Lee [41]. They show cumulative empirical margin distributions and claim that the minimum (tail) has been pushed up by boosting. Although not clear from the presentation, it can be seen that the variance of these distributions is reduced via boosting - this result is analogous with results on the edge distribution obtained in Section 3.1 of this Thesis.

Now, boosting uses reweighting and forces the base classifier to deal with more difficult (i.e. harder to classify) sections of the input space. This active form of shifting a classifier's focus should result in uncorrelated errors. For example, at iteration 1 a classifier misclassifies observations in a particular section of the data.

Reweighting increases the importance placed on such observations and decreases the relative importance of correctly classified observations. The aim of this is to correctly classify those observations with the new high weights. In subsequent iterations this methodology will lead to zero or negative correlation between the errors made on individual observations by each classifier. Those classifiers with lower weighted error are given a higher voting weight in the combined classifier. Bagging on the other hand does not actively reweight examples but takes a bootstrap sample at each iteration. The original dataset is still disturbed but not as aggressively as boosting. In this case we would not necessarily expect zero or negative correlation between errors on individual classifiers as each observation retains a selection probability of $\frac{1}{N}$.

2.6 Recent Published Work on Boosting and Ensemble Learning

Since its inception in 1995, boosting has been the subject of many theoretical and empirical studies [3, 7, 41, 42]. It is widely recognised that noisy data or patterns are difficult to learn. Such data can result in the overall performance being sacrificed as boosting tries to support increased margins on incorrect classification at the expense of data with already high margins. *AdaBoost* performs gradient descent in a margin related function , concentrating on patterns which are hard to learn. Empirical studies have demonstrated boosting's poorer performance on noisy data [34, 35]. Even the frontier minimum margin paper by Schapire et. al. offers explanation for boosting's superior performance, resulting in test

error bounds involving the fraction of observations with a margin smaller than threshold value. Clearly if a higher proportion of observations were associated with higher margin, poor generalisation error would result. Variations of the standard AdaBoost algorithm have been proposed [31, 35] which balance the number of margin errors (misclassifications) and the magnitude of the margin. These new algorithms involve new parameters which in practice are not easily interpretable. Räetsch et. al. [36] propose a new boosting algorithm which also allows for the presence of a certain amount of noise, using a parameter which represents the fractions of observations permitted to have low margin. The algorithm ν-Arc allows the user to specify the fraction (ν) of observations that are difficult to learn. Such patterns will have lower margins (high edge) and ν-Arc allows for a certain fraction of margins to lie on the incorrect side of the decision boundary. Some of these observations will be misclassified by the combined hypothesis and some will be classified correctly, despite having a small margin. Essentially these low margin observations will have negligible weights when the next stage of hypothesis building is occurring. A linear programming problem motivates ν-Arc, and in such circumstances it is not possible to reduce the influence of noisy data more than penalising them linearly. Once a pattern is recognised as being noisy, its classification by the final model is irrelevant. Rätsch et. al. tolerate noise if it results in benefits to other parts of the data by substantially increasing margin on 'non-noisy' observations.

Chapter 3

Variance Reduction of the Edge

3.1 Empirical Trials on Edge and Margin

It would be beneficial at this point to introduce new notation for the quantity $edge_i(m, \mathbf{c})$ as defined in equation (2.5.1). Since the *AdaBoost* algorithm is used for all trials with known values of the c_m's, the $\mathbf{c}$ term may be dropped and replaced by an i to indicate that the edge values are calculated for each observation. Therefore

$$edge_i(m, \mathbf{c}) \quad = \quad edge(m, i) \tag{3.1.1}$$

For example, $edge(5, 2)$ would be the value of edge calculated for the second observation after 5 iterations.

In all experiments discussed in this Chapter, the decision tree learner $c4.5$ with default options and default pruning was used as the base classifier, with a boosted ensemble being built from $m = 50$ iterations. Datasets tested are a selection from the UCI Machine Learning Repository [2]. The datasets were chosen to provide a representative mixture of size and boosting performance previously reported. The results shown in Figures 3.1 - 3.6 for the variance and mean of the edge

across observations are representative for all UCI datasets tested. Table 3.1 gives a listing of the datasets and their features as used in Thesis.

Table 3.1: Summary of UCI datasets used in this study - as per [34]

dataset	Cases	Classes	Continuous Attributes	Discrete Attributes
bands	512	2	20	20
colic	368	2	10	12
credit	690	2	7	13
glass	214	6	9	0
heart-h	294	2	8	5
heart-c	303	2	8	5
hepatitis	155	2	6	13
letter	20000	26	16	0
census	32000	2	6	8

The original training data was shuffled randomly and split into 10 equal-sized partitions. Each of the ten partitions was used in turn as a test set for the ensemble generated using the remaining 90% of the original data as a training set. The shuffling step was added to ensure randomness in the order of class distributions prior to partitioning.

At each iteration, the values for $edge(m,i)$ were calculated for each observation in the training set. The mean and variance of $edge(m,i)$ were calculated at each iteration over all observations as follows:

$$\hat{E}_i[edge(m,i)] = \frac{1}{N}\sum_{i=1}^{N} edge(m,i)$$

$$\hat{Var}_i[edge(m,i)] = \frac{1}{N}\sum_{i=1}^{N}(edge(m,i) - \hat{E}_i[edge(m,i)])^2$$

Note the i subscript on the expectation and variance calculations indicate that expectation is taken over all observations for each iteration. The results of these

trials appear in graphical form in Figures 3.1, 3.2 and 3.3. These figures show plots of $\hat{E}_i[edge(m,i)]$ and $\hat{Var}_i[edge(m,i)]$ across observations versus the number of boosting trials for the *glass*, *letter* and *colic* datasets. Each plotted symbol represents the result across observations for a single fold of the 10-fold cross-validation trial. Whilst results are only presented for 3 UCI datasets, the results are representative of all datasets tested. The same signature mean and variance behaviour is observed, independent of the dataset being tested.

An apparent exponential decrease in variance is noted for all datasets tested, perhaps indicating an asymptote of zero or some small value, ϵ. This possible limit has not as yet been determined theoretically and these initial results prompt the question "does boosting homogenise the edge?". The most dramatic variance decay is seen in boosting trials $m \leq 5$ i.e. most of the 'hard' work appears to be done in the first few trials. This observation is consistent with several authors noting in earlier published empirical studies that little additional benefit is gained after 10 boosting trials [12, 34]. The presence of a similar increasing trend in the average edge can also be seen in Figures 3.1, 3.2 and 3.3. If the above variance trends are truly exponential, replotting Figures 3.1, 3.2 and 3.3 on a log scale will show a linear trend. Refer to Figures 3.4, 3.5 and 3.6, which exhibit non-linear trends when $\hat{Var}_i[edge(m,i)]$ is plotted on a log scale. Hence this leads to the conclusion that the observed decrease is not strictly exponential.

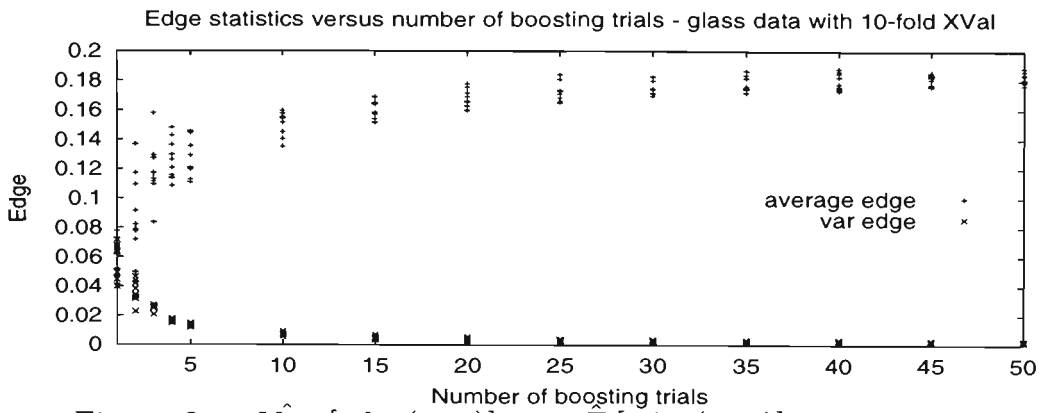

Figure 3.1: $\hat{Var}_i[edge(m,i)]$ and $\hat{E}_i[edge(m,i)]$ vs. number of boosting trials : *glass* data.

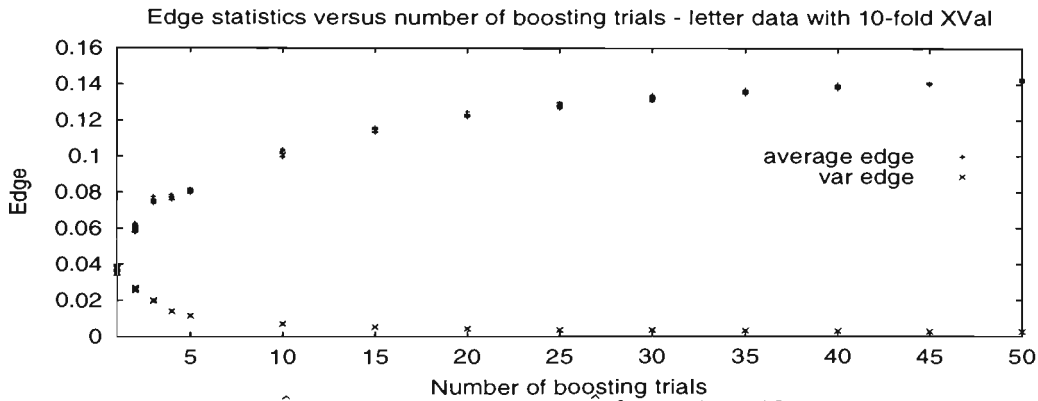

Figure 3.2: $\hat{Var}_i[edge(m,i)]$ and $\hat{E}_i[edge(m,i)]$ vs. number of boosting trials : *letter* data.

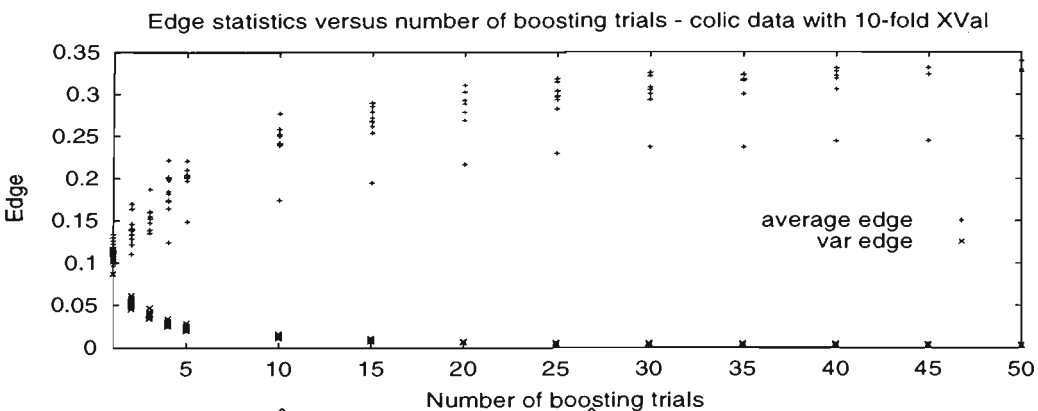

Figure 3.3: $\hat{Var}_i[edge(m,i)]$ and $\hat{E}_i[edge(m,i)]$ vs. number of boosting trials : *colic* data.

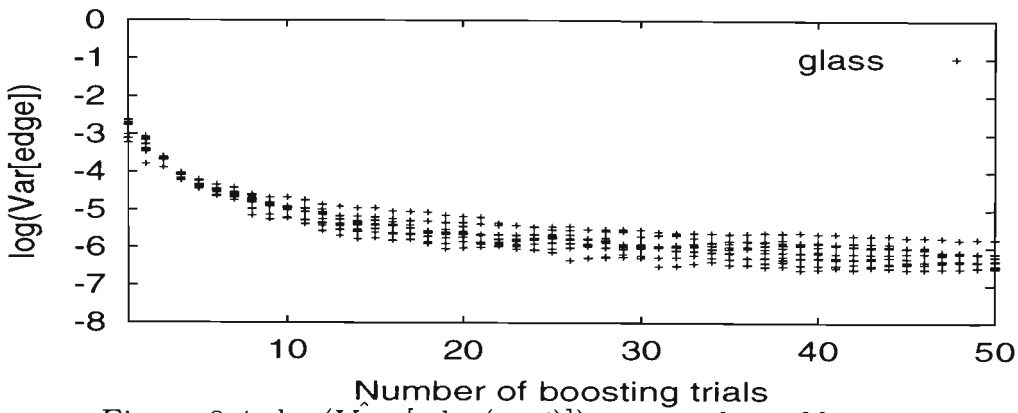

Figure 3.4: $\log(\hat{Var}_i[edge(m,i)])$ vs. number of boosting trials : *glass* data.

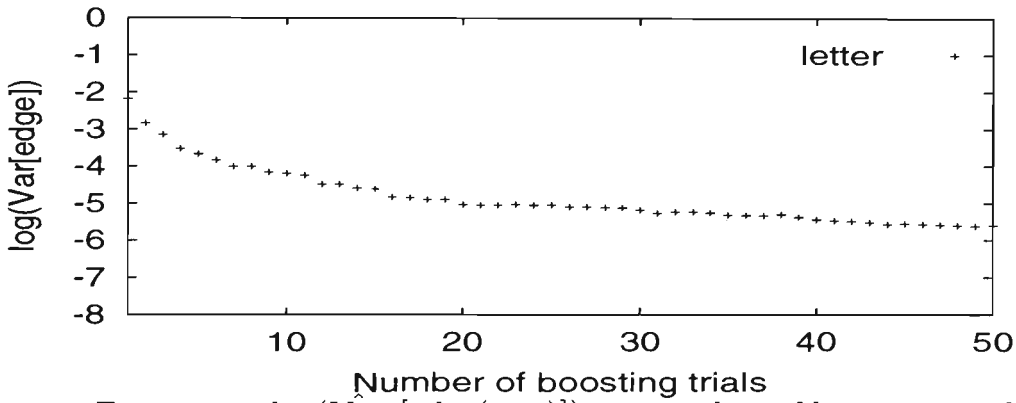

Figure 3.5: $\log(\hat{Var}_i[edge(m,i)])$ vs. number of boosting trials : *letter* data.

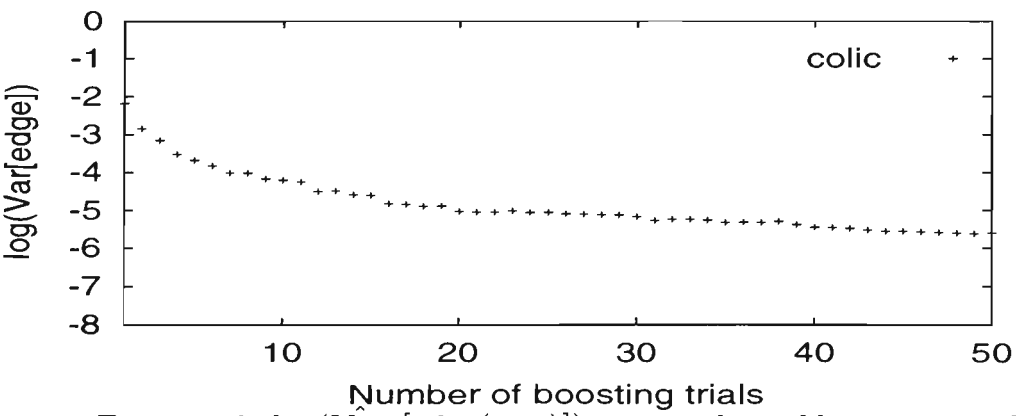

Figure 3.6: $\log(\hat{Var}_i[edge(m,i)])$ vs. number of boosting trials : *colic* data.

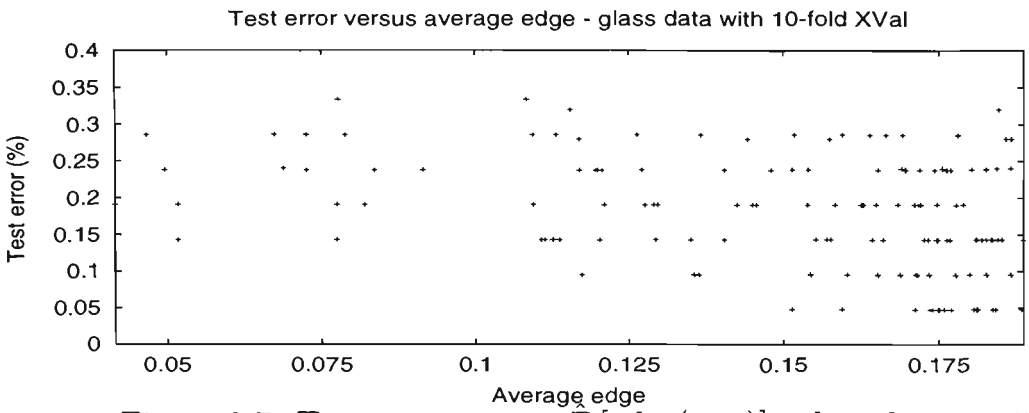

Figure 3.7: Test set error vs. $\hat{E}_i[edge(m,i)]$: *glass* data m=50 iterations.

It has been suggested in recent studies [3, 41] that reduction in test set error may correlate with a reduction in the edge values (equivalent to an increase in margin values). With the exception of the *letter* dataset, plotting the average edge versus test set error for all crossvalidated folds showed no correlation between decreasing edge values and a reduction in test set error for all datasets but the letter dataset. A possible explanation for this may be the larger dataset size and the increased number of response classes. Refer to Figures 3.7, 3.8 and 3.9, where test error was measured as the percentage of incorrect classifications on the 10% leave aside test set in the cross-validation procedure. The correlation coefficients for these plots are -0.134, -0.713 and -0.076 respectively. It was also noted that boosting with equal votes where the vote for each classifier was equal to $\frac{1}{m}$ resulted in a similar 'exponential' decrease in variance of the edge.

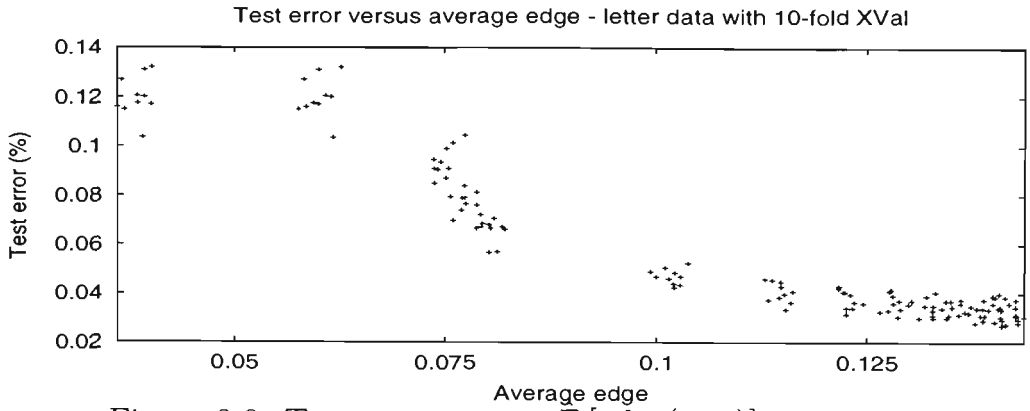

Figure 3.8: Test set error vs. $\hat{E}_i[edge(m,i)]$: *letter* data $m = 50$ iterations.

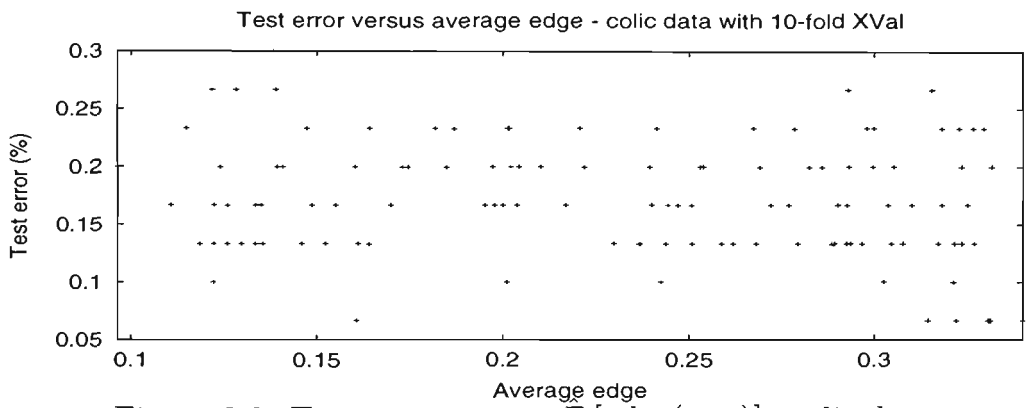

Figure 3.9: Test set error vs. $\hat{E}_i[edge(m,i)]$: *colic* data $m = 50$ iterations.

3.2 Theoretical Analysis of Average Edge Trends

A simplified expression for average edge may be derived as follows:

$$\begin{aligned}
\hat{E}_i[edge(m,i)] &= \hat{E}_i[\frac{1}{\sum c_j}\sum_{j=1}^{m} c_j I_j(i)] \\
&= \frac{1}{\sum_{j=1}^{m}\log(\frac{1}{\beta_j})}\hat{E}_i[\sum_{j=1}^{m}\log(\frac{1}{\beta_j})I_j(i))] \\
&= \frac{1}{\sum_{j=1}^{m}\log(\frac{1}{\beta_j})}\sum_{j=1}^{m}\log(\frac{1}{\beta_j})\hat{E}_i[I_j(i)]
\end{aligned}$$

Now, define e_j as $\hat{E}_i[I_j(i)]$, giving:

$$\hat{E}_i[edge(m,i)] = \frac{1}{\sum_{j=1}^{m} \log(\frac{1}{\beta_j})} \sum_{j=1}^{m} \log(\frac{1}{\beta_j}) e_j \tag{3.2.1}$$

Empirical results from Section 3.1 show an increase in average edge as the number of boosting trials (m) increases (Figures 3.1 - 3.3).

The condition for this to occur is $\hat{E}_i[edge(m,i)] \leq e_{m+1}$ and is derived below.

$$\begin{aligned}
\hat{E}_i[edge(m+1,i)] &= \frac{1}{\sum_{j=1}^{m+1} \log(\frac{1}{\beta_j})} \sum_{j=1}^{m+1} \log(\frac{1}{\beta_j}) e_j \\
&= \frac{\hat{E}_i[edge(m,i)] \times \sum_{j=1}^{m+1} log(\frac{1}{\beta_j}) e_j}{\hat{E}_i[edge(m,i)] \sum_{j=1}^{m+1} \log(\frac{1}{\beta_j})} \\
&= \frac{\hat{E}_i[edge(m,i)] \times \sum_{j=1}^{m+1} log(\frac{1}{\beta_j}) e_j}{\hat{E}_i[edge(m,i)] \sum_{j=1}^{m} \log(\frac{1}{\beta_j}) + \hat{E}_i[edge(m,i)] \log(\frac{1}{\beta_{m+1}})} \\
&= \hat{E}_i[edge(m,i)] \times \frac{\sum_{j=1}^{m+1} log(\frac{1}{\beta_j}) e_j}{\sum_{j=1}^{m} \log(\frac{1}{\beta_j}) e_j + \hat{E}_i[edge(m,i)] \log(\frac{1}{\beta_{m+1}})} \\
&= \hat{E}_i[edge(m,i)] \times factor_m
\end{aligned}$$

For $\hat{E}_i[edge(m+1,i)] \geq \hat{E}_i[edge(m,i)]$, $factor_m \geq 1$ and since numerator and denominator are both positive, the condition becomes:

$$\begin{aligned}
\sum_{j=1}^{m+1} \log(\frac{1}{\beta_j}) e_j &\geq \sum_{j=1}^{m} \log(\frac{1}{\beta_j}) e_j + \hat{E}_i[edge(m,i)] \log(\frac{1}{\beta_{m+1}}) \\
\log(\frac{1}{\beta_{m+1}}) e_{m+1} &\geq \hat{E}_i[edge(m,i)] \log(\frac{1}{\beta_{m+1}}) \\
\hat{E}_i[edge(m,i)] &\leq e_{m+1}
\end{aligned}$$

Therefore the average edge will increase between the m-th and $(m+1)$-th boosting trials if the unweighted error of the $(m+1)$-th trial is greater than or equal to the average edge calculated at the m-th trial. Empirically it is seen that $max(\hat{E}_i[edge(m,i)]) \leq max(e_m)$, where the maximum is taken over estimated edges for all observations at iteration m. Generally the average edge increases

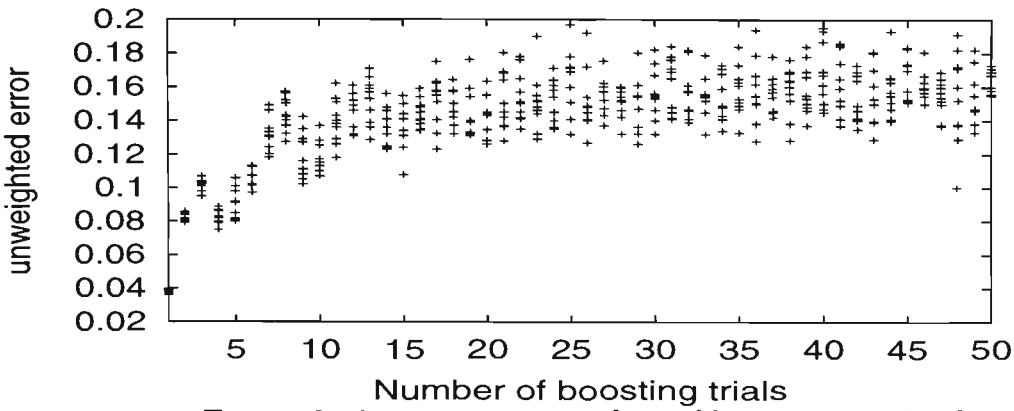

Figure 3.10: e_m versus number of boosting trials. *letter* data.

but stays below a threshold of the maximum unweighted error of hypotheses indicating that the maximum unweighted error is an upper bound on average edge.

Empirical evidence from this study indicates that $e_i < e_{m+1}$ $(i \leq m)$, with the results for the *letter* dataset given in Figure 3.10. This plot is indicative of e_m trends observed for other datasets tested i.e. the training set error, e_i appears to be increasing.

3.3 Theoretical Analysis of Variance Reduction Trends

We have seen empirical results of variance reduction trends for the edge when boosting is applied (Section 3.1). It would be beneficial to quantify these empirical observations and develop theory to give structured representation to the empirical results observed to date.

An alternative expression for $\hat{Var}_i[edge(m,i)]$ is derived below. Firstly, the definition of edge follows that given in equation (2.5.1) [3] and the new notation

introduced in Section 3.1 in equation (3.1.1). Retaining the definition of c_j, an expression of $edge(m,i)$ is given as:

$$\begin{aligned} edge(m,i) &= \sum_{j=1}^{m} \frac{c_j I[h(\mathbf{x}_i) \neq y_i]}{\sum c_j} \\ &= \frac{1}{\sum c_j} \sum_{j=1}^{m} c_j I[h(\mathbf{x}_i) \neq y_i] \end{aligned}$$

Let e_j represent the unweighted error of the j-th hypothesis and ϵ_j the weighted error of the j-th hypothesis. i.e.

$$\begin{aligned} e_j &= \frac{1}{n} \sum_{i=1}^{n} I[h(\mathbf{x}_i) \neq y_i] \\ \epsilon_j &= \sum_{i=1}^{n} I[h(\mathbf{x}_i) \neq y_i] w_i^j \end{aligned}$$

The following elementary statistical rules are used in the variance derivation:

- $\hat{Var}[X_1 + X_2 + \ldots X_n] = \hat{Var}[\sum X_i] = \sum \hat{Var}[X_i] + 2 \sum \sum_{i<j} \hat{Cov}[X_i, X_j]$
- $\hat{Var}[aX] = a^2 \hat{Var}[X]$
- $\hat{Cov}[aX, bY] = ab\hat{Cov}[X, Y]$
- $\hat{Cov}[X, Y] = \hat{E}[XY] - \hat{E}[X]\hat{E}[Y]$

So, letting $I[h_j(\mathbf{x}_i) \neq y_i] = I_j(i)$:

$$\begin{aligned}
\hat{Var}_i[edge(m,i)] &= \hat{Var}_i[\frac{1}{\sum c_j}\sum_{j=1}^{m} c_j I_j(i)] \\
&= \frac{1}{(\sum c_j)^2}\hat{Var}_i[\sum_{j=1}^{m} c_j I_j(i)] \\
&= \frac{1}{(\sum c_j)^2}\left(\sum_{j=1}^{m}\hat{Var}_i[c_j I_j(i)] + 2\sum\sum_{j<k}^{m}\hat{Cov}_i[c_j I_j(i), c_k I_k(i)]\right) \\
&= \frac{1}{(\sum c_j)^2}\left(\sum_{j=1}^{m} c_j^2\hat{Var}_i[I_j(i)] + 2\sum\sum_{j<k}^{m} c_j c_k\hat{Cov}_i[I_j(i), I_k(i)]\right)
\end{aligned}$$

Now for *AdaBoost* , $c_j = \log(\frac{1-\epsilon_j}{\epsilon_j}) = \log(\frac{1}{\beta_j})$.

Therefore,

$$\begin{aligned}
\hat{Var}_i[edge(m,i)] &= \frac{\sum_{j=1}^{m}(\log\frac{1}{\beta_j})^2\hat{Var}_i[I_j(i)] + 2\sum\sum_{j<k}^{m}\log(\frac{1}{\beta_j})\log(\frac{1}{\beta_k})\hat{Cov}_i[I_j(i),I_k(i)]}{(\sum_{j=1}^{m}\log\frac{1}{\beta_j})^2} \\
&= \frac{\sum_{j=1}^{m}(\log\beta_j)^2\hat{Var}_i[I_j(i)] + 2\sum\sum_{j<k}^{m}\log\beta_j\log\beta_k\hat{Cov}_i[I_j(i),I_k(i)]}{(\sum_{j=1}^{m}\log\beta_j)^2}
\end{aligned} \tag{3.3.1}$$

Now, $I_j(i)$ is a Bernoulli random variable with parameter $p = e_j$, where e_j represents the unweighted error of the j-th hypothesis. Therefore, $\hat{Var}_i[I_j(i)] = e_j(1-e_j)$, and

$$\begin{aligned}
\hat{Var}_i[edge(m,i)] &= \frac{\sum_{j=1}^{m}(\log\beta_j)^2 e_j(1-e_j)}{(\sum_{j=1}^{m}\log\beta_j)^2} \\
&+ \frac{2\sum\sum_{j<k}^{m}\log\beta_j\log\beta_k\hat{Cov}_i[I_j(i),I_k(i)]}{(\sum_{j=1}^{m}\log\beta_j)^2}
\end{aligned} \tag{3.3.2}$$

But $\hat{Cov}_i[I_j(i), I_k(i)] = \hat{E}_i[I_j(i)I_k(i)] - \hat{E}_i[I_j(i)]\hat{E}_i[I_k(i)]$.

Therefore,

$$\begin{aligned}
\hat{Var}_i[edge(m,i)] &= \frac{\sum_{j=1}^{m}(\log\beta_j)^2 e_j(1-e_j)}{(\sum_{j=1}^{m}\log\beta_j)^2} + \frac{2\sum\sum_{j<k}^{m}\log\beta_j\log\beta_k\hat{E}_i[I_j(i)I_k(i)]}{(\sum_{j=1}^{m}\log\beta_j)^2} \\
&- \frac{2\sum\sum_{j<k}^{m}\log\beta_j\log\beta_k e_j e_k}{(\sum_{j=1}^{m}\log\beta_j)^2}
\end{aligned} \tag{3.3.3}$$

The expression derived for $\hat{Var}_i[edge(m,i)]$ in equation (3.3.3) is dependent only

on ϵ_j, e_j and $\hat{E}_i[I_j(i)I_k(i)]$. It is unlikely that there is a simple expression for $\hat{E}_i[I_j(i)I_k(i)]$, where $I_j(i), I_k(i)$ are Bernoulli random variables, since the expectation depends on the interactions of errors of hypotheses $h_j(\mathbf{x})$ and $h_k(\mathbf{x})$. However, it may be possible to determine bounds on this expectation.

Intuitively this result makes sense, since at each iteration, the learner attempts to correctly predict observations that were predicted incorrectly at the previous iteration. For this to happen, the indicator variables for unweighted error should be negatively correlated for pair-wise iterations. If errors were positively correlated, voting could degrade performance since individual hypotheses may consistently vote incorrectly on some observations and never be given the chance to explore different areas of the training set. Hence, from the expression derived, negative or zero covariance terms will result in non-increasing values for $\hat{Var}_i[edge(m,i)]$. This equates to the predictions of the $h_j(\mathbf{x})$'s being negatively correlated or uncorrelated, which is equivalent to Breiman's recent notion of "equalizing" [7].

3.3.1 Examining Components of $\frac{\sum_{j=1}^{m}(\log\beta_j)^2 e_j(1-e_j)}{(\sum_{j=1}^{m}\log\beta_j)^2}$

In empirically examining the $\frac{\sum_{j=1}^{m}(\log\beta_j)^2 e_j(1-e_j)}{(\sum_{j=1}^{m}\log\beta_j)^2}$ term in equation (3.3.3) , the following was noted:

- $\log\beta_m$ shows no trending as m increases for the glass and colic datasets, but shows variance reduction and possible cycles for letter. Refer to Figures 3.11, 3.12 and 3.13. Note, however, that the values of β_j are highly variable for the *colic* dataset. This is an interesting result as boosting degrades

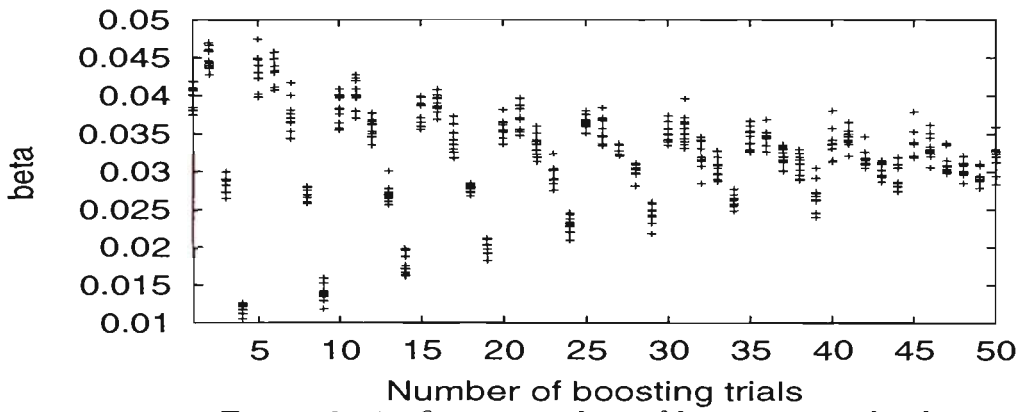

Figure 3.11: β_j vs. number of boosting trials: *letter* data

performance on this dataset.

- $\sum_{j=1}^{m} \log \beta_j$ is linear in m, suggesting that $\log \beta_j$ is constant.

- Since $\log \beta_j$ appears to be constant, $\frac{1}{(\sum_{j=1}^{m} \log \beta_j)^2}$ is strongly a $\frac{1}{m^2}$ type curve. Hence this normalising factor has a strong decaying effect. Refer to Figure 3.14. This is akin to the central limit theorem in statistics, which seems to apply here since boosting forms a weighted average of the individual classifications and differs only in the dependence between the classifiers, a feature of boosted ensembles.

- The $(\log \beta_j)^2 e_j(1-e_j)$ term exhibits quite a random scatter with no trending as m increases for the *glass* and *colic* datasets but a trend is detected for the *letter* dataset. Refer to Figures 3.15 and 3.16.

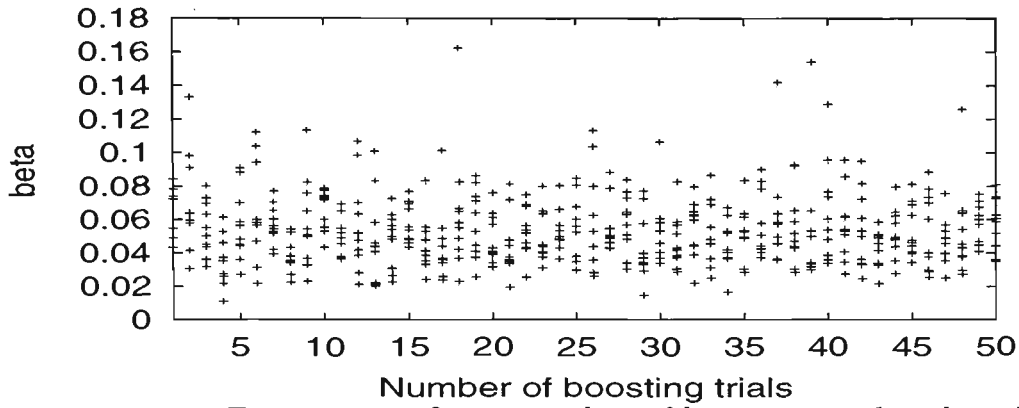

Figure 3.12: β_j vs. number of boosting trials: *glass* data

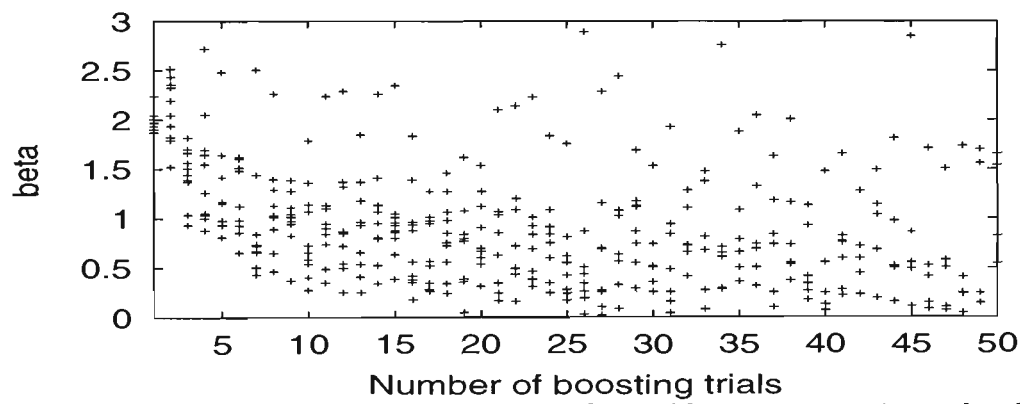

Figure 3.13: β_j vs. number of boosting trials: *colic* data

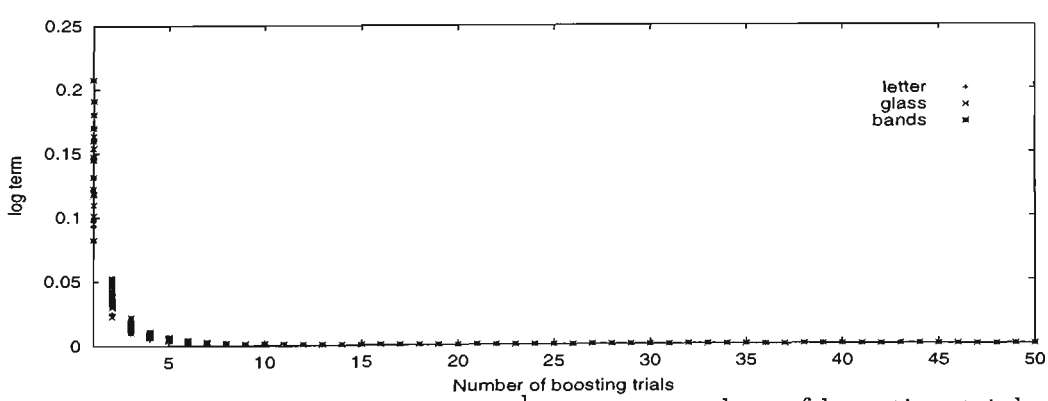

Figure 3.14: $\frac{1}{(\sum \log \beta_j)^2}$ vs. number of boosting trials.

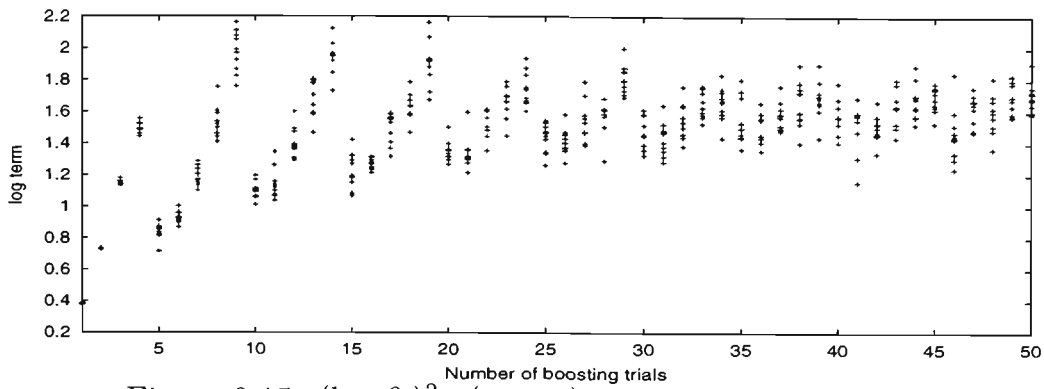

Figure 3.15: $(\log \beta_j)^2 e_j(1-e_j)$ vs. number of boosting trials: *letter* data.

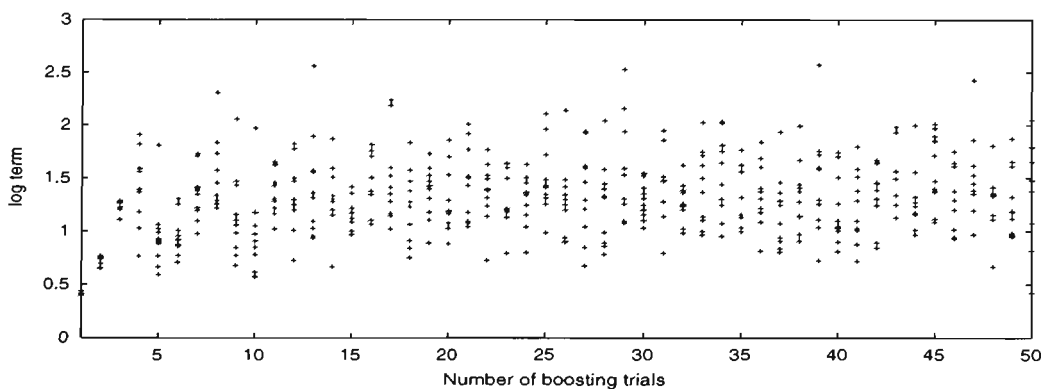

Figure 3.16: $(\log \beta_j)^2 e_j(1-e_j)$ vs. number of boosting trials: *glass* data.

3.3.2 Proving that $\hat{Var}_i[edge(m,i)]$ is a Monotonic Non-Increasing Function in m.

An alternative expression for $\hat{Var}_i[edge(m,i)]$ has been derived in equation (3.3.2) as:

$$\hat{Var}_i[edge(m,i)] = \frac{\sum_{j=1}^{m}(\log\beta_j)^2 e_j(1-e_j) + 2\sum\sum_{j<k}^{m}\log\beta_j\log\beta_k\hat{Cov}_i[I_j(i),I_k(i)]}{(\sum_{j=1}^{m}\log\beta_j)^2}$$

To prove that this is a monotonic non-increasing function in m, it must be shown that $\hat{Var}_i[edge(m,i)] \geq \hat{Var}_i[edge(m+1,i)]\ \forall m \geq 1$.

$$\begin{aligned}
\hat{Var}_i[edge(m+1,i)] &= \frac{\sum_{j=1}^{m+1}(\log\beta_j)^2 e_j(1-e_j)+2\sum\sum_{j<k}^{m+1}\log\beta_j\log\beta_k\hat{Cov}_i[I_j(i),I_k(i)]}{(\sum_{j=1}^{m+1}\log\beta_j)^2} \\
&= \frac{\sum_{j=1}^{m}(\log\beta_j)^2 e_j(1-e_j)+2\sum\sum_{j<k}^{m}\log\beta_j\log\beta_k\hat{Cov}_i[I_j(i),I_k(i)]}{(\sum_{j=1}^{m+1}\log\beta_j)^2} \\
&+ \frac{(\log\beta_{m+1})^2 e_{m+1}(1-e_{m+1})+2\sum_{j=1}^{m}\log\beta_j\log\beta_{m+1}\hat{Cov}_i[I_j(i),I_{m+1}(i)]}{(\sum_{j=1}^{m+1}\log\beta_j)^2} \\
&= \frac{(\sum_{j=1}^{m}\log\beta_j)^2}{(\sum_{j=1}^{m+1}\log\beta_j)^2}\hat{Var}_i[edge(m,i)] \\
&+ \frac{(\log\beta_{m+1})^2 e_{m+1}(1-e_{m+1})+2\log\beta_{m+1}\sum_{j=1}^{m}\log\beta_j\hat{Cov}_i[I_j(i),I_{m+1}(i)]}{(\sum_{j=1}^{m+1}\log\beta_j)^2} \\
&= C_m\hat{Var}_i[edge(m,i)] + A_m
\end{aligned}$$

Ideally, we seek the distribution of A_m, which is essentially a cross-product covariance term and difficult to conceptualise. Hence alternative expansions for $edge(m+1,i)$ and $\hat{Var}_i[edge(m,i)]$ are developed below.

$$\begin{aligned}
edge(m+1,i) &= \frac{\sum_{j=1}^{m+1}\log\beta_j I_j(i)}{\sum_{j=1}^{m+1}\log\beta_j} \\
&= \frac{\sum_{j=1}^{m}\log\beta_j I_j(i)+\log\beta_{m+1}I_{m+1}(i)}{\sum_{j=1}^{m+1}\log\beta_j} \\
&= \left(\frac{\sum_{j=1}^{m}\log\beta_j}{\sum_{j=1}^{m+1}\log\beta_j}\right) edge(m,i) + \frac{\log\beta_{m+1}I_{m+1}(i)}{\sum_{j=1}^{m+1}\log\beta_j} \\
&= \left(1 - \frac{\log\beta_{m+1}}{\sum_{j=1}^{m+1}\log\beta_j}\right) edge(m,i) + \frac{\log\beta_{m+1}I_{m+1}(i)}{\sum_{j=1}^{m+1}\log\beta_j}
\end{aligned} \tag{3.3.4}$$

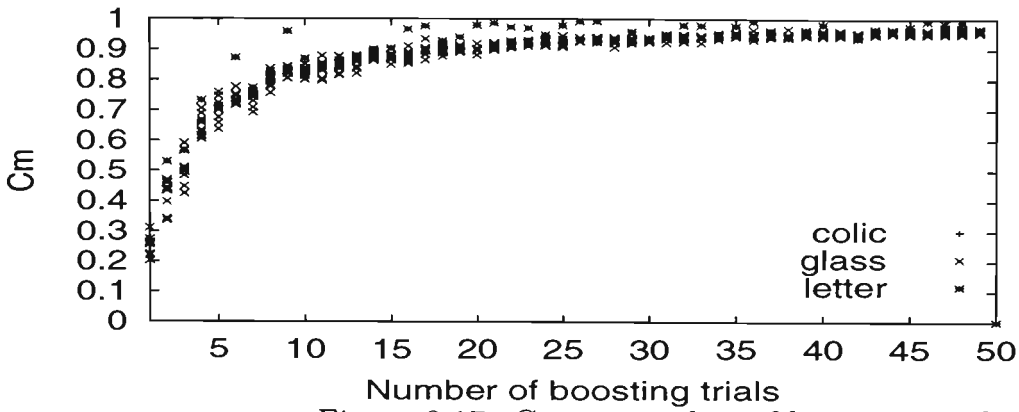

Figure 3.17: C_m vs. number of boosting trials.

Now,

$$\begin{aligned}
\hat{Var}_i[edge(m+1,i)] &= \left(1 - \frac{\log\beta_{m+1}}{\sum_{j=1}^{m+1}\log\beta_j}\right)^2 \hat{Var}_i[edge(m,i)] \\
&+ \frac{(\log\beta_{m+1})^2 e_{m+1}(1-e_{m+1})}{(\sum_{j=1}^{m+1}\log\beta_j)^2} \\
&+ \left(\frac{\sum_{j=1}^{m}\log\beta_j}{\sum_{j=1}^{m+1}\log\beta_j}\right)\left(\frac{\log\beta_{m+1}}{\sum_{j=1}^{m+1}\log\beta_j}\right) \hat{Cov}_i[I_{m+1}(i), edge(m,i)]
\end{aligned} \tag{3.3.5}$$

The variance expression derived in equation (3.3.5) differs from that derived in equation (3.3.2) in its expression of the A_m term. The latter two terms in equation (3.3.5) are equal to the A_m term in equation (3.3.2).

The C_m term is equal in both cases and is clearly less than 1, approaching 1 as m increases. This quantity is plotted as Figure 3.17.

The new expansion derived as equation (3.3.5) contains terms which may be used to explain the variance reduction behaviour. The second term in equation (3.3.5) behaves as a $O(\frac{1}{m^2})$ term and will approach zero rapidly as m increases. The covariance term will be neither strictly positive or negative but given the insights we have on boosting, we expect this term to be negative and larger in earlier iterations as boosting does the majority of the difficult work in early iterations

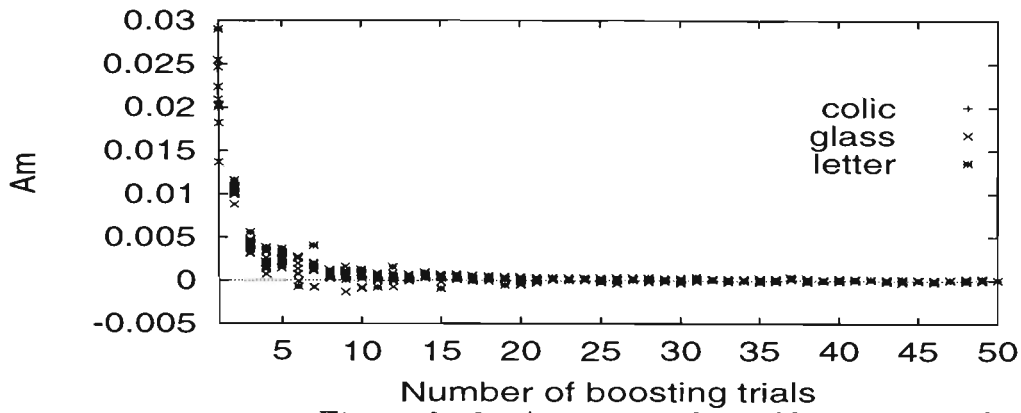

Figure 3.18: A_m vs. number of boosting trials.

as it attempts to 'balance' misclassifications across observations. A decrease in this covariance is noted in later iterations as mathematical restrictions become influential when the number of pair-wise covariances increases. This notion is reinforced in Section 3.4.2 where we observe the covariance (correlation) term having less effect as the number of boosting iterations increases (Refer to Figure 3.22). The edge variance decreases dramatically in the first few iterations then varies little as the covariance term is less significant and the $O(\frac{1}{m^2})$ term becomes smaller and smaller.

The above discussion is an intuitive proof for the variance reduction observed empirically. An exact form of the covariance term is intractable due to the nature of the boosting algorithm and edge values being dependent on the misclassification or otherwise of each observation at each iteration. A plot of the A_m term is included as Figure 3.18 where we observe A_m approaching 0 in the limit.

3.3.3 Developing the Recurrence Relation for $\hat{Var}_i[edge(m+1,i)]$

In Section 3.3.2, the following recurrence relation was developed for $\hat{Var}_i[edge(m+1,i)]$.

$$\hat{Var}_i[edge(m+1,i)] = C_m \hat{Var}_i[edge(m,i)] + A_m \tag{3.3.6}$$

But:

$$\begin{aligned}
C_m \hat{Var}_i[edge(m,i)] + A_m &= C_m \left(C_{m-1} \hat{Var}_i[edge(m-1,i)] + A_{m-1} \right) + A_m \\
&= C_m C_{m-1} \left(C_{m-2} \hat{Var}_i[edge(m-2,i)] + A_{m-2} \right) \\
&+ C_m A_{m-1} + A_m \\
&\vdots \\
&= \hat{Var}_i[edge(1,i)] \prod_{l=1}^{m} C_l + \frac{\sum_{l=1}^{m} A_l (\sum_{j=1}^{l+1} \log \beta_j)^2}{(\sum_{j=1}^{m+1} \log \beta_j)^2} \\
&= \hat{Var}_i[edge(1,i)] \prod_{l=1}^{m} \frac{(\sum_{j=1}^{l} \log \beta_j)^2}{(\sum_{j=1}^{l+1} \log \beta_j)^2} + \frac{\sum_{l=1}^{m} A_l (\sum_{j=1}^{l+1} \log \beta_j)^2}{(\sum_{j=1}^{m+1} \log \beta_j)^2} \\
&= \frac{e_1(1-e_1)(\log \beta_1)^2}{(\sum_{j=1}^{m+1} \log \beta_j)^2} + \frac{\sum_{l=1}^{m} A_l (\sum_{j=1}^{l+1} \log \beta_j)^2}{(\sum_{j=1}^{m+1} \log \beta_j)^2} \\
&= \frac{e_1(1-e_1)(\log \beta_1)^2 + \sum_{l=1}^{m} A_l (\sum_{j=1}^{l+1} \log \beta_j)^2}{(\sum_{j=1}^{m+1} \log \beta_j)^2}
\end{aligned}$$

And so,

$$\hat{Var}_i[edge(m,i)] = \frac{e_1(1-e_1)(\log \beta_1)^2 + \sum_{l=1}^{m-1} A_l (\sum_{j=1}^{l+1} \log \beta_j)^2}{(\sum_{j=1}^{m} \log \beta_j)^2}$$

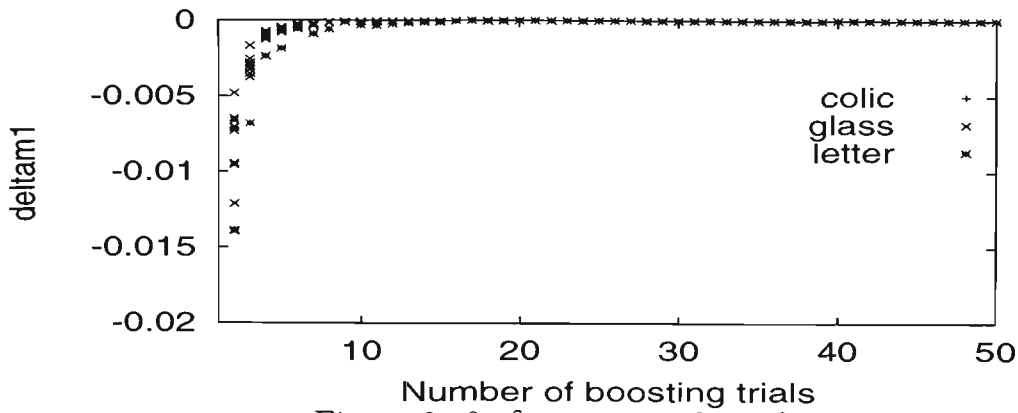

Figure 3.19: $\delta_{m,1}$ vs. number of boosting trials.

Let $\delta_m = \hat{Var}_i[edge(m+1,i)] - \hat{Var}_i[edge(m,i)]$.

$$
\begin{aligned}
\delta_m &= \frac{e_1(1-e_1)(\log\beta_1)^2 + \sum_{l=1}^{m} A_l(\sum_{j=1}^{l+1}\log\beta_j)^2}{(\sum_{j=1}^{m+1}\log\beta_j)^2} \\
&- \frac{e_1(1-e_1)(\log\beta_1)^2 + \sum_{l=1}^{m-1} A_l(\sum_{j=1}^{l+1}\log\beta_j)^2}{(\sum_{j=1}^{m}\log\beta_j)^2} \\
&= e_1(1-e_1)(\log\beta_1)^2\left(\frac{1}{(\sum_{j=1}^{m+1}\log\beta_j)^2} - \frac{1}{(\sum_{j=1}^{m}\log\beta_j)^2}\right) \\
&+ \frac{\sum_{l=1}^{m} A_l(\sum_{j=1}^{l+1}\log\beta_j)^2}{(\sum_{j=1}^{m+1}\log\beta_j)^2} - \frac{\sum_{l=1}^{m-1} A_l(\sum_{j=1}^{l+1}\log\beta_j)^2}{(\sum_{j=1}^{m}\log\beta_j)^2} \\
&= e_1(1-e_1)(\log\beta_1)^2\left(\frac{1}{(\sum_{j=1}^{m+1}\log\beta_j)^2} - \frac{1}{(\sum_{j=1}^{m}\log\beta_j)^2}\right) \\
&+ \sum_{l=1}^{m-1} A_l(\sum_{j=1}^{l+1}\log\beta_j)^2\left(\frac{1}{(\sum_{j=1}^{m+1}\log\beta_j)^2} - \frac{1}{(\sum_{j=1}^{m}\log\beta_j)^2}\right) + A_m \\
&= \delta_{m,1} + \delta_{m,2} + A_m
\end{aligned}
$$

The $\delta_{m,1}$ and $\delta_{m,2}$ quantities are plotted for the *glass*, *colic* and *letter* datasets as Figures 3.19-3.20.

Now,

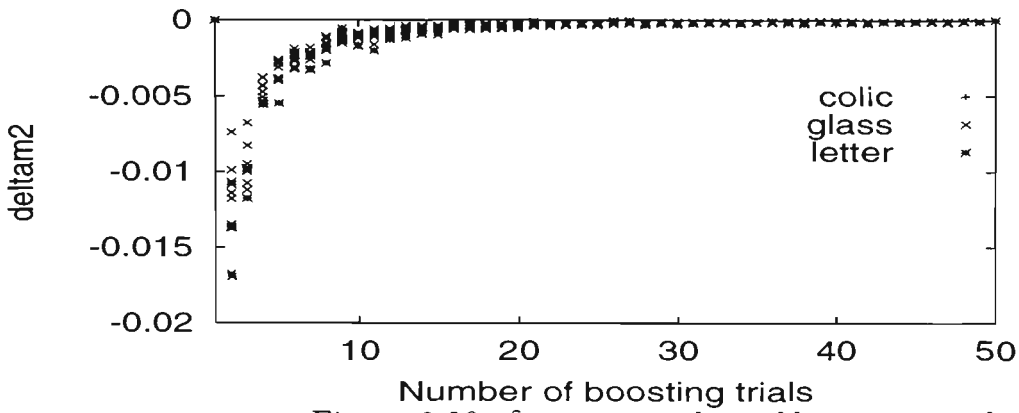

Figure 3.20: $\delta_{m,2}$ vs. number of boosting trials.

$$\begin{aligned}\delta_{m,1} &= e_1(1-e_1)\left(\frac{1}{(\sum_{j=1}^{m+1}\log\beta_j)^2}-\frac{1}{(\sum_{j=1}^{m}\log\beta_j)^2}\right)\\ &< 0\end{aligned}$$

An upper bound for $\delta_{m,2}$ is obtained as follows:

$$\begin{aligned}\hat{Var}_i[edge(m+1,i)]-\hat{Var}_i[edge(m,i)] &= \delta_{m,1}+\delta_{m,2}+A_m\\ &= \delta_{m,1}+\delta_{m,2}+\hat{Var}_i[edge(m+1,i)]\\ &- C_m\hat{Var}_i[edge(m,i)]\\ \hat{Var}_i[edge(m,i)](C_m-1) &= \delta_{m,1}+\delta_{m,2}\\ 0 &> \delta_{m,1}+\delta_{m,2}\\ \delta_{m,2} &< -\delta_{m,1}\\ \delta_{m,2} &< \|\delta_{m,1}\|\end{aligned}$$

This upper bound for $\delta_{m,2}$ is loose and does not give any indication of the sign of $\delta_{m,2}$. However, we know $\delta_{m,1}+\delta_{m,2}<0$ and again the complex A_m term is key in determining the sign of δ_m. We have already seen that the A_m term is equivalent to the last two terms in equation (3.3.5) and are also plotted in Figure 3.18. An

intuitive proof has already been given for A_m to approach zero in the limit which then suggests $\delta_m < 0$ as desired for a variance reduction.

3.4 General Forms of Voting Systems

Mathematical analysis of variance reduction may be simplified by considering general forms of voting systems. This may also allow the variance to be partitioned into components pertaining to the voting mechanism and those pertaining to the method of formation of a sequence of classifiers. In boosting, consecutive classifiers are formed via an adaptive procedure but for bagging they are formed via a sequence of bootstrap replicates. Examples of possible schemes to consider are :

- Scenario 1: All m classifiers make identical predictions at each iteration and hence have the same individual error rates with $\hat{corr}_i[I_j(i), I_k(i)] = 1$. The voting weight of the j-th classifier $= c_j$ (In this case c_j is normalised so that $\sum_{j=1}^{m} c_j = 1$).

- Scenario 2: The m classifiers do not make identical predictions at each iteration but have the same individual error rates with $\hat{corr}_i[I_j(i), I_k(i)] = \rho$. $(\frac{-1}{m-1} \leq \rho < 1)$ Voting weight of the j-th classifier $= \frac{1}{m}$.

- Scenario 3: The m classifiers do not make identical predictions at each iteration but have the same individual error rates with $\hat{corr}_i[I_j(i), I_k(i)] = \rho$ $(\frac{-1}{m-1} \leq \rho < 1)$. Voting weight of the j-th classifier $= c_j$ (In this case c_j is normalised so that $\sum_{j=1}^{m} c_j = 1$).

- Scenario 4: The m classifiers do not make identical predictions at each iteration and have differing individual error rates with $\hat{corr}_i[I_j(i), I_k(i)] = \rho$ ($\frac{-1}{m-1} \leq \rho < 1$). Voting weight of the j-th classifier $= c_j$ (In this case c_j is normalised so that $\sum_{j=1}^{m} c_j = 1$).

The variance expression, $\hat{Var}_i[edge(m, i)]$ from equation (3.3.3) for each of these scenarios is developed in the following sections.

3.4.1 Scenario 1: All m classifiers make identical predictions at each iteration and hence have the same individual error rates with $\hat{corr}_i[I_j(i), I_k(i)] = 1$.

If all m classifiers make identical predictions at each iteration, all m classifiers will have the same individual error rates with $\hat{corr}_i[I_j(i), I_k(i)] = 1$. Voting weight of the j-th classifier is equal to c_j ($\sum_{j=1}^{m} c_j = 1$) and hence:

- $e_j = e_k = e$
- $\hat{Var}_i[I_j(i)] = \hat{Var}_i[I_k(i)] = e(1 - e)$
- $\hat{corr}_i[I_j(i), I_k(i)] = 1$
- $\hat{Cov}_i[I_j(i), I_k(i)] = e(1 - e)$

$$
\begin{aligned}
\hat{Var}_i[edge(m,i)] &= \sum_{j=1}^{m}(c_j)^2 e_j(1-e_j) + 2\sum\sum_{j<k}^{m} c_j c_k \hat{Cov}_i[I_j(i), I_k(i)] \\
&= \sum_{j=1}^{m}(c_j)^2 e(1-e) + 2\sum\sum_{j<k}^{m} c_j c_k e(1-e) \\
&= e(1-e)\sum_{j=1}^{m}(c_j)^2 + 2e(1-e)\sum\sum_{j<k}^{m} c_j c_k \\
&= e(1-e)\left(\sum_{j=1}^{m}(c_j)^2 + 2\sum\sum_{j<k} c_j c_k\right) \\
&= e(1-e)(\sum_{j=1}^{m} c_j)^2 \\
&= e(1-e)
\end{aligned}
$$

This expression for variance is constant and independent of m. Therefore, if this type of voting scheme was employed, no reduction in the variance of edge values would occur. This scenario is equivalent to a single classifier ensemble being built and hence does not apply to boosting.

3.4.2 Scenario 2: The m classifiers do not make identical predictions at each iteration but have the same individual error rates with $\hat{corr}_i[I_j(i), I_k(i)] = \rho$. Voting weight of the j-th classifier $= \frac{1}{m}$

In this scenario, the m classifiers do not make identical predictions at each iteration but have the same individual error rates, in which case, $\hat{corr}_i[I_j(i), I_k(i)] = \rho$ ($\frac{-1}{m-1} \leq \rho < 1$). The voting weight for all classifiers is equal to $\frac{1}{m}$. According to Breiman's recent work [7], this seems to be the most likely scenario - i.e. "equalizing" where the training set misclassification rate is nearly constant across the training set i.e. the e_j's are equalised. This perhaps explains why such a good

match is obtained when overlaying empirical results onto the plot of theoretical variance obtained in Scenario 2. (refer to Figure 3.22). In this case:

- $c_j = \frac{1}{m}$
- $e_j = e_k = e$
- $\hat{Var}_i[I_j(i)] = \hat{Var}_i[I_k(i)] = e(1-e)$
- $\hat{corr}_i[I_j(i), I_k(i)] = \rho$
- $\hat{Cov}_i[I_j(i), I_k(i)] = \rho e(1-e)$

Under these conditions, edge values would vary between 0 and 1 and we would not expect the variance of these edge values to be constant as m increases.

Now,

$$\begin{aligned}
\hat{Var}_i[edge(m,i)] &= \tfrac{1}{m^2}\left(\textstyle\sum_{j=1}^{m} e_j(1-e_j) + 2\sum\sum_{j<k}^{m} \hat{Cov}_i[I_j(i), I_k(i)]\right) \\
&= \tfrac{1}{m^2}\left(\textstyle\sum_{j=1}^{m} e(1-e) + 2\sum\sum_{j<k}^{m} \rho e(1-e)\right) \\
&= \tfrac{1}{m^2}\left(me(1-e) + 2\rho e(1-e)\sum\sum_{j<k}^{m} 1\right) \qquad (3.4.1) \\
&= \tfrac{1}{m^2}\left(me(1-e) + 2\rho e(1-e)\tfrac{m(m-1)}{2}\right) \\
&= \tfrac{e(1-e)}{m}\left(1 + \rho(m-1)\right)
\end{aligned}$$

The first term in this expression may be considered the voting component and the second term involving ρ the component pertaining to the method of classifier formation. Figure 3.21 shows the value of this variance expression with e fixed at 0.04 and ρ varying. ($\frac{-1}{1-m} \leq \rho \leq 1$). The theoretical justification for ρ having a lower limit of $\frac{-1}{m-1}$ is given in [28]. For this reason, it can be seen that the

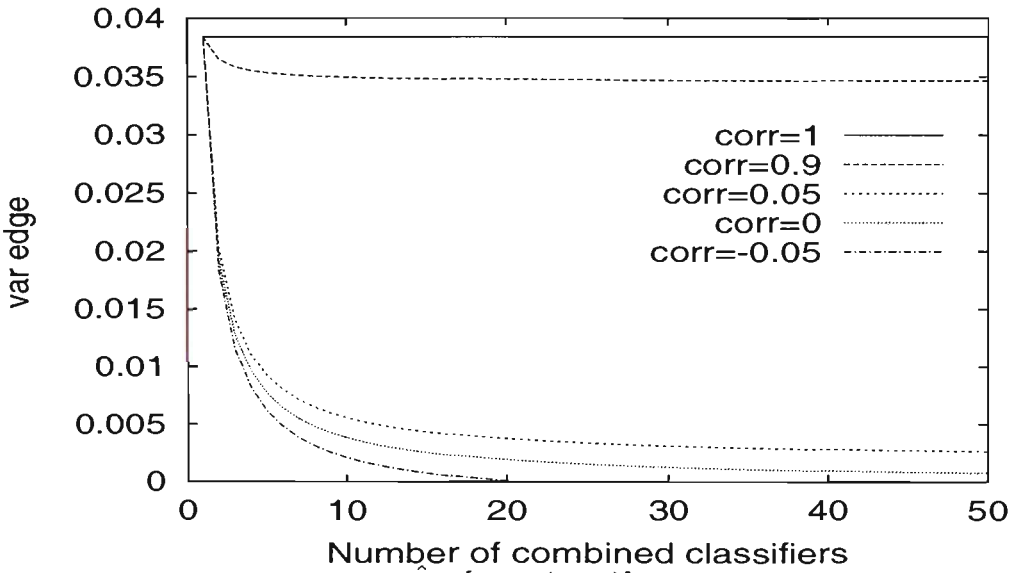

Figure 3.21: Plot of $\hat{Var}_i[edge(m,i)]$ for varying ρ vs. number of combined classifiers ($e = 0.04$).

variance becomes negative at $m = 20$ when $\rho = -0.05$.

To check the degree of $\hat{corr}_i[I_j(i), I_k(i)]$ for the *letter* dataset, the variance values obtained empirically are overlaid onto the theoretical variance trend graph drawn as Figure 3.21. The value of e is again set at 0.04 which is a close match to the values of e_j obtained empirically for the letter data. Referring to Figure 3.22, it may be concluded that $\hat{corr}_i[I_j(i), I_k(i)]$ for the *letter* data is in the range $0 \leq \hat{corr}_i[I_j(i), I_k(i)] \leq 0.10$.

A limitation of this scenario is the assumption of the unweighted errors being equal for each iteration. This is clearly not the case in practice and in fact it has been observed empirically that the e_j's tend to increase as the number of boosting iterations increases. This assumption leads to an underestimation of variance as the $e(1-e)$ term is deflated. This is also evident in Figure 3.22 where the empirical data (marked by a '+') falls above the $\frac{e(1-e)}{m}(1+\rho(m-1))$ line for

all levels of ρ.

It should also be noted that the $\hat{\rho} = 0$ line on Figure 3.22 demonstrates the variance reduction for a pure binomial model. We see this clearly not the case for the example dataset plotted and conclude that a correlation term is certainly present and significant in any edge variance expressions. This is particularly so in early iterations as boosting works hard to correctly classify difficult observations, resulting in significant negative $\hat{corr}_i[I_j(i), I_k(i)]$ between iterations for such observations.

An extension to this class of voting system is to consider the where only neighbouring pairs of errors have correlation ρ. In this scenario we have:

$$\hat{Var}_i[edge(m,i)] = \frac{e(1-e)}{m}\left(1+\rho\frac{(m-1)}{m}\right)$$

In practise, the above expression for $\hat{Var}_i[edge(m,i)]$ will not differ significantly from the expression derived as 3.4.1 in this Section.

These empirical results for $\hat{Var}_i[edge(m,i)]$ confirm Breiman's notion of "equalization", whereby an ensemble classifier equalises the number of times a particular observation is classified incorrectly throughout the m trials. Looking at this notion, it may be interesting to track the percentage of incorrect classifications per observation as m increases. This percentage should become approximately constant as m increases. This is observed in Figures 3.23 and 3.24 for one cross validated fold for the *bands* and *letter* datasets. It can be seen for the *bands* dataset that the percentage incorrect for each observation has settled down to a

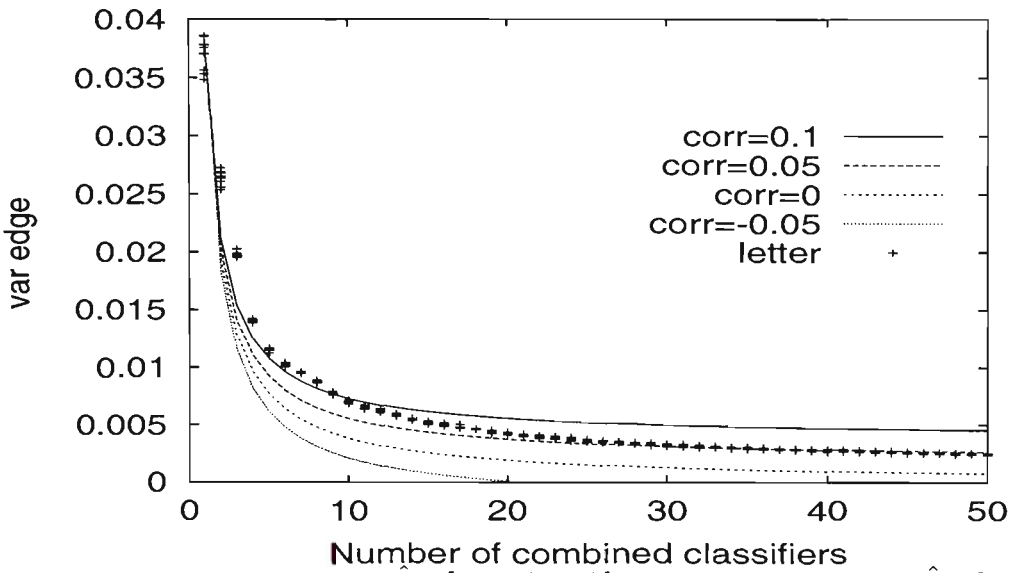

Figure 3.22: Plot of $\hat{Var}_i[edge(m,i)]$ for varying ρ and $\hat{Var}_i[edge(m,i)]$ for the *letter* data vs. number of combined classifiers ($e = 0.04$).

reasonably tight band about 20%. For the *letter* data, this band also falls around 20%, although it is difficult to distinguish visually because of the large number of observations being plotted on a single graph.

Now, if in fact boosting is working at equalising the percentage of misclassifications across all observations, there may be some observations for which this is not possible. Such observations may be considered to be noise or outliers. Removal of such observations could lead to improved classification accuracy if an ensemble is retrained without these observations. This notion is extended in Section 4.2, where the next set of empirical studies tracks the percentage incorrect at $m = 1, 5, 10, 50$. The resulting $edge(m,i)$'s are plotted to check if any observations still have a high percentage of incorrect classifications after $m = 50$ iterations. This would coincide with the observations having a higher average edge value as they would always contribute an incorrect vote. Figure 3.23 shows

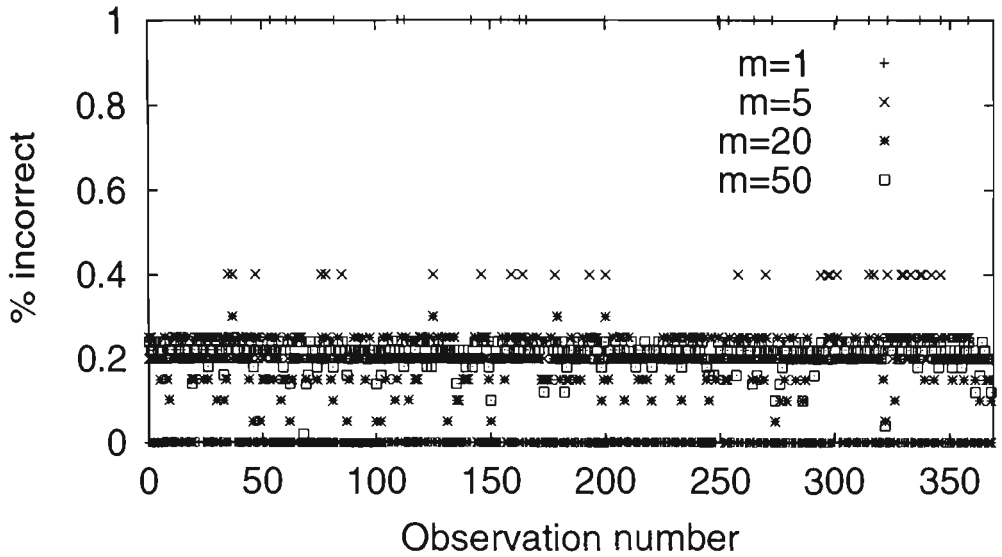

Figure 3.23: % incorrect per observation number - varying m : *bands* data.

this scenario for one fold of the *bands* dataset. It can be seen that a small subset of the observations have edge value equal to 0.4 after only 5 iterations and are easily distinguished from the main cluster of observations.

3.4.3 Scenario 3 : The m classifiers do not make identical predictions at each iteration but have the same individual error rates with $\hat{corr}_i[I_j(i), I_k(i)] = \rho$. Voting weight of the j-th classifier $= c_j$ $(\sum_{j=1}^{m} c_j = 1)$.

In this scenario, the m classifiers do not make identical predictions at each iteration but have the same individual error rates. In this case, $\hat{corr}_i[I_j(i), I_k(i)] = \rho$ $(\frac{-1}{m-1} \leq \rho < 1)$. Voting weight of the j-th classifier $= c_j$ $(\sum_{j=1}^{m} c_j = 1)$.

So,

- $e_j = e_k = e$
- $\hat{Var}_i[I_j(i)] = \hat{Var}_i[I_k(i)] = e(1-e)$

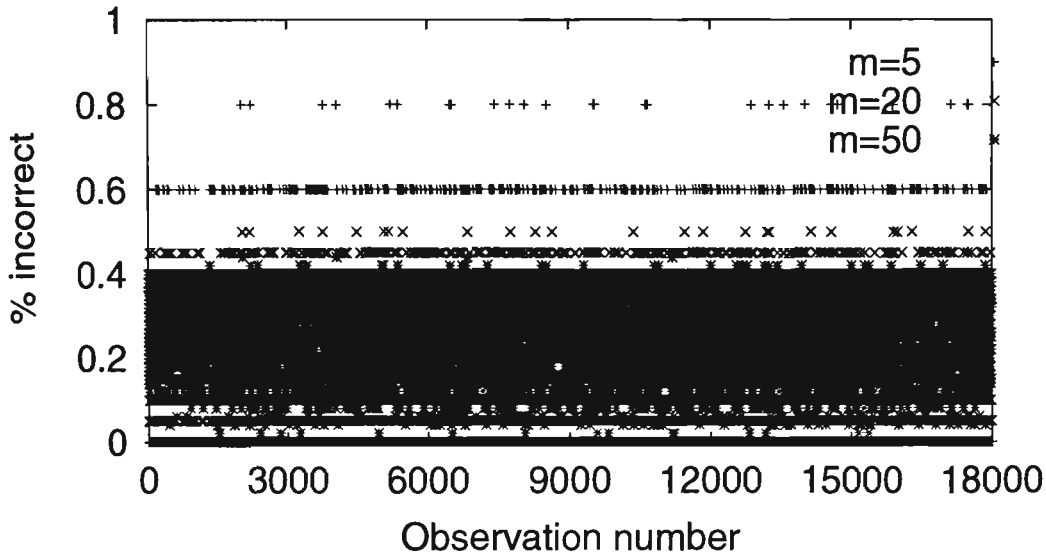

Figure 3.24: % incorrect per observation number - varying m : *letter* data.

- $\hat{corr}_i[e_j, e_k] = \rho$
- $\hat{Cov}_i[I_j(i), I_k(i)] = \rho e(1-e)$

Again, the edge values would vary between 0 and 1 and it is not expected that the variance of these edge values to be constant as m increases. The value of $\hat{Var}_i[edge(m,i)]$ in this case is derived below.

$$\begin{aligned}
\hat{Var}_i[edge(m,i)] &= \sum_{j=1}^{m}(c_j)^2 e_j(1-e_j) + 2\sum\sum_{j<k}^{m} c_j c_k \hat{Cov}_i[I_j(i), I_k(i)] \\
&= \sum_{j=1}^{m}(c_j)^2 e(1-e) + 2\sum\sum_{j<k}^{m} c_j c_k \rho e(1-e) \\
&= e(1-e)\left(\sum_{j=1}^{m}(c_j)^2 + 2\rho\sum\sum_{j<k}^{m} c_j c_k\right)
\end{aligned}$$

Now, $(\sum_{j=1}^{m} c_j)^2 = \sum_{j=1}^{m}(c_j)^2 + 2\sum\sum_{j<k}^{m} c_j c_k$. Therefore,

$$\sum\sum_{j<k}^{m} c_j c_k = \frac{1 - \sum_{j=1}^{m}(c_j)^2}{2}$$

And,

$$\begin{aligned} \hat{Var}_i[edge(m,i)] &= e(1-e)[\textstyle\sum_{j=1}^{m}(c_j)^2 + \rho(1 - \sum_{j=1}^{m}(c_j)^2)] \\ &= e(1-e)[\textstyle\sum_{j=1}^{m}(c_j)^2(1-\rho) + \rho] \end{aligned} \tag{3.4.2}$$

It may be interesting and beneficial to test the value of $\hat{Var}_i[edge(m,i)]$ on varying distributions of c_m. Empirically, it is observed that the distribution of c_m does not have a consistent form across datasets. (Refer to Figures 3.11 - 3.13)

A simplified distribution for c_j is outlined below.

Let $c_1 \ldots c_r$ have equal weights totalling w.

$$\begin{aligned} c_j|_{j=1\ldots r} &= \frac{w}{r} \\ c_j|_{j=r+1\ldots m} &= \frac{1-w}{m-r} \end{aligned}$$

Now,

$$\begin{aligned} (\sum_{j=1}^{m} c_j)^2 &= (\sum_{j=1}^{r} c_j)^2 + (\sum_{j=r+1}^{m} c_j)^2 \\ &= r \times (\frac{w}{r})^2 + (m-r) \times (\frac{1-w}{m-r})^2 \\ &= \frac{w^2}{r} + \frac{(1-w)^2}{(m-r)} \end{aligned}$$

And so,

$$\begin{aligned} \hat{Var}_i[edge(m,i)] &= e(1-e)[\rho + (1-\rho)(\frac{w^2}{r} + \frac{(1-w)^2}{(m-r)})] \\ &= e(1-e) \times F(m,r,w,\rho) \end{aligned}$$

It can be seen that in each $\hat{Var}_i[edge(m,i)]$ expression derived to this point, the quantity $e(1-e)$ is multiplied by a factor, F. When comparing voting schemes for

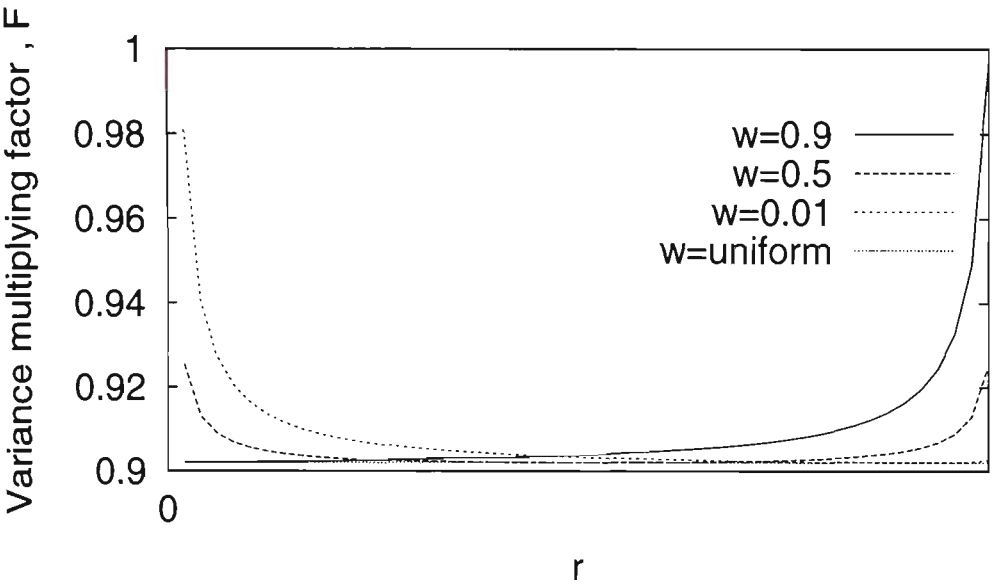

Figure 3.25: Variance multiplying factor for $m = 50$ and $\rho = 0.9$.

maximum variance reduction, further analysis of this factor will be paramount. In the first case (Section 3.4.1), this factor was equal to 1, in the second case (Section 3.4.2) the factor was equal to $\frac{1}{m}[1 + \rho(m-1)]$ and now the factor is a more complicated function of m, r, w, ρ. Since F is a multiplicative factor and if boosting results in variance reduction, it would be of benefit to test conditions for $min(F)$ since a minimum F will result in greatest variance reduction, conjectured to give boosting its greatest efficiency.

The function $F(m, r, w, \rho)$ is plotted for $m = 50$, $\rho = 0.1, 0.5, 0.9$, $w = 0.01, 0.5, 0.9$ and varying r. On the same plot, the value of $\frac{1+\rho(m-1)}{m}$ is overlaid, which is the equivalent multiplicative function for equal voting weights of $c_j = \frac{1}{m}$ (denoted as *w=uniform* in Figures 3.26 and 3.25).

Referring to Figures 3.25 and 3.26 it can be seen that:

- The lowest multiplicative factor and hence lowest $\hat{Var}_i[edge(m, i)]$ occurs

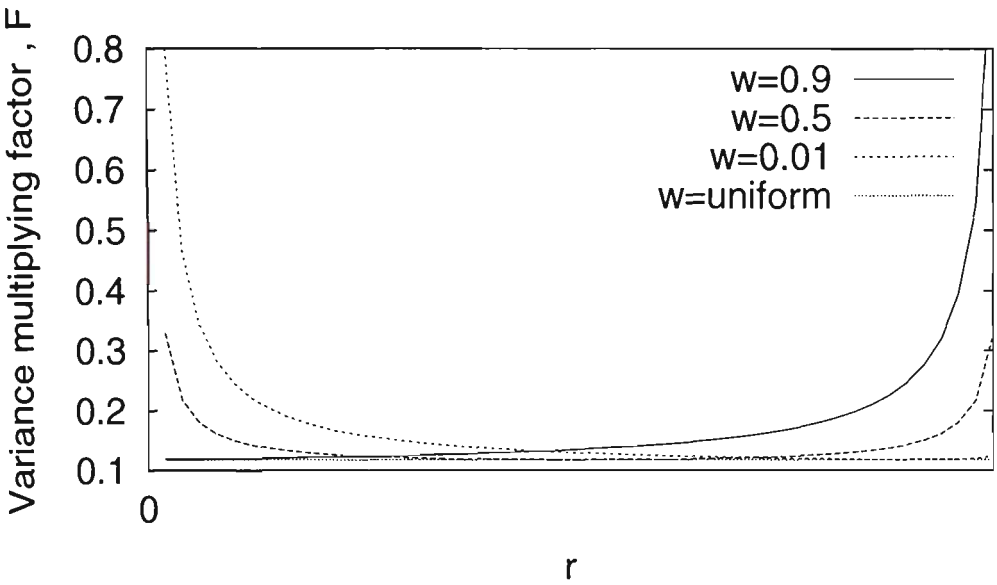

Figure 3.26: Variance multiplying factor for $m = 50$ and $\rho = 0.1$.

when $c_j = \frac{1}{m}$ (denoted as *w=uniform* in Figures 3.26 and 3.25).

- Low multiplicative factors occur for high $r \Leftrightarrow$ low w , r $\approx \frac{m}{2}$ and low $r \Leftrightarrow$ high w combinations

- When the values of c_j are very unbalanced lower variance occurs.

This is the same principle which applies in many fields (e.g. finance, logistics) where diversity increases variation and variation is minimised when uniform inputs are used.

3.4.4 Scenario 4 : The m classifiers do not make identical predictions at each iteration and have differing individual error rates with $\hat{corr}_i[I_j(i), I_k(i)] = \rho$. Voting weight of the j-th classifier $= c_j$ $(\sum_{j=1}^{m} c_j = 1)$.

The m classifiers do not make identical predictions at each iteration and have differing individual error rates. In this case, $corr_i[I_j(i), I_k(i)] = \rho(\frac{-1}{m-1} \leq \rho < 1)$.

Voting weight of the j-th classifier $= c_j$ $(\sum_{j=1}^{m} c_j = 1)$ and hence:

- $e_j \neq e_k$
- $Var_i[I_m(i)] = e_m(1 - e_m)$
- $corr_i[e_j, e_k] = \rho$
- $Cov_i[I_j(i), I_k(i)] = \rho\sqrt{e_j(1 - e_j)}\sqrt{e_k(1 - e_k)}$

$$\begin{aligned} Var_i[edge(m,i)] &= \sum_{j=1}^{m}(c_j)^2 e_j(1-e_j) + 2\sum\sum_{j<k}^{m} c_j c_k Cov_i[I_j(i), I_k(i)] \\ &= \sum_{j=1}^{m}(c_j)^2 e_j(1-e_j) + 2\rho\sum\sum_{j<k}^{m} c_j c_k \sqrt{e_j(1-e_j)}\sqrt{e_k(1-e_k)}\} \end{aligned}$$

This expression is unable to be simplified further, yet is the variance expression for the most general of voting systems. *AdaBoost* forms such a system yet it appears that the variance structure for *AdaBoost* closely matches Scenario 3 whereby the classifiers do not make equal predictions at each iteration but the pair-wise correlation between errors is equal across iterations. The theoretical and empirical results presented in this Chapter point to the notion of boosting equalising the proportion of misclassifications for each observation as the number of boosting trials increases.

This Section has confirmed that the most likely form of the $\hat{Var}_i[edge(m,i)]$ is an $\frac{e(1-e)}{m}$ expression multiplied by a factor which is a function of ρ and m. Variance

decay is quite dramatic in early iterations and tapers off in later iterations as more restrictions are placed on the strength of the covariance term.

Chapter 4

Noise Detection

4.1 Edge and Margin

In Section 2.4, we saw that recent explanations on the success of boosting and ensemble methods have their foundations in edge and margin analysis. The definitions of these two measures are repeated below for convenience.

Assume we have a base learner which produces hypothesis $h_j(\mathbf{x})$ at the j-th iteration, and an error indicator function, $I_j(\mathbf{x}_i) = I(h_j(\mathbf{x}_i) \neq y_i)$. Let c_j represent the vote for the j-th hypothesis with $\sum_{j=1}^{m} c_j = 1$. Then, after m iterations:

- $edge(m, i)$ = total weight assigned to all incorrect classes. Recall the definition from Section 3.1 and Equation (3.1.1).

- $margin_i(m, \mathbf{c})$ = total weight assigned to the correct class minus the maximal weight assigned to any incorrect class.

Schapire et al. [41] claim that boosting is successful because it creates a higher margin distribution and hence increases the confidence of correct classification. Schapire et. al. claim that

> boosting is particularly good at finding classifiers with large margins in that it concentrates on those examples whose margins are small (or negative) and forces the base learning algorithm to generate good classification for those examples.

i.e. boosting gives a higher 'distribution' of margins with boosting giving an overall higher margin distribution than bagging. Breiman, however, claims the high margin explanation is incomplete and introduces new ensemble techniques which actively improve margin distributions but do not result in improved generalisation error [7].

Section 3.1 demonstrated the tendency for boosting to 'balance' the edge (or margin) in its quest to classify more difficult observations. Using this property, a method of detecting noise and difficult boundaries in training data will be presented in later sections of this Chapter.

4.2 Extending Variance Trends to Detect Noise

The notion of 'balancing' has been discussed previously in Section 2.4. This property may be exploited to detect noise or 'difficult' sections in the training data. Since the distribution of edge values is expected to become more uniform as the number of boosting trials increases, deviations from this distribution may be assumed to be caused by noisy or incorrect data, or data falling on or near a classification boundary. Noisy data is certainly difficult to classify and some observations may be too difficult to classify correctly in all but a few iterations.

Such observations would have high initial edge values which remain high due to persistent misclassification. If these deviations from the overall edge distribution can be detected at say, $m = 10 - 20$ iterations, then the associated observations may be deemed to be noisy data. Identification and if necessary, removal of such observations should lead to improved classification accuracy on training and test data. The optimal choice of m is still unclear but at $m = 10 - 20$, computing time is still relatively small and deviations from the distribution should already be apparent. Empirical studies have also noted that little gain in boosting is made after 10 iterations [4, 12, 34].

To test this hypothesis, and check whether 'offending' noisy data could be identified, noise was injected into the *letter* and *census* datasets by assigning random class labels to 5% of the data. These datasets were chosen because of their size (20,000 and 32,000 observations respectively). To perform this randomisation, the data was shuffled, then the first 5% of observations were assigned a random class label before the data was reshuffled again. This ensured no systematic bias in the noise while still retaining the observation number for later comparison. The $edge(m, i)$ values were captured after 15 iterations (i.e. $edge(15, i)$) and plotted against observation number as displayed in Figures 4.1 and 4.2.

Referring to Figure 4.1 for the *letter* dataset, a clear distinction between edge values can be seen around observation 1000. For the *letter* data, observations 0-1000 were deliberately relabelled with random class values to simulate noise. The remainder of the observations were untouched, yet it is still possible that

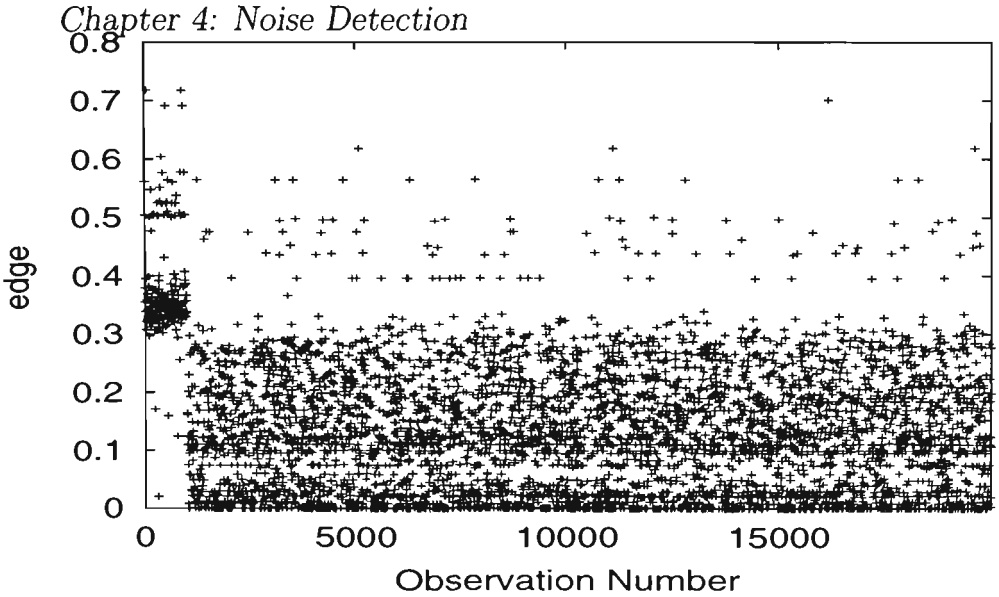

Figure 4.1: $edge(15, i)$ vs. observation number: *letter* data with class labels randomly assigned for observations 1-1000.

some of these observations may be noisy but as yet undetected in the raw UCI [2] dataset. For the *census* data plotted in Figure 4.2, another clear boundary is evident, this time with earlier observations (i.e. observations 0-1500) showing failure to reach lower edge values - (note the gap in the plot on the lower left hand corner). Although this example is contrived with known noise being introduced, it demonstrates an important result in being able to identify and eliminate noise. These results indicate that truncating the edge distribution or setting a threshold on edge values and re-learning could be an effective way of reducing noise and improving model performance.

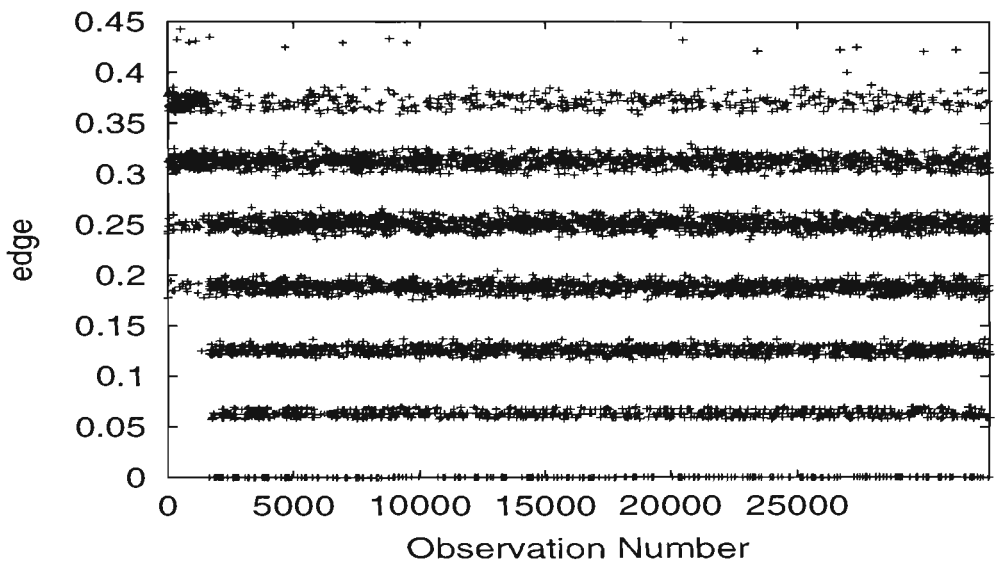

Figure 4.2: $edge(15, \mathbf{x}_i)$ vs. observation number : *census* data with class labels randomly assigned for observations 1-1600.

4.3 Results on the *BHP* Dataset

As is widely known, noise is inherent in many real-life databases and the industrial domain is no exception. A dataset provided by *BHP* Steel has been collected over a 6 month period on a continuous steel coating process [1]. The dataset comprises 5447 observations, 27 input (predictor) variables and a *0/1* response representing a steel coating defect. A response value of *0* indicates defect free steel and a response value of *1* flags the presence of a defect. The 27 input variables include zone temperature, line speed, cleaner mode, pressure, supplier, steel dimensions, steel strength etc.

The *BHP* data was run through 10 boosting iterations using *c*4.5 with default options as the base learner. The $edge(m, i)$ values were saved after each iteration for all observations, and sample statistics calculated after 10 iterations. Plotting $edge(10, i)$ versus observation number in Figure 4.3 shows a distinct band of lower edges for observation numbers in the vicinity of $2500 - 3000$.

Closer inspection reveals this to be observations $2480 - 3021$. Because the noise in this dataset is not contrived as for the *letter* and *census* datasets, analysis can proceed by trying to determine reasons for the area of low noise, possibly improving classification accuracy and ultimately reducing defect losses through more robust defect prediction. To proceed with analysis on this dataset, a new variable $obsflag$ was appended to the existing *BHP* dataset, creating a new dataset named *BHPflag.* The $obsflag$ variable was assigned a value of 1 if the

[1] **BHP Steel** : Metal Coating Production Line, Port Kembla, NSW Australia

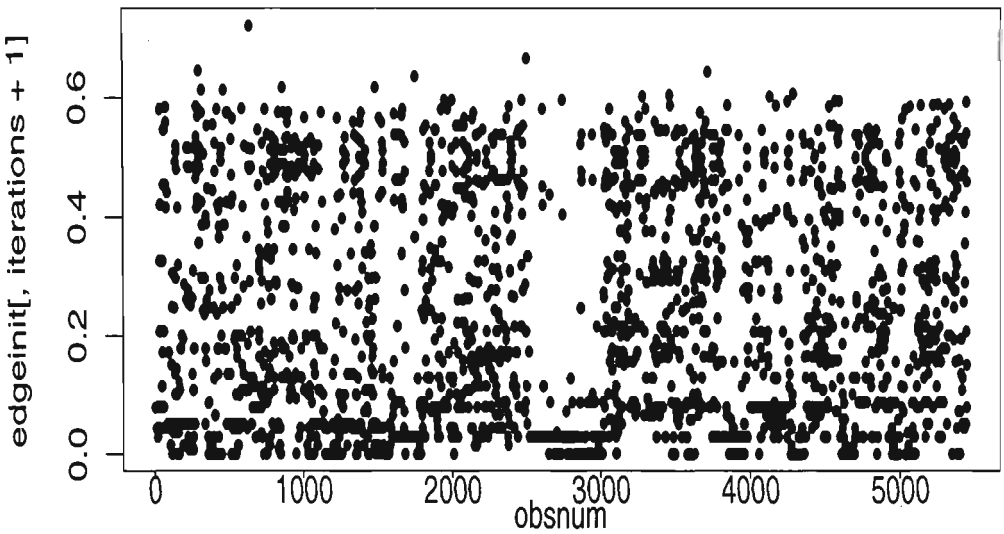

Figure 4.3: Plot of $edge(10, i)$ vs. observation number : *BHP* data.

observation number fell within the range $2480 - 3021$ inclusive and 0 otherwise. At this stage it is suspected that the central band of observations belong to a context change where the process is operating under different conditions to those of the remainder of the data. In order to determine what, if anything is unique about the predictors (operating conditions) in this centre band of observations, a single $c4.5$ decision tree was fitted. All previous 27 input variables were used as predictors, with the response being the new $obsflag$ indicator variable for observation number defined previously. The resulting tree shown in Figure 4.4 had a simple structure with only 8 leaf nodes. To check consistency, an *Splus* tree was also built, making use of the recursive partitioning *rpart* function [43]. The *Splus* tree resulted in very similar split and branch variables. In simple terms, there are two distinct operating regimes as determined in Figure 4.4 via the decision path given input attributes.

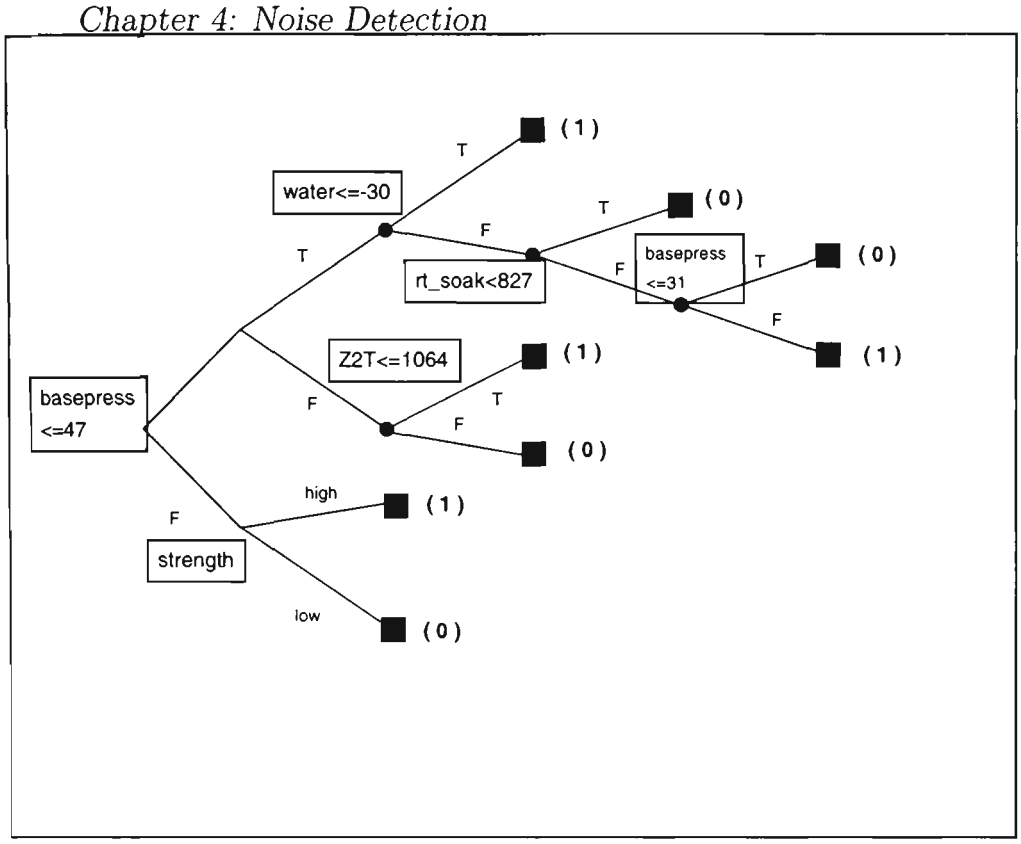

Figure 4.4: Decision tree for $obsflag$: BHP data.

1. observations 2480-3021

2. observations 1-2479 and observations 3022-end

These trees give concise decision rules for several different operating regimes using combinations of the input predictors. The advantage of such trees is that future observations can be classified into various operating regimes using the known values of their input attributes. Refer to Section 4.5 for a comprehensive list of all cross-validation results for the BHP dataset.

Sections 4.4 and 4.5 detail how each regime has a different mechanism for defect occurrence. If the BHP data was shuffled, these contexts would probably go undetected. Yet as a continuous industrial process, preserving the order of observations is vital in retaining key information regarding the process. Time variables in such processes are important markers in the progression of the process.

If a decision tree is built to classifiy defect presence including observation number (*obsnum*) as a predictor, context change may be detected if any split points on *obsnum* are returned. This would detect a context change in the classification variable (*defects=0,1*) but may not detect context changes in the input variables. When this notion is tested on the *BHP* data, the resulting decision tree is highly complex but does have an *obsnum* split point at $obsnum = 2338$. This split point occurs at depth 3 on the decision tree and is perhaps not as apparent as the context change detected in the $edge(10, i)$ versus observation number plot (Figure 4.3), yet still alludes to different structure beyond observation 2338.

When checking for incorrect predictions, most misclassification came from several clusters of consecutive observations. Two things could be occurring in relation to these misclassified clusters.

1. Observations falling within the range $2480 - 3021$ may have 'slipped' out of the common operating setup, resulting in brief sections where the process is not operating consistently. This may result in predictions implying that the process is operating outside the unique conditions in this flagged section.

2. Smaller clusters of observations outside observation range $2480 - 3021$ may have operating conditions which fall within those modelled by the decision tree in Figure 4.4 for the easily detected section of $2480 - 3021$. These smaller clusters are not visually detected by the $edge(10, i)$ versus observation number plot but may still be important when separating the dataset into 2 smaller datasets, each with distinct contexts.

Hence for future unseen observations, the first stage of classification would be to initially classify into $obsflag$ classes (2 sub contexts) according to the path followed in Figure 4.4. This now points to a possible 2 stage classification process. The next stage in the classification process for defects on the *BHP* data is described in Section 4.3.1.

4.3.1 Splitting the *BHP* Dataset

The *BHP* data can now be split into two distinct subsets, according to the decision tree drawn previously in Figure 4.4. Each of these subsets may be used as training data in their own right requiring us to embark on 2 new classification processes. The splitting should reduce any confounding caused by having data from two distinct contexts merged into a single dataset. It was decided not to remove any of the misclassified observations from the *obsflag* classification process as the process operating variables are within a reasonable range and cannot be considered outliers, but perhaps may be considered examples of sub-optimal process control.

In stage 2 of the analysis on *BHP* data, the data was split into two smaller datasets according to the following rules: *BHPsmall* = observations predicted as $obsflag = 1$ and *BHPlarge* = observations predicted as $obsflag = 0$ (according to predictions made by Figure 4.4). It was noted that the defect rate (original response variable) is significantly different for each sub-dataset. The *BHPsmall* dataset has a steel coating defect rate of 2.96% as opposed to the defect rate of 16.69% for the *BHPlarge* dataset. This already shows a clear distinction between

the 2 groups. If we can use information from the inputs (predictors) in each of the split datasets to determine further conditions for defects, the classification process will be considerably more robust than the initial single stage process using the entire *BHP* dataset.

4.4 Results on the *BHPsmall* Dataset

As outlined in Section 4.3.1, the dataset *BHPsmall* contains all observations which were predicted to have $obsflag = 1$ based on the decision tree drawn as Figure 4.4. Now that this subset of the *BHP* dataset is free of context differences, the process reverts to the original classification problem. Recall the aim of the classifier was to predict the presence of a steel coating defect (*0/1*) using a combination of 27 available process variables.

Two decision tree algorithms were applied (*Splus* and *c*4.5) to try to predict *0/1* defects as a response to 27 input (predictor) variables. *c*4.5 produced the simple decision tree shown in Figure 4.5.

Therefore, according to the decision tree output by *c*4.5, the key input predictors are simply cleaner mode and supplier. The error rate of this decision tree on the training data is 1.85%. Checking test set error via 10-fold cross-validation gives an estimated generalization error of 2.98%. A tree can also be built using the *rpart* function within *Splus*. Again the resulting tree has a simple structure with high accuracy on the training set. However, the split variables and points for this alternative tree building method are different from *c*4.5's. This *Splus* tree has a

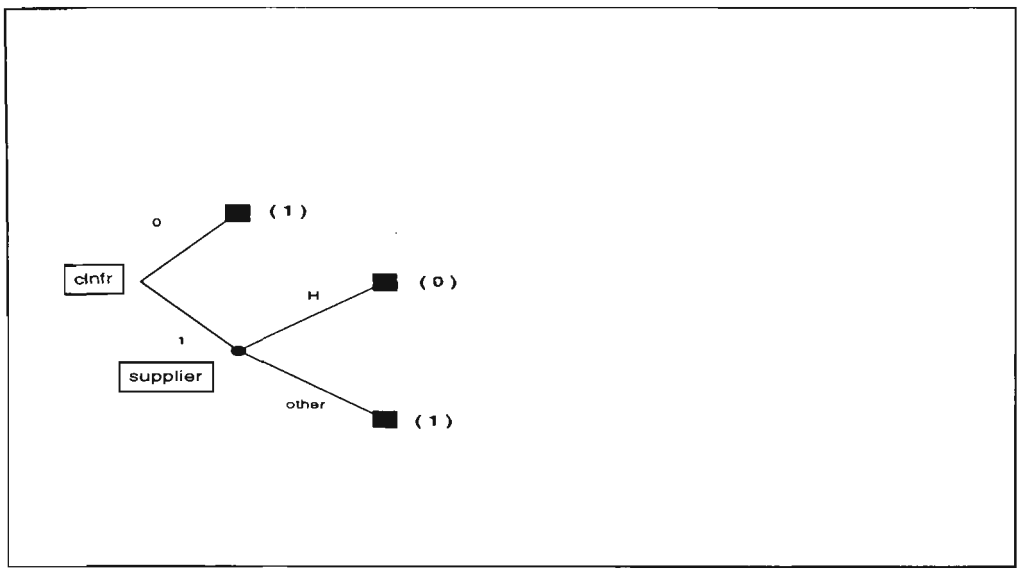

Figure 4.5: Decision tree for defect=*(0/1)* : *BHPsmall* data.

single split with error rate of 2.03% on the training data.

Checking those observations where an error was made shows that most errors resulted from failing to detect a defect. Also, the errors made by *c*4.5 are occurring are an overlap of 80% on the observations which were misclassified by the *Splus* *rpart* function. Therefore observations which are likely to be misclassified seem independent of the method of classification. These observations are the same as those identified in Figure 5.11 as potential outliers.

4.5 Results on the *BHPlarge* Dataset

The *BHPlarge* dataset contains all observations from the original *BHP* dataset which were classified as $obsflag = 0$ according to the tree in Figure 4.4. Recall that the resulting *BHPlarge* dataset has a significantly higher defect rate of 16.69%. A single tree using *Splus*' *rpart* function results in another simple

tree but it has a high error rate on the training data. (16.3%). Checking the results from a single decision tree built using default settings for *c*4.5 gives a very complex tree with 11.4% training set error. 10-fold cross-validation results in estimated test set error of 19.43%, which reduces to 17.07% when 10 iteration boosting is applied.

Table 4.1: Cross-validation results for *BHP* dataset.

dataset	**response**	**method**	**mean**
BHP	defect (0/1)	*c*4.5 single	15.44(4.64)
BHP	defect (0,1)	*c*4.5 10 boost	15.50(4.70)
BHPflag	obsflag (0/1)	*c*4.5 single	2.05(2.60)
BHPflag	obsflag (0/1)	*c*4.5 10 boost	2.62(5.33)
BHPsmall	defect (0,1)	*c*4.5 single	2.98(4.55)
BHPsmall	defect (0,1)	*c*4.5 10 boost	3.16(4.54)
BHPlarge	defect (0,1)	*c*4.5 single	19.43(3.58)
BHPlarge	defect (0,1)	*c*4.5 10 boost	17.07(5.45)

4.6 Boundary Detection

This Chapter has presented some interesting results from an empirical study on 2 larger UCI datasets and an industrial dataset, demonstrating that it is possible to track deviations in the edge (or margin) distribution when trying to identify the existence or otherwise of noisy data. Although this concept was only tested on known datasets from the UCI repository [2], as well as the BHP industrial dataset, initial results are certainly encouraging and further application and research on more UCI repository datasets should lead to a strong framework for detecting noise using boosting. Subsequent models will then be more robust for unseen observations.

It is possible that the notion of disturbed edge distributions can be extended to detect boundaries and express them in an easily interpretable and user friendly manner. In 2 and 3 dimensions, boundaries between classes can be easily detected by eye. However, as dimensionality increases, visually interpretable boundaries are rarely apparent. Obtaining easily explainable classification boundaries is a key goal of many analysts. For example in the medical field, it could be the distinction between mis-diagnosis of a disease; in industry, operating in boundary regions may cause a higher defect rate than is profitable. In the financial sector, borderline decisions are vital when deciding whether or not to enter or exit a position.

Instead of applying edge diagnostics to detect single observations with high final or average edge, there is the possibility of detecting a cluster of observations, all with high final average edge. In this case, we may conclude that such observations fall along or near decision boundaries. By isolating these observations we may gain some insight into areas of the input space which are critical for prediction. If we are then able to determine unique, defining characteristics of the input variables for such observations, interpretable boundaries can be unearthed. Knowing these boundaries will enable us to treat future observations with similar input properties with caution when classifying unseen cases.

Refer to Figure 4.6 which shows an example of the possible change in $edge(m, i)$ for 4 contrived observations as the number of iterations increases. The plotted number in Figure 4.6 refers to the observation number.

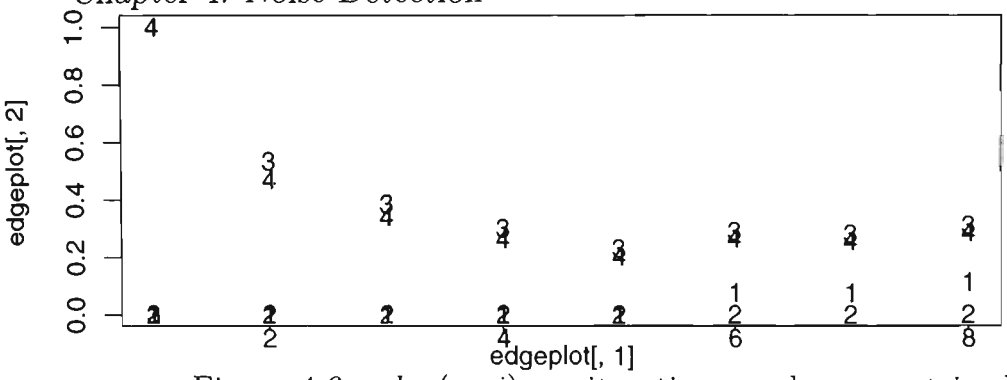

Figure 4.6: $edge(m, i)$ vs. iteration number : contrived dataset.

- Observation 1 is classified correctly in iterations 1-5 but then 'sacrificed' at iteration 6 and classified incorrectly, hence the increase in edge.

- Observation 2 is always classified correctly as it has a consistent edge value of 0.

- Observation 3 is classified correctly in the first iteration but incorrectly in some later iterations (this can be detected by the rise and fall in the edge plot).

- Observation 4 is initially classified incorrectly and the negative slope in the $edge(m, i)$ time series shows that boosting is able to classify observation 4 correctly in later iterations.

This data was a contrived series of 0/1 sequences to demonstrate the behaviour of the edge under varying model performance and was not a result of applying a classifier on an existing dataset.

The best edge measure for the detection of boundaries is unclear. Measures are sought which will capture and differentiate behaviour between observations such as those plotted in Figure 4.6.

It would also make sense to calculate a least squares regression for each observation with $edge(m, i)$ as the response and the iteration number, m as the predictor. This is a similar notion to fitting a time series model but the length of the series may restrict robustness.

Observations which are consistently classified correctly or incorrectly will have an estimate of slope close to 0 as the value of the edge should be relatively unchanged from 0. However, observations falling on border regions will either have their initial correct classification sacrificed or have their initial incorrect classification 'boosted' to a correct classification in subsequent iterations. Hence for such sets of observations, the edge will stray from its initial value. The resulting $\hat{\beta}_1$ for such observations will be either negative or strongly positive, with the latter also being reinforced by a high R^2 value. However, after testing all statistics empirically, it was seen that sufficient information can be gained from the $\hat{E}[edge(10, i)]$ versus $\hat{Var}[edge(10, i)]$ plots, with the information contained on regression diagnostic plots being redundant. This seems contradictory to earlier results where trends are present in both the mean and variance. However, after 10 iterations, the dramatic changes in these statistics have already taken place and these statistics are equally as stable as R^2 and $\hat{\beta}_1$.

It is suggested to calculate the following statistics after $m = 10$ iterations, with subsequent analysis confirming these statistics as feasible options. Note the expectations and averaging are not taken over all observations as in Section 3.1 and the i subscript is omitted in expectation and variance terms, resulting in

each observation having a mean and variance edge value calculated over $m = 10$ iterations.

- $edge(10, i)$ = final edge after 10 iterations
- $\hat{E}[edge(10, i)] = \frac{1}{10} \sum_{m=1}^{10} edge(m, i)$
- $\hat{Var}[edge(10, i)] = \frac{1}{10} \sum_{m=1}^{10} (edge(m, i) - \hat{E}[edge(10, i)])^2$

4.6.1 A Simulated Classification Problem

Two contrived datasets were created using the random number generators available in *Splus*. Both datasets contain 2 classes, each class being a sample of 100 observations generated from distinct bivariate normal distributions with $\boldsymbol{\mu}_1 = (1, 1)^T + \boldsymbol{\mu}_2$ and common $\boldsymbol{\sigma}$. $\boldsymbol{\mu}_1, \boldsymbol{\mu}_2$ and $\boldsymbol{\sigma}$ were generated randomly. Noise was added to the variates via the *rnorm* term in *Splus*. The final dataset is of the form $(X1_i, X2_i, class_i)$ $(i = 1 \ldots 200)$ with 100 observations being generated for each class. No deliberate outliers are added, although the noise results in a handful of outlying points. Random seeds of 2 and 24 were used in order to replicate results, with the datasets being names *mvn2* and *mvn24* respectively. Figures 4.7 and 4.8 show scatter plots the raw data for each seed. There is a clear overlap between the two classes in each case. This will result in some uncertainty when determining decision boundaries between the classes.

For each dataset plotted in Figures 4.7 and 4.8, the following analysis steps were performed to determine class boundaries and possible outliers.

- c4.5 was run using default parameters for $m = 10$ boosting trials. For the

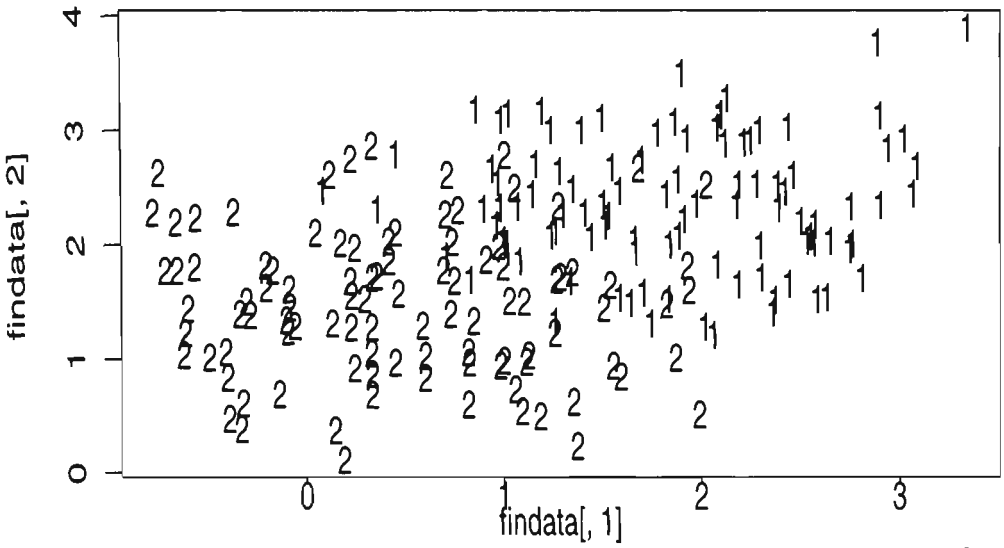

Figure 4.7: $X2$ vs $X1$ with symbols representing class : random dataset *seed=24*.

seed=24 dataset, only 8 iterations were performed before boosting terminated. It is widely known that boosting will terminate when too much noise is present in the training data.

- $edge(10, i)$, $\hat{E}[edge(10, i)]$ and $\hat{Var}[edge(10, i)]$ were calculated at the completion of 10 iterations (as outlined in Section 4.6).

- Scatter plots of $edge(10, i)$ versus observation number and $\hat{Var}[edge(10, i)]$ versus $\hat{E}[edge(10, i)]$ were formed.

From the resulting scatter plots, it may be possible to flag unusual observations. Unusual observations may be noise, outliers or observations lying on or near boundaries. From Figure 4.9, it is observed that $edge(8, i)$ exhibits gaps for observations 100 onwards possibly indicating class 2 is easier to classify. The $\hat{Var}[edge(10, i)]$ versus $\hat{E}[edge(10, i)]$ plot shown as Figure 4.10 shows 2 relationships : one a negative relationship between $\hat{Var}[edge(10, i)]$ and $\hat{E}[edge(10, i)]$ and the other a positive one. The separation of these clusters may help in detecting boundaries or regions of classification uncertainty. The idea behind this

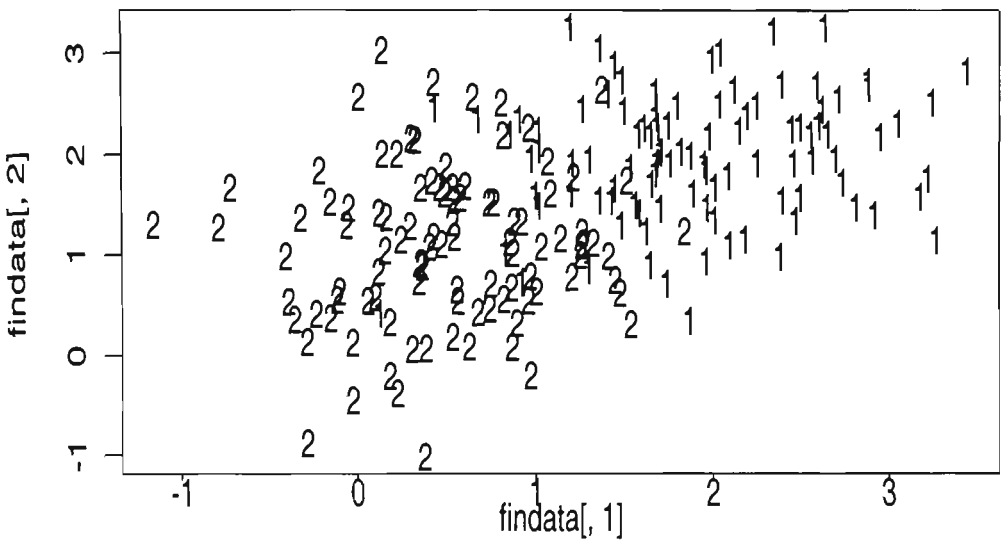

Figure 4.8: $X2$ vs $X1$ with symbols representing class : random dataset *seed=2*.

procedure is that an observation falling into the smaller 'negative' cluster has more variation in the edge and is probably a boundary point with classification decisions alternating at each iteration.

A new variable, $edgeflag$ is appended to the existing *mvn24* dataset, creating the *mvn24flag* dataset. Observations falling in the top right hand cluster in Figure 4.10 are assigned an $edgeflag$ value of 1 and the remainder of the observations assigned an $edgeflag$ value of 0. By then plotting the original $X1, X2$ variates with the symbol of the plotted point representing the value of the new $edgeflag$ variable, regions of 'difficult' observations are apparent. It is clear that non-linear boundaries exist if we are trying to determine conditions on $X1, X2$ which result in $edgeflag$ values of 1. This plot is shown as Figure 4.11. In order to model these boundaries according to the predictor variables, decision trees were again applied to each dataset with the predictors as $X1, X2$ and the response being the

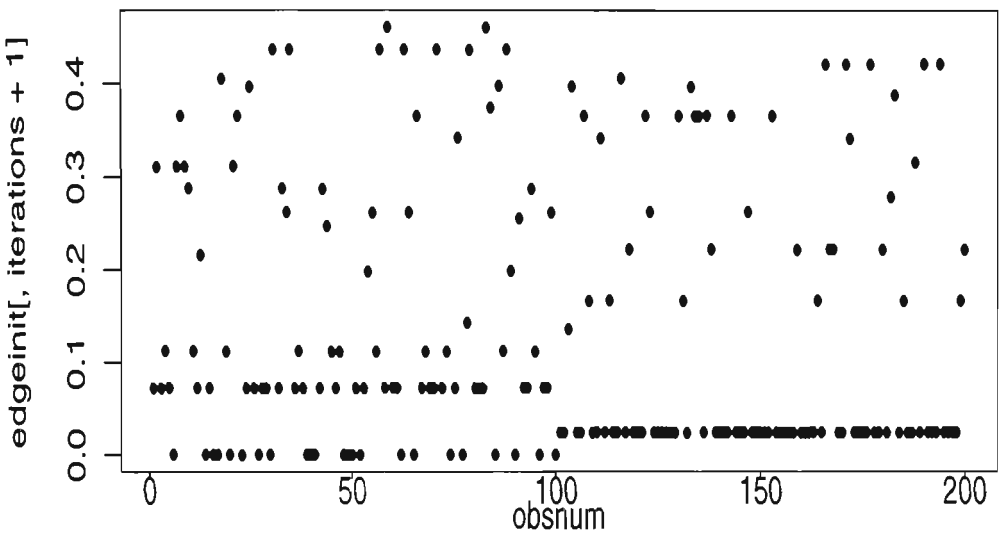

Figure 4.9: $edge(8, i)$ vs. observations number: *seed=24.*

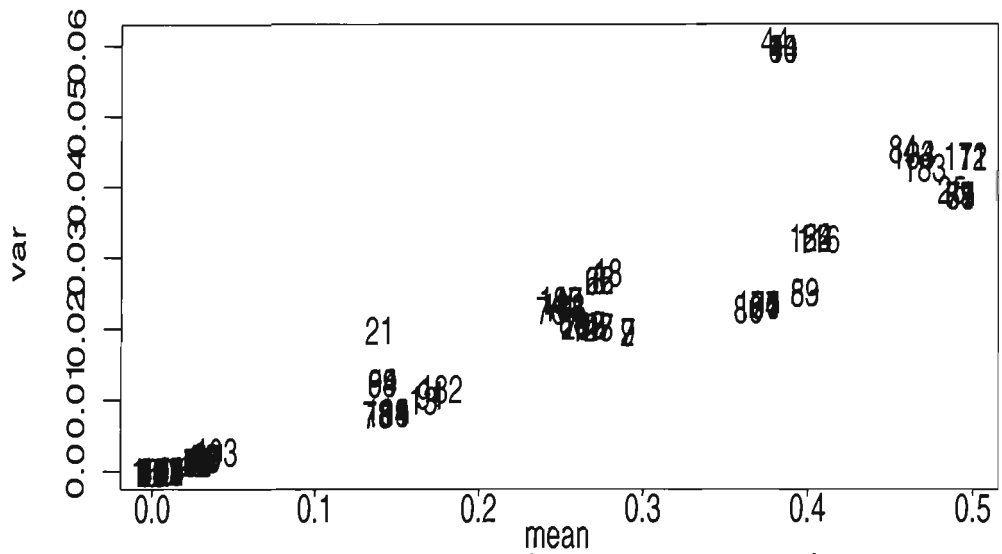

Figure 4.10: Plot of $\hat{Var}_i[edge(8, i)]$ vs. $\hat{E}_i[edge(8, i)]$: seed=24.

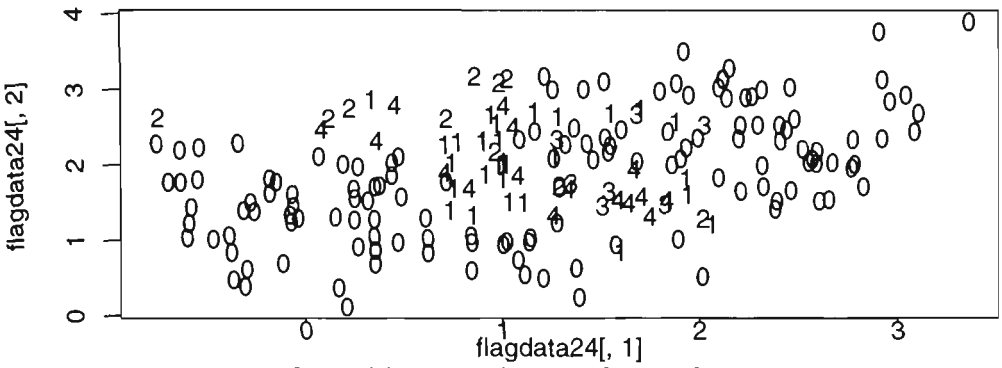

Figure 4.11: Plot of X1 vs. X2 with symbols representing number of edge flags triggered for each observation: seed=24.

0, 1 *edgeflag* value.

The preceding process is applied to the *mvn2* dataset, resulting in a plot of $\hat{Var}[edge(10,i)]$ versus $\hat{E}[edge(10,i)]$ shown as Figure 4.12. again a cluster of observations is evident in the top right hand corner of the plots, exhibiting a negative relationship between $\hat{Var}[edge(10,i)]$ and $\hat{E}[edge(10,i)]$. Assigning an *edgeflag* value of 1 to observations falling in this smaller cluster will lead to a new binary classification problem to determine structure pertaining to which observations are expected to fall in this smaller cluster in future, and would most likely be observations falling on or near borders or difficult decision boundaries. This solution to this classification problem is referenced in Section 4.6.2 at the conclusion of this Chapter.

4.6.2 Using Edgeflags on the *mvn24* Dataset

A new dataset was created by appending a new variable, *edgeflag* to the existing *mvn24* dataset. A value of 1 is assigned to this *edgeflag* variable if an observation falls in the upper right hand corner cluster of Figure 4.10 and a 0 otherwise.

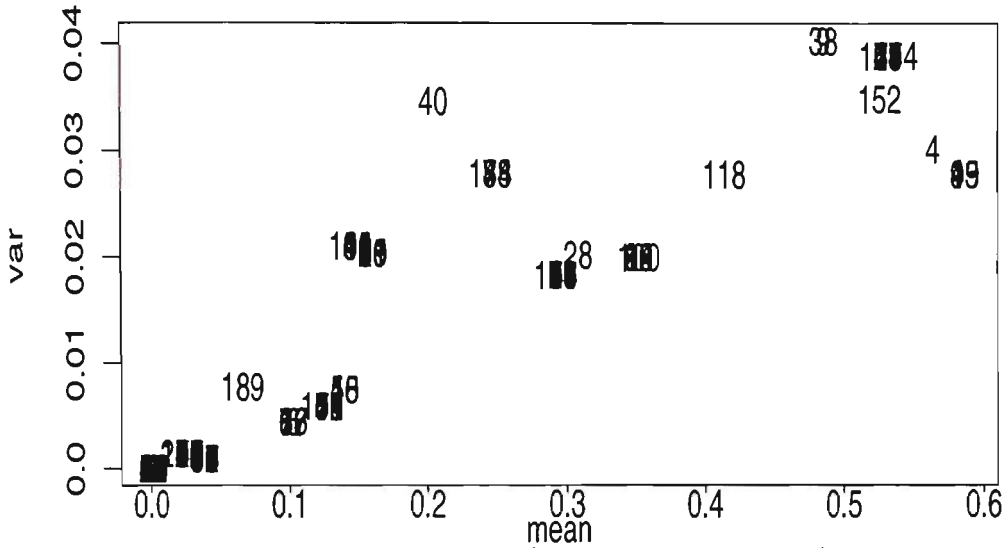

Figure 4.12: Plot of $\hat{Var}[edge(10,i)]$ vs. $\hat{E}[edge(10,i)]$: seed=2.

This creates a new classification problem with the new $edgeflag$ variable as the binary response. Only 2 input predictors are used since using response class *(1,2)* will be futile when the class value of unseen data is sought. Interestingly, the decision trees output by *Splus* and c4.5 have similar structure. The resulting trees are much simpler when compared to the trees produced when the initial 2 class problem was attempted using basic tree learning algorithms. Because of the non-linearity of boundaries, it is proposed to try to fit a neural network (Multi-Layer Perceptron) to the $edgeflag$ variable.

A neural network is trained using this $edgeflag = 0/1$ variable as its response and the initial $X1, X2$ as its predictors. It was decided not to use the initial response class *(1,2)* as a predictor because the model will be unable to generalise to unseen cases if it requires knowledge of the class. Refer to Figures 4.13 and 4.14 for contour plots of the surface generated by neural networks for seeds of

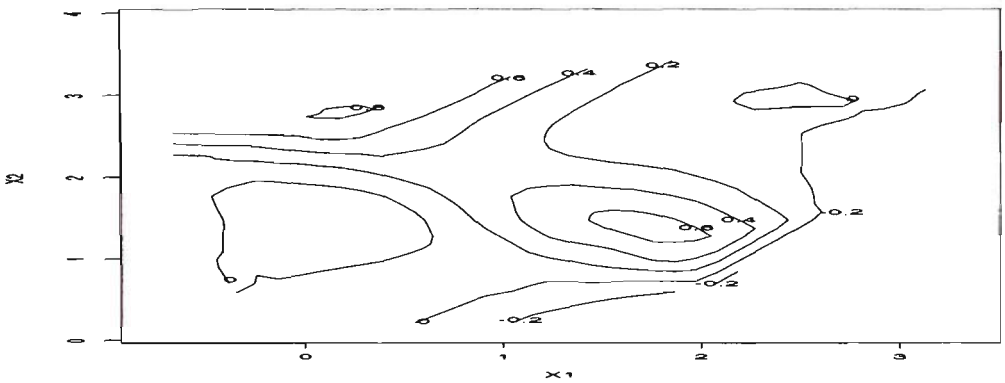

Figure 4.13: Contour plot of predicted edge flag (via neural network):*seed=24*.

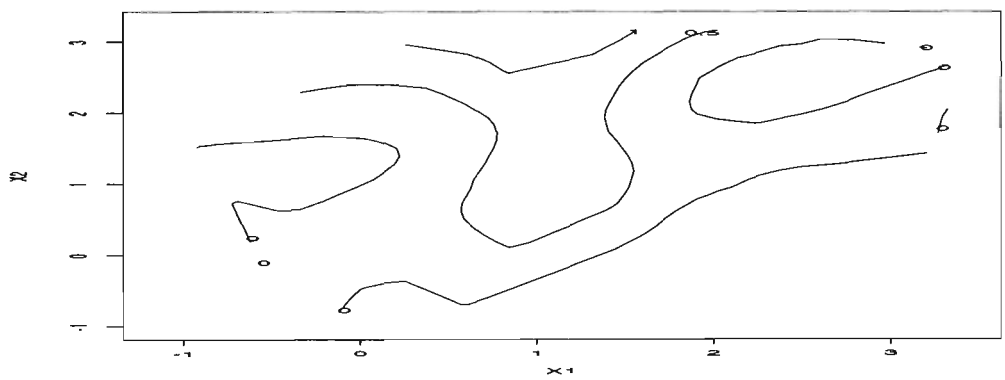

Figure 4.14: Contour plot of predicted edge flag (via neural network):*seed=2*.

24 and 2 respectively . The higher values are certainly occurring on the border regions between the two classes in each of the simulated datasets. This can also be represented by a distinctive ridge in a 3 dimensional surface plot.

An alternative method of boundary modelling is to use logistic regression with the $edgeflag = 0/1$ as the response. For the two simulated datasets (*seed=2,24*), a second order model was chosen which was of the form:

$$logit(edgeflag) = \beta_0 + \beta_1 X1 + \beta_2 X2 + \beta_3 X1^2 + \beta_4 X2^2 + \beta_5 X1X2$$

The second order terms were included to allow for some curvature of the boundary.

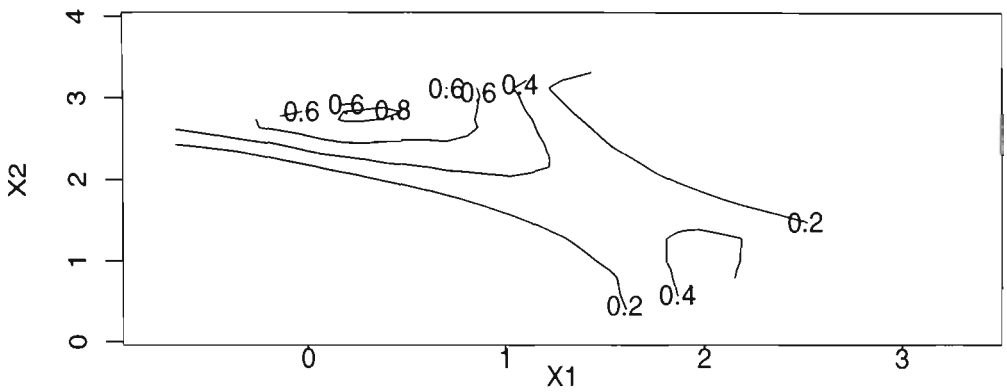

Figure 4.15: Contour plot of predicted edge flag using a quadratic logistic model : *seed=24*.

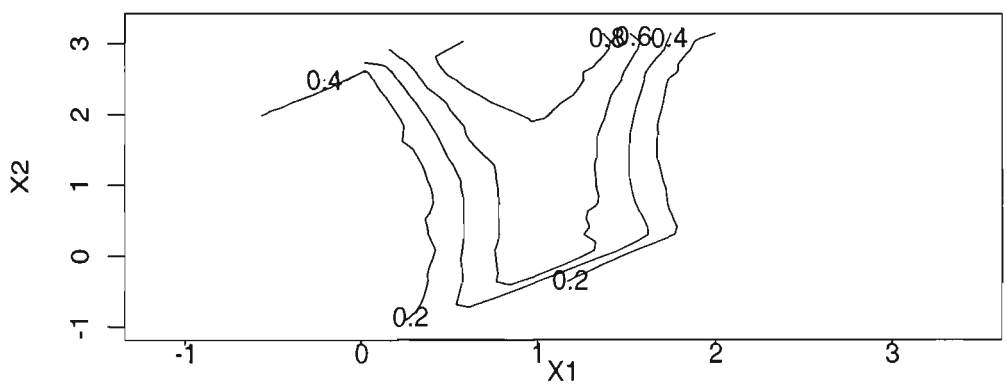

Figure 4.16: Contour plot of predicted edge flag using a quadratic logistic model : *seed=2*.

Refer to Figures 4.15 and 4.16 for the resulting contour plots of the predicted *edgeflag* variable. The contours follow the same direction as those detected using a neural network. It is interesting to note that both have high predictions in the top left hand corner with a central band of high predictions indicating the boundary region for the *seed=24* dataset, and consistency of methods. Chapter 5 will extend these results to datasets from the UCI [2] repository and confirm the success of the methodology suggested in this Section.

Chapter 5

Further Exploitation of Edge Distributions

5.1 Background

The ability to classify unseen observations efficiently is a desirable property of many data mining algorithms. Numerous barriers may be present when an optimal classification model is being sought. In particular, unavoidable dataset symptoms such as noisy data, outliers and 'fuzzy' boundaries inhibit a model's performance. In Chapter 4, a noise detection process was introduced, whereby observations unable to be 'equalised' with high final edge were flagged as unique noisy or outlying observations. Demonstration of this process on induced noise on the UCI *letter* and *census* datasets proved the process to be simple and robust. Testing on a real-life industrial dataset also demonstrated the technique's ability to detect noise in temporal data.

The noise detection process outlined in Chapter 4 is founded purely on flagging observations having high values of the final edge after m iterations. An obvious extension to the use of final edge as the indicator to the presence of noise is

the notion of measuring deviation from the overall edge distribution for each observation across iterations. Anomoly detection may be optimised by monitoring edge statistics as opposed to monitoring only the final edge. Since edge values are calculated for each observation at each iteration, they will form a distribution for each observation across iterations. Appropriate sample statistics for each edge distribution may be valuable in shedding light not only on unique observations, but on clusters of observations possibly forming border or 'fuzzy' regions, or groups of observations that are behaving differently to the remainder of the dataset. Chapter 4 demonstrated a simple example of this notion in detecting borders on a simulated bivariate binary classification problem.

Section 4.6.2 demonstrated this idea by calculating $\hat{E}[edge(10,i)]$, $\hat{Var}[edge(10,i)]$ and $edge(10,i)$ for two example binary classification problems. Assigning thresholds to these statistics and flagging observations whose associated edge statistics fall beyond the thresholds led to repeated highlighting of clusters of observations along the boundaries or difficult decision points. These observations alternate between correct and incorrect classification at each iteration and determining the location of such observations with respect to the input variables is invaluable for future classification as we can be wary of particular combinations of the input variables which are difficult to classify.

Instead of only detecting single, unique outliers or border regions, it would be of benefit to be able to detect clusters of observations with properties dissimilar to the rest of the data. By isolating these observations and attempting to determine

their collective structure, we may gain insight into areas of the input space which should be segregated, or at the least, treated with caution.

Another barrier to successful classification is the presence of more than one context within a dataset i.e. different sections of the dataset being classified according to significantly different models. An extreme example of this is when two sections of the data are the inverse of each other. For binary classification this means that groups of observations may have identical values of their input variables but some members of the group will have opposite class labels. Undetected, this results in increased generalisation error and a more complex model prone to over-fitting as the classifier is torn between classifying observations which oppose each other. The concealed presence of such sub-contexts has previously been ignored, and subsequently been attributed to unmodellable noise and deterioration in generalisation error. If such situations can be detected and either removed or partitioned in the final modelling stage, significant gains may be made in both model simplification, robustness and improved generalisation performance.

Two issues now arise:

1. Detecting clusters is a different problem to detecting boundaries as we are no longer assuming a global model is applicable for the entire dataset. Different sections of the data may behave differently and in extreme cases be the inverse of each other. Are the methods introduced in Sections 4.6 - 4.6.2 applicable to the problem of detecting clusters other than boundary points within a dataset?

2. The use of sample statistics suggested in Section 4.6.2 have been shown to yield constructive results on the datasets tested. No theoretical justification, however, has been provided to substantiate the use of such statistics.

In theory, decision tree methodology should be capable of accommodating Issue 1 (assuming the correct split is taken on significant input variables). Also assumed is the existence of the appropriate branching variable that will distinguish between the two models. In many datasets it is suspected that key variables required to distinguish between classes have not been measured and hence groups of observations distinguishable by such variables appear to behave differently to the majority of the data since the information available is incomplete.

This Chapter attempts to address both issues outlined above. Firstly, some empirical studies on UCI [2] datasets are described which result in clusters and inverse sub-models being detected. Theory for these studies is detailed in Section 5.7.1, which in turn tackles Issue 2 detailed above.

As discussed in Section 4.6, to facilitate noise and boundary detection, it is unclear as to which edge statistics would be the most appropriate. The same notion holds for detecting clusters of observations with unique properties. Measures are sought which will capture and differentiate behaviour deviant from the main 'core' of data. Observations belonging to the main 'core' of data that are consistently classified correctly will have a low mean and variance of the edge. Observations that are persistently misclassified will also have a low edge variance but high average edge. Groups of difficult observations that collectively behave in the same

manner but differ in structure from the majority of observations will have a high edge variance as boosting will alternate between correct and incorrect classification. Such clusters should fall in a common location when the variance of the edge is plotted against the mean edge.

It is therefore proposed to calculate only the mean and variance of the sequence of edge values for each observation. Depending on the succession of correct or incorrect classifications for each observation, these edge statistics will vary greatly between observations. As mentioned above, it is suspected that observations that can be grouped together will have similar mean and variance of their edge distributions. More formally, for $m = 10$ boosting trials, the edge measures would be calculated as follows:

- $\hat{E}[edge(10, i)] = \frac{1}{10} \sum_{m=1}^{10} edge(m, i)$
- $\hat{Var}[edge(10, i)] = \frac{1}{10} \sum_{m=1}^{10} (edge(m, i) - \hat{E}[edge(10, i)])^2$

The next step in the cluster identification process encompasses the plotting of $\hat{Var}[edge(10, i)]$ versus $\hat{E}[edge(10, i)]$. Any clusters or sub contexts within the dataset should become apparent when this diagnostic plot is drawn. It will be shown on the UCI [2] *colic* and *heart-h* datasets, as well as a large industrial domain dataset, that signature behaviour is observed when this plot is drawn. Figure 5.1 is a sketch of the typical regions observed when $\hat{Var}[edge(10, i)]$ is plotted against $\hat{E}[edge(10, i)]$. We observe a large group of observations exhibiting a positive relationship between edge variance and mean edge and a smaller group exhibiting a negative one. The positive cluster is significantly denser with many

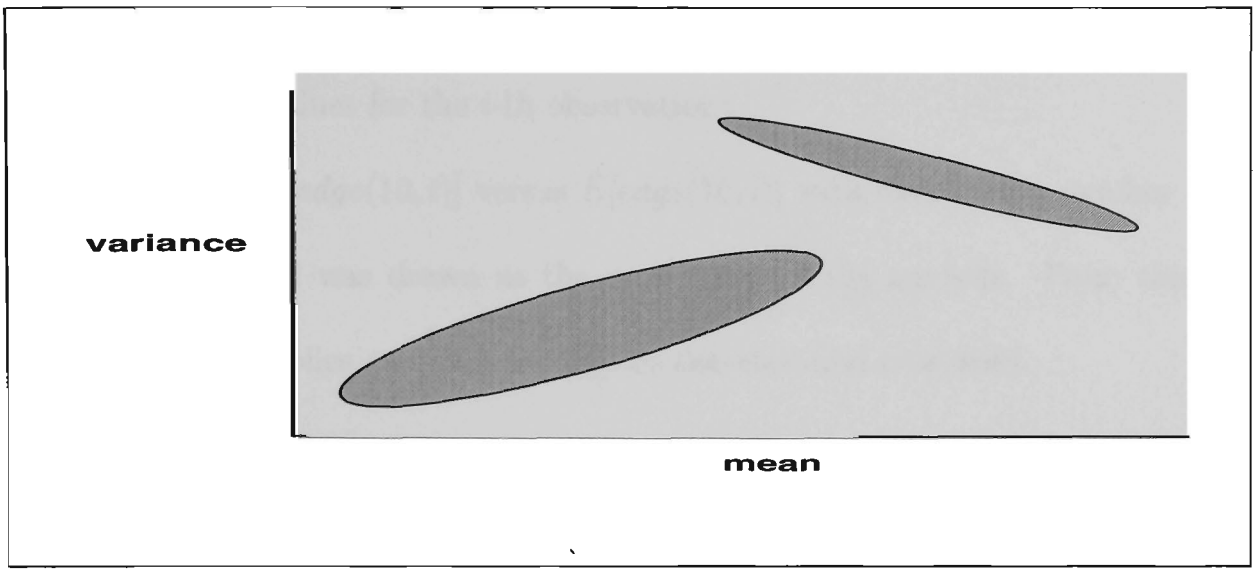

Figure 5.1: Typical plot of $\hat{Var}[edge(10,i)]$ vs. $\hat{E}[edge(10,i)]$

observations having a mean and variance of their edge both equal to zero. Could these observations falling into the smaller, negative cluster belong to a different context?

Sections 5.2.1-5.2.6 demonstrate this phenomenon in detail on both the UCI *colic* and *heart-h* datasets.

5.2 Basic Methodology used on all Datasets

The following Sections describe an empirical study undertaken on a selection of UCI [2] datasets. For all datasets, the term 'boosting' refers to the *AdaBoost* algorithm presented in Section 2.3.1. Unless otherwise described for a particular dataset, 10 boosting iterations were applied using $c4.5$ with default options as the base learner. Edge values for each observation ($edge(m,i)$, $m = 1 \ldots 10$) were stored at the completion of each iteration. At the completion of the 10

boosting iterations, the mean ($\hat{E}[edge(10,i)]$) and variance ($\hat{Var}[edge(10,i)]$) of the 10 stored edge values were calculated for each observation (the parameter i representing values for the i-th observation).

A plot of $\hat{Var}[edge(10,i)]$ versus $\hat{E}[edge(10,i)]$ with observation number as the plotted symbol was drawn as the next stage in the analysis. From this plot, outliers, anomolies and clusters may be detected and examined.

5.2.1 Results on the *colic* Dataset

The *colic* dataset contains 300 training observations with 23 input attributes and 30% missing values on the input attributes. The aim is to predict a binary classification variable *(yes/no)* pertaining to whether or not a horse's lesion was surgical. Input attributes are a combination of continuous and categorical variables and include : was surgery undertaken (yes/no), pulse, respiratory rate, age and a categorical pain variable.

Decreased generalisation performance has been observed on this UCI [2] dataset when *AdaBoost* is applied [34]. In [34], Quinlan estimates generalisation error using 10-fold cross-validation with $c4.5$ using default options as the base learner. This error was reported to be 14.92% on a single decision tree, and when boosted this generalisation error increased to 18.83%. As a check on the code used for this Thesis, Quinlan's experiments were repeated resulting in average generalisation errors of 14.99% and 20.35% respectively. Differences in these figures are due to the randomness in splitting into the individual folds used in cross-validation. They are, however, in the same order of magnitude as reported in [34]. Reasons

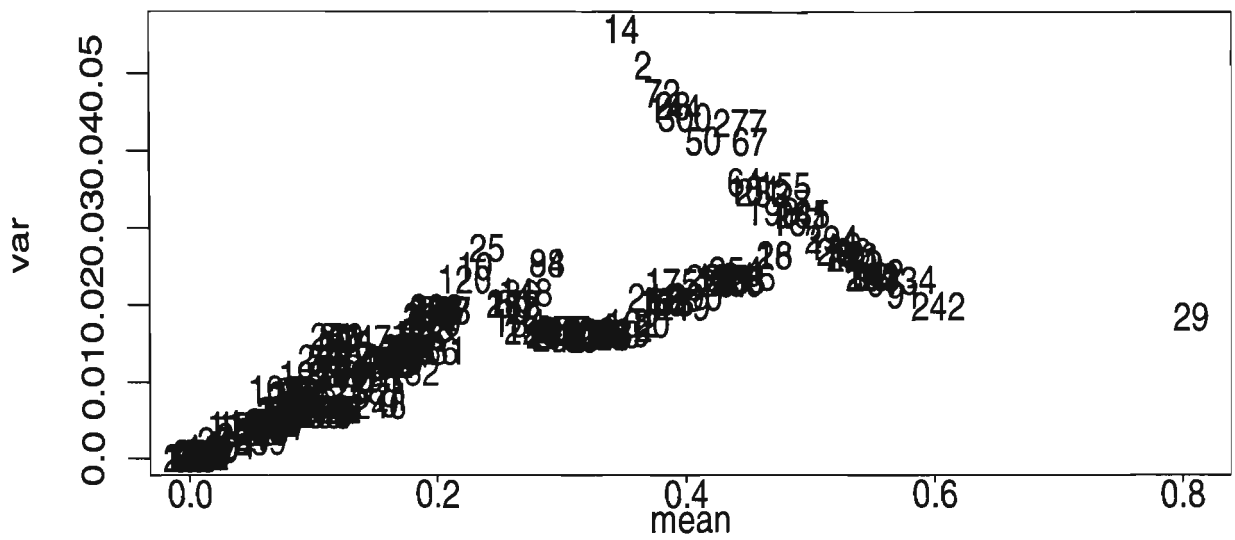

Figure 5.2: Plot of $\hat{Var}[edge(10,i)]$ vs. $\hat{E}[edge(10,i)]$: *colic* data.

for *AdaBoost* 's reduced performance are un-addressed to date and the edge and margin methodology previously outlined in Section 4.6 may shed some new light on this dataset and demonstrate the effectiveness of edge and margin diagnostics in dealing with difficult data.

As with all other datasets tested in this study, 10 boosting trials were run as outlined in Section 5.2. Refer to Figure 5.2 to see at least two distinct clusters of observations when $\hat{Var}[edge(10,i)]$ is plotted against $\hat{E}[edge(10,i)]$, with the plotted symbol representing the observation number. This may imply at least two distinct contexts with one context encompassing those observations with the positive relationship between $\hat{Var}[edge(10,i)]$ and $\hat{E}[edge(10,i)]$ and the other those observations exhibiting a negative relationship. The negative cluster incorporates a smaller section of the data than the larger positive cluster. This negative relationship is a potential flag for another smaller context being present

within the dataset. The positive cluster is significantly more concentrated with many observations having mean and variance both equal to zero. For some observations, symmetry about the line $\hat{E}[edge(10, i)] = 0.5$ (plotted as the X-axis) is also apparent in Figure 5.2. This symmetry is a key feature leading to provable results on the structure of the dataset. This feature and subsequent proofs are discussed in detail in Section 5.4.

There is also the notable observation (*29*) located to the far right of the plot, indicating a probable outlier. Examining observation *29* reveals a vector of missing values for all but 3 of the 23 input predictors. In further classification, this observation can deservedly be removed as an outlier. It is also interesting to note that observation *29* would have been flagged as a suspected outlier if the noise detection process from Section 4.6 was applied.

Analysis proceeds by partitioning the original (*colic*) dataset into smaller datasets according to the two clusters as observed in Figure 5.2. Cluster membership may now be used as the target (class) variable in a new classification problem. This may illuminate differences between the two clusters with respect to the input space and improve classification accuracy as confounding on contexts (clusters) will be removed.

5.2.2 Using Edgeflags on the *colic* Dataset

A new dataset named *colicflag* is created which results from appending a new response (class) variable to the *colic* dataset. This new response variable, $edgeflag$,

is assigned a value of 1 if an observation lies in the negatively sloping cluster located in the upper right hand corner of Figure 5.2 and 0 otherwise. This creates a new classification problem with $edgeflag = 0/1$ as the new class variable. A model built on $edgeflag$ using the initial 23 predictor variables may lead to the discovery of difficult border regions or subcontexts present within the *colic* dataset. If $c4.5$ or other classification algorithms can find robust structure in differentiating $edgeflag = 0$ from $edgeflag = 1$ using the available 23 predictor variables, then it may be concluded that such structure forms boundary regions of uncertainty or differentiates between two contexts within the dataset. Because an aim of this study is interpretability, single trees are fitted as opposed to boosted ensembles since boosted models become un-interpretable very quickly.

The distribution of $edgeflag$ values over the original *yes/no* response is tabulated in Table 5.1. A total of 14.1% of the observations were flagged with $edgeflag = 1$. This is certainly a smaller subsection of the data than those observations assigned $edgeflag$ values of 0, as was observed in Figure 5.2. It is also worth noting for this dataset that a higher proportion of the initial response $=no$ are flagged with $edgeflag = 1$ as seen below in Table 5.1. This is indicative of the original response$=no$ being more difficult to classify.

Table 5.1: Distribution of $edgeflag$ over initial response : *colic* data.

	edgeflag=0	**edgeflag=1**
lesion surgical=no	79(72.5%)	30(27.5%)
lesion surgical=yes	183(95.8%)	8(4.2%)

Two methodologies may be applied in proceeding with the analysis on the *colicflag* data:

1. The original response (*yes/no*) may be included as a predictor resulting in a binary classification problem on $edgeflag$ with 24 input variables.

2. The original response (*yes/no*) may be omitted resulting in a binary classification problem on $edgeflag$ with the original 23 input variables.

It is still unclear as to the benefits of using the initial response (*yes/no*) as a predictor variable since the value of this variable will be unknown when the class of future observations is sought. After testing, however, it was seen that the only robust modelled separation of $edgeflag$ classes was obtained by including the original response (*lesion surgical=yes/no*) in the predictors. The original *(yes/no)* response was returned as the initial spilt on the decision tree, indicating different mechanisms occurring for different values of the initial response.

Conversely, building a single $c4.5$ decision tree with default options using all 300 observations in the *colicflag* dataset as the training set with 24 predictors (including the original response) resulted in 0.3% training set error. Testing the robustness of the resulting decision tree via 10-fold cross-validation resulted in almost identical trees being built for each of the 10 data folds. Average test set error based on 10-fold cross-validation was 0.34%. Refer to Table 5.2 for a comprehensive listing of generalisation errors estimated via 10-fold cross-validation for all trials on the *colic* dataset.

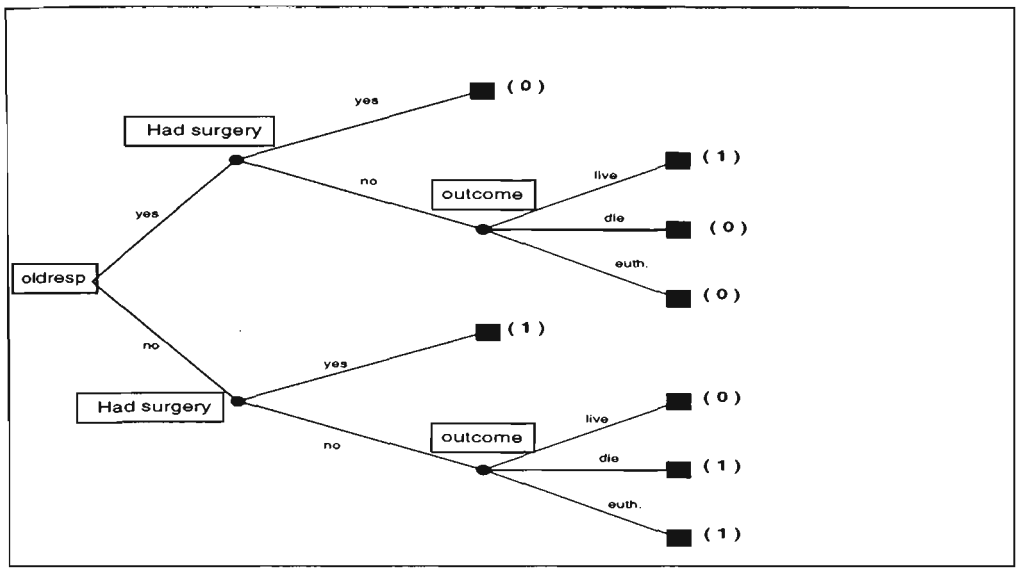

Figure 5.3: Decision tree for $edgeflag = 0/1$ response : *colic* data.

The decision tree resulting from classification of *edgeflag=0/1* using 24 input predictors is drawn as Figure 5.3, with the numbers in brackets on the leaf nodes referring to the predicted value of $edgeflag$ as (0) or (1). Note that the two subtrees beyond the '*Had surgery*' split variable are the inverse of each other. A possible model and the derivation of edge statistics for this behaviour is introduced in Section 5.6.2.

The problem with the decision tree drawn as Figure 5.3 is the use of the original response variable as the initial split variable, indicating that it is key in determining the structure difference between $edgeflag = 1$ ('difficult') observations and $edgeflag = 0$ ('standard') observations. Having a two-stage model requiring prior knowledge of the original class variable will be futile in predicting future unseen cases.

However, building a decision tree for the response variable $edgeflag$ without the

original response variable (*lesion surgical =yes/no*) included in the predictors resulted in significantly higher training and test set errors. Also apparent was a highly complex tree unable to be pruned sensibly. Pruning resulted in a root node classifying all $edgeflag$s to the default value of 0, hardly a robust solution. This phenomenon still demonstrates something of interest in the *colic* dataset (and other UCI datasets as will be seen in later sections of this Chapter). Boosting results in deteriorated generalisation error on the *colic* data and it can be seen that the information provided by the classification variable is incomplete since this variable is required to separate the data into clearly observable clusters. A conjecture is that boosting is more likely to fail when there is more than one context within the dataset or there is more than one mechanism of classifying the original data using the input predictors available. This is seen later in Section 5.6.2 where such a mechanism requires the presence of a sub-model which is the inverse of the main model classifying the majority of the dataset.

However, if the value of the target class on unseen data is unpredictable given the input data available, we will have to default to all new observations belonging to the larger cluster using this method. This may seem ad-hoc but we see in Section 5.2.3 that the classification model built on the larger subsection of the *colic* data is highly generalisable with simple structure . Section 6.1.1 introduces an alternative method of estimating generalisation error which applies to the heuristic of assigning all unseen observations to the majority cluster when cluster membership is unable to be determined given the input variables available.

5.2.3 Second Stage Classification of *colic* Dataset

Using the $edgeflag$ variable defined in Section 5.2.2, the *colic* dataset is split into two distinct datasets. These datasets are labelled *horsec0* and *horsec1* respectively, with 0 and 1 representing the value of $edgeflag$ determined in Section 5.2.2. The number of observations in each dataset is 262 and 38 respectively. The initial binary classification problem of *surgery required=yes/no* using 23 input predictors may then be re-estimated for each sub-dataset (*horsec0, horsec1).* For interpretability, a $c4.5$ decision tree using default values was used as the classification method. The results are encouraging with training error for the *horsec0* dataset being only 0.38% (as opposed to 12.7% on the entire *colic* dataset). Figure 5.4 shows the simple decision tree produced for the *horsec0* dataset in classifying the original *surgery required=yes/no* response. The leaf nodes refer to the predicted value of *surgery required.* Recall the *horsec0* dataset comprises the 262 observations which were flagged with $edgeflag = 0$ from the original boosting and edge analysis. These observations are deemed 'standard' according to the notion of belonging to the main cluster with a positive relationship between $\hat{Var}[edge(10,i)]$ and $\hat{E}[edge(10,i)]$. This is confirmed in the low training and test set error obtained on these data points. Figure 5.5 shows the resulting decision tree for classifying *lesion surgical=y/n* on the *horsec1* dataset.

It can be seen that the tree built for the *horsec0* dataset is inverse to that built for the *horsec1* dataset. The negative and positive relationships observed in Figure 5.2 are symptomatic of this result. Refer to Section 5.6.2 for more detail on the

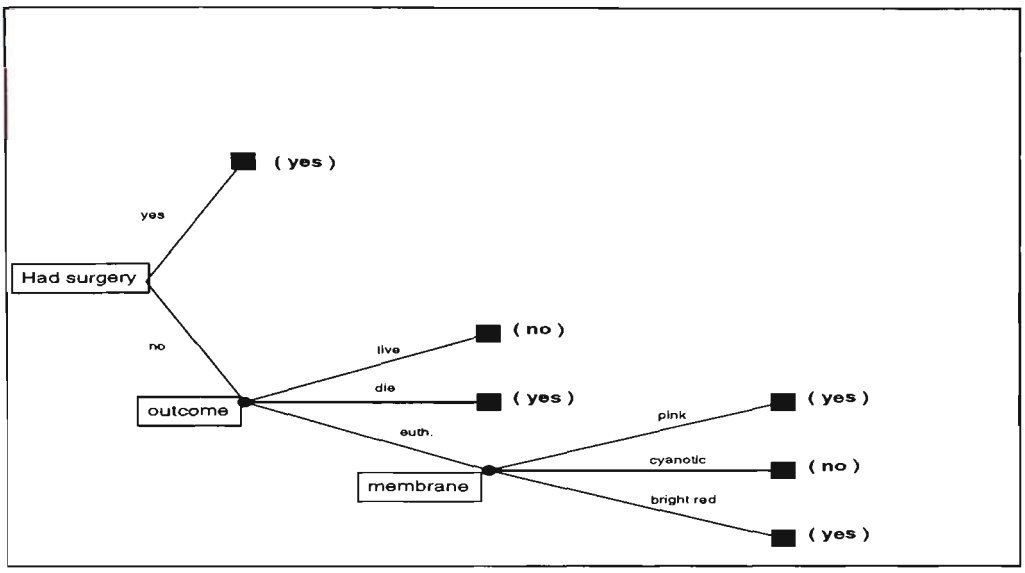

Figure 5.4: Decision tree for *lesion surgical=yes/no* response : *horsec0* data.

mechanism causing this.

Generalisation error as estimated via 10-fold cross-validation is reported in Table 5.2. The **mean** column refers to the mean generalisation error (%) as estimated on the leave-out fold in a 10-fold cross-validation . The bracketted figure alongside the mean refers to the standard deviation of this error as estimated on the 10 leave-out folds.

It should be noted, however, that on the *horsec1* dataset, test sets will only contain 3-4 observations due to the small size of the segregated dataset. These are probably not representative and will result in a higher standard deviation of the error estimate. Dramatic improvement in generalisation error is noted for the *horsec0* dataset but this generalisation error is not representative given that

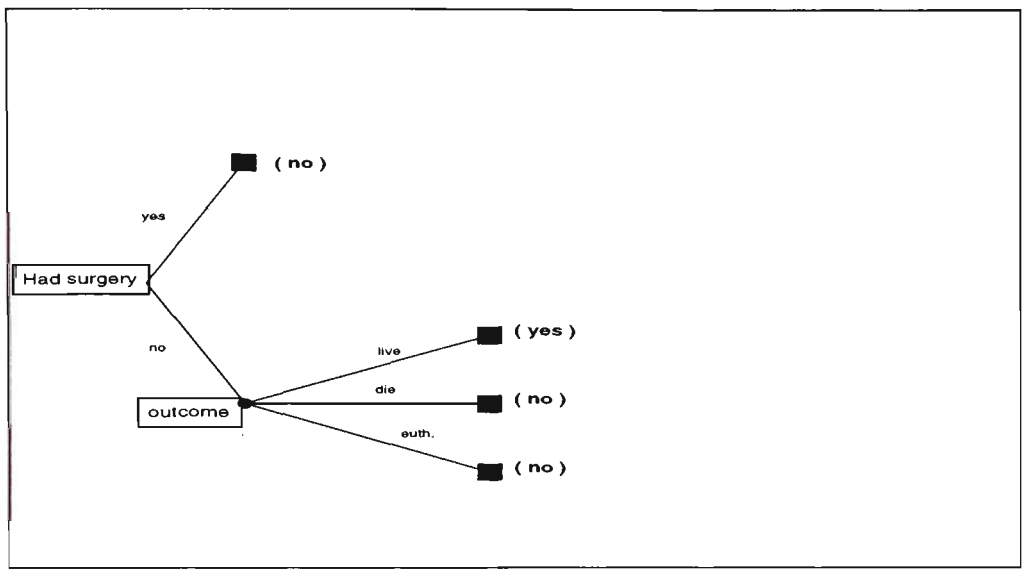

Figure 5.5: Decision tree for *lesion surgical=yes/no* response : *horsec1* data.

this is a classifier built on only one cluster which only holds for observations belonging to the majority cluster. For unseen observations, cluster membership is unpredictable and we default to the classifier built on the majority cluster. This will almost certainly give incorrect predictions for any observations in the smaller (inverse) cluster, hence inflating generalisation error. Section 6.1.1 in Chapter 6 outlines a method of generalisation error which is robust to this situation and gives an unambiguous estimate of generalisation error.

Table 5.2: Cross-validation results for *colic* dataset.

dataset	**response**	**method**	**mean**
colic	Lesion surgical (y/n)	*c*4.5 single	14.99(8.76)
colic	Lesion surgical (y/n)	*c*4.5 10 boost	21.01(7.21)
colic2flag	edgeflag (0/1)	*c*4.5 single with original response	0.34(1.08)
colic2flag	edgeflag (0/1)	*c*4.5 single w/out original response	11.66(8.33)
horsec0	Lesion surgical (y/n)	*c*4.5 single	1.53(2.68)
horsec0	Lesion surgical (y/n)	*c*4.5 10 boost	0.38(1.20)
horsec1	Lesion surgical (y/n)	*c*4.5 single	3.33(10.25)
horsec1	Lesion surgical (y/n)	*c*4.5 10 boost	3.33(10.25)

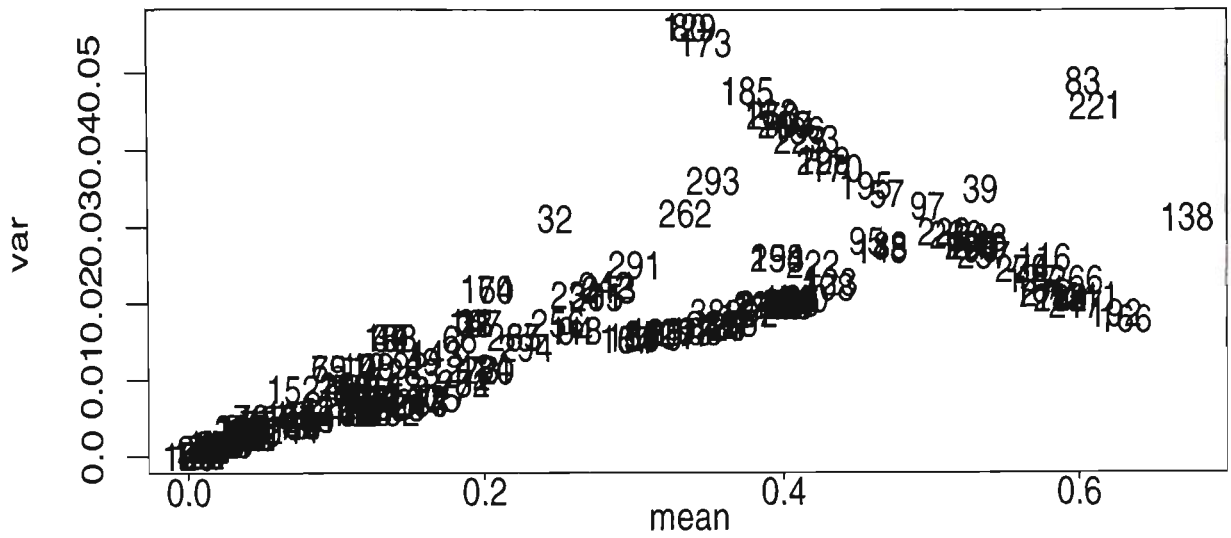

Figure 5.6: Plot of $\hat{Var}[edge(10,i)]$ vs. $\hat{E}[edge(10,i)]$: *heart-h* data.

5.2.4 Results on the *heart-h* Dataset

This UCI [2] dataset was collected from 294 patients in Hungary, with the *heart-h* database requiring binary classification as to whether or not a patient had heart disease. Sick patients were assigned a response class variable of *1* and well patients a class of *0*. There are 13 input predictors such as sex, age and cholesterol. Using the method outlined in Section 5.2, 10 iteration boosting was applied and the resulting edge statistics calculated and plotted in Figure 5.6, with the plotted symbol referring to the observation number.

From Figure 5.6, three (3) possible outliers are observed (obs. *83, 221, 138*). On inspection there are no obvious reasons for these points to be discarded as outliers and hence they are retained. There is also the distinct possibility of 2 contexts

being present as seen in the observations exhibiting a positive $\hat{Var}[edge(10,i)]$ versus $\hat{E}[edge(10,i)]$ relationship perhaps opposing those exhibiting a negative one. As with the *colic* data in Figure 5.2, symmetry about the line $\hat{E}[edge(10,i)] = 0.5$ is noted for a selection of observations in Figure 5.6.

5.2.5 Using Edgeflags on the *heart-h* Dataset

A new dataset, *hearthflag* is created by appending a new class variable to the existing *heart-h* dataset. The class variable, *edgeflag*, is assigned a value of 1 if an observation falls into the negative cluster appearing in the top right hand corner of Figure 5.6 and a 0 otherwise. In total, 12.03% of all observations are assigned values of $edgeflag = 1$. Table 5.3 shows the distribution of $edgeflag$ over the original response variable for the *hearthflag* dataset. It can be seen that the original response of *sick(1)* has a higher proportion of its observations flagged as $edgeflag = 1$. (28.3% as opposed to 4.3%), implying that *"sick"* patients are more difficult to classify or comprise the majority of the smaller sub-context.

Table 5.3: Distribution of $edgeflag$ over initial response : *heart-h* data.

	edgeflag=0	**edgeflag=1**
well (0)	180(95.7%)	8(4.3%)
sick (1)	76(71.7%)	30(28.3%)

As was observed for the *colic* data, when trying to fit a tree to the data to differentiate $edgeflag = 0$ versus $edgeflag = 1$, the tree requires the original response to make robust differentiation between $edgeflag$ classes. Refer to Figure 5.7, where the resulting decision tree splits on the original response (*sick/well*) and sub-trees beyond the initial split are the inverse of each other.

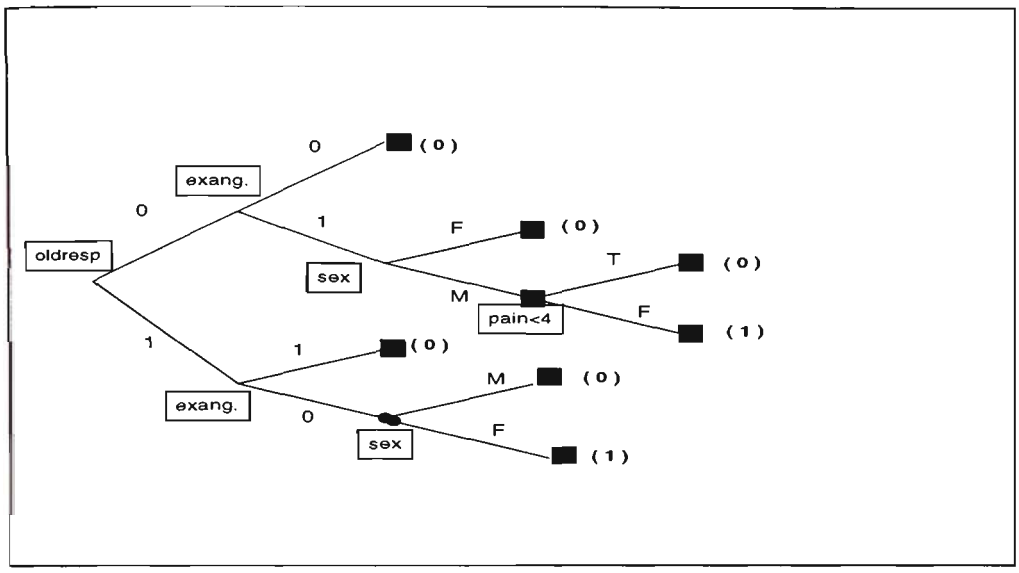

Figure 5.7: Decision tree for $edgeflag = 0/1$ on the *hearthflag* dataset

5.2.6 Second Stage Classification of the *heart-h* Dataset

New datasets are created by splitting the *hearthflag* dataset according to the value of $edgeflag$. The resulting *hearth1* and *hearth0* datasets can now be used in 2 new prediction problems to classify *sick* versus *well* patients. Figure 5.8 shows the resulting tree which classifies *sick/well* for the 87.97% of data assigned to the *hearth0* dataset. Figure 5.9 shows the resulting tree which classifies *sick/well* for the 12.03% of data assigned to the *hearth1* dataset, with bracketted digits on the leaves referring to the predicted values of *sick(1)* versus *well(0)* patients. Table 5.4 gives generalisation error results estimated via 10-fold cross-validation for the *heart-h* dataset and its permutations.

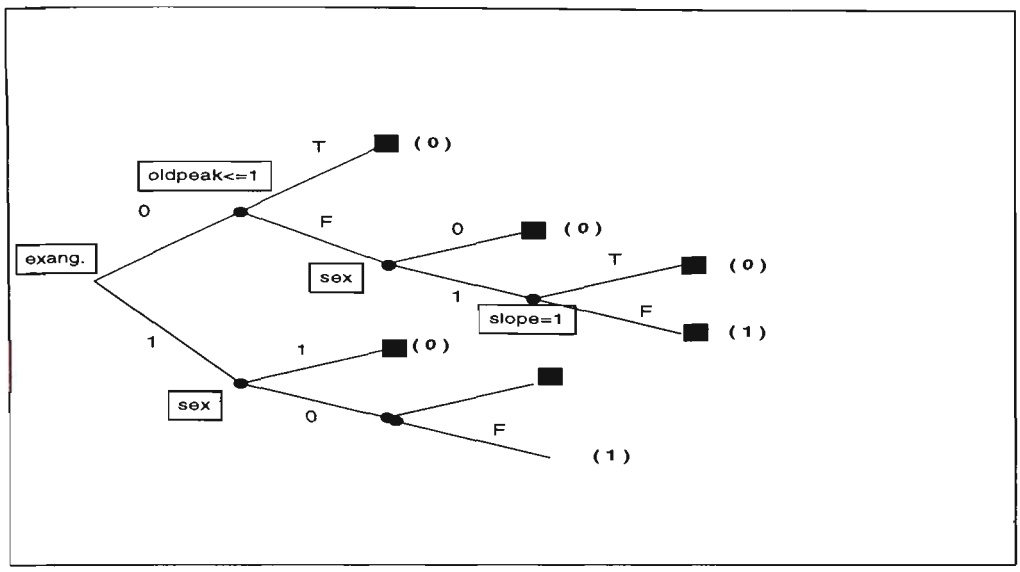

Figure 5.8: Decision tree for *sick* = 0/1 on *hearth0* dataset

Table 5.4: Cross-validation results for *heart-h* dataset.

dataset	**response**	**method**	**mean**
heart-h	Patient (sick/well)	*c*4.5 single	23.73(26.59)
heart-h	Patient (sick/well)	*c*4.5 10 boost	26.81(17.41)
hearthflag	edgeflag (0/1)	*c*4.5 single	3.75(3.40)
hearth0	Patient (sick/well)	*c*4.5 single	7.94(6.94)
hearth0	Patient (sick/well)	*c*4.5 10 boost	5.21(7.33)
hearth1	Patient (sick/well)	*c*4.5 single	6.66(14.04)
hearth1	Patient (sick/well)	*c*4.5 10 boost	23.27(35.30)

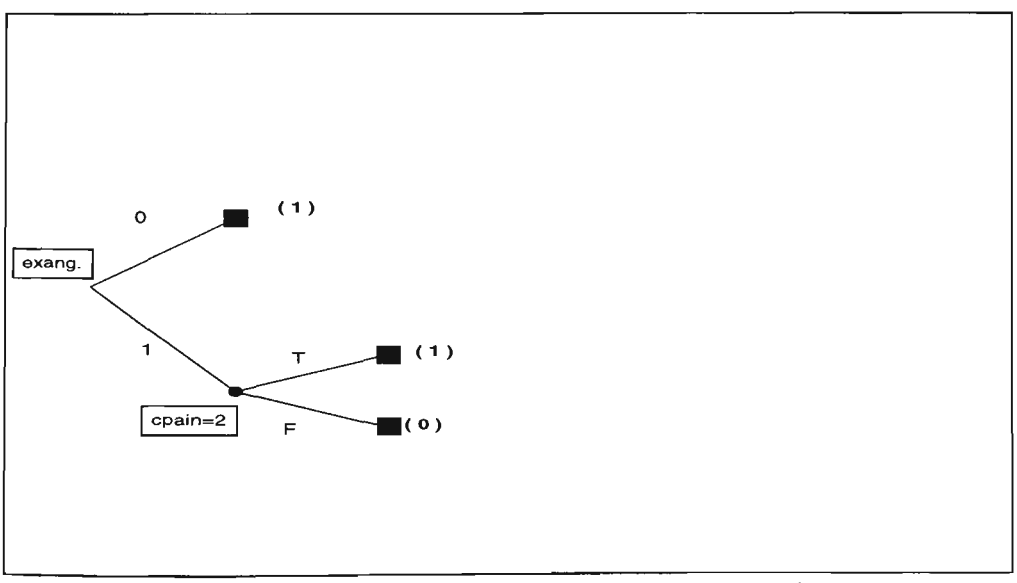

Figure 5.9: Decision tree for *sick* = 0/1 on *hearth1* dataset

5.3 Revisiting the *BHP* data

In Chapter 4, the *BHP* dataset was used as an example dataset in the testing of the noise detection procedure. What the procedure unearthed was a dataset comprising 2 contexts. One section (observations 2480-3021) comprises low noise data with consistently low edges and a significantly higher proportion of class=0 (defect free) observations. The remainder of the *BHP* dataset had a significantly higher defect rate with greater edge variation. The difference in operating condition between these two subsections was predictable. Recall from Section 4.3.1, the *BHP* data was successfully partitioned into smaller datasets; *BHPlarge* and *BHPsmall* according to the decision tree drawn in Figure 4.4.

It would be interesting to apply the edge analysis procedure undertaken on the *colic* and *heart-h* datasets in Sections 5.2.1 - 5.2.6 to the *BHP* data. Will this new methodology detect the already known change in context detected in Section 4.3.1?

Shown as Figure 5.10, is the plot of $\hat{Var}[edge(10,i)]$ vs $\hat{E}[edge(10,i)]$ for the entire *BHP* dataset, with the plotted symbols representing the observation number. In Figure 5.10, distinct bands can be seen, possibly inferring different contexts or clusters of observations with similar properties. Also of interest is the smaller cluster of observations located to the far right of plot, possibly representing a cluster of outliers.

In Section 4.3, we observed the sequence of observations in the range $2480-3021$

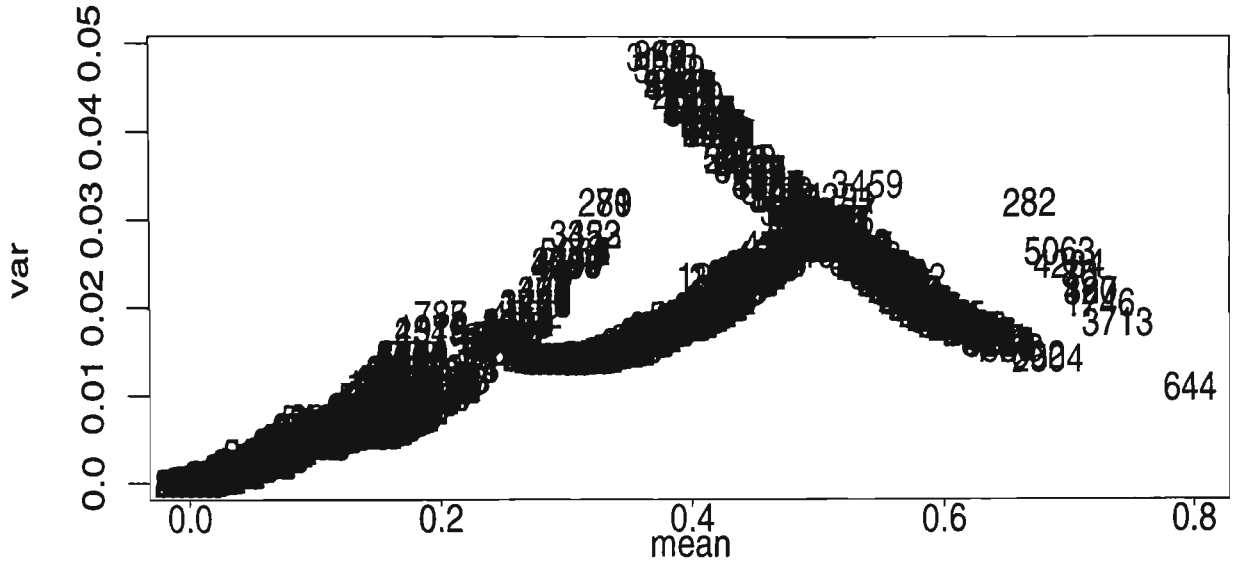

Figure 5.10: Plot of $\hat{Var}[edge(10,i)]$ vs. $\hat{E}[edge(10,i)]$: *BHP* data.

being modelled as having significantly different input conditions than the remainder of the *BHP* data. By re-plotting Figure 5.10 for observations $2480 - 3021$ only, we would expect to see these observations appear in a common location on the plot. The new plot is included as Figure 5.11.

From this plot it can be seen that the observations from $2480-3021$ do not strictly lie in only one of the distinct bands seen in Figure 5.10. There also appear to be 2 possible outliers in observations *3019* and *2489*. Although discouraging, Figure 5.10 is still exhibiting the signature mean/variance behaviour of the *colic* and *heart-h* datasets. i.e. a smaller group of observations with a negative relationship between $\hat{Var}[edge(10,i)]$ and $\hat{E}[edge(10,i)]$ and a selection of observations showing symmetry about the line $\hat{E}[edge(10,i)] = 0.5$.

It is still of interest to proceed with splitting the *BHP* dataset into *BHPlarge* and *BHPsmall* according to the tree drawn as Figure 4.4 from Chapter 4 and

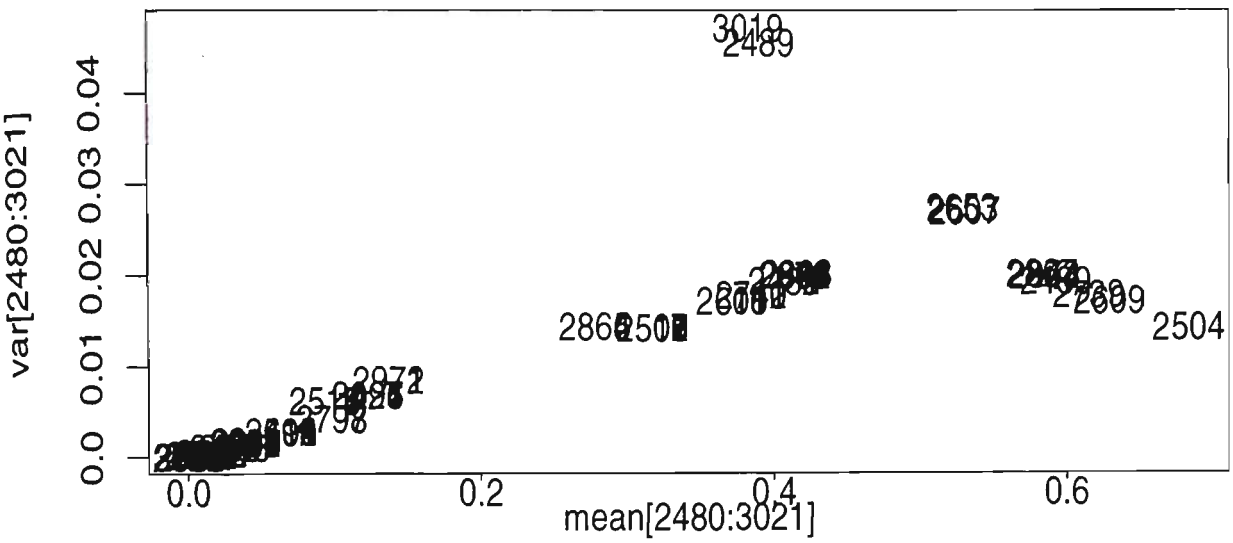

Figure 5.11: Plot of $\hat{Var}[edge(10,i)]$ vs. $\hat{E}[edge(10,i)]$: *BHP* data observations 2480-3021.

test edge diagnostics on each sub-dataset.

5.3.1 Edge Diagnostics and Boosting on the *BHPsmall* Dataset

The notion of 10 trial boosting and checking the resulting edge diagnostics may be used to detect outliers, difficult border regions or sub contexts present in the *BHPsmall* dataset. The already high accuracy noted in Section 4.4 may be improved when further edge diagnostics are applied. Plotting $\hat{Var}[edge(10,i)]$ versus $\hat{E}[edge(10,i)]$ in Figure 5.12 shows a group of points with high final mean edge above 0.5 and a pair of observations with high edge variance. Again, symmetry about the line $\hat{E}[edge(10,i)] = 0.5$ is evident in Figure 5.12 (as was also apparent in Figures 5.2 and 5.6 for the *colic* and *heart-h* datasets respectively). Figure 5.12 also flags the possibility of 2 outliers in observations *2607* and *2653*.

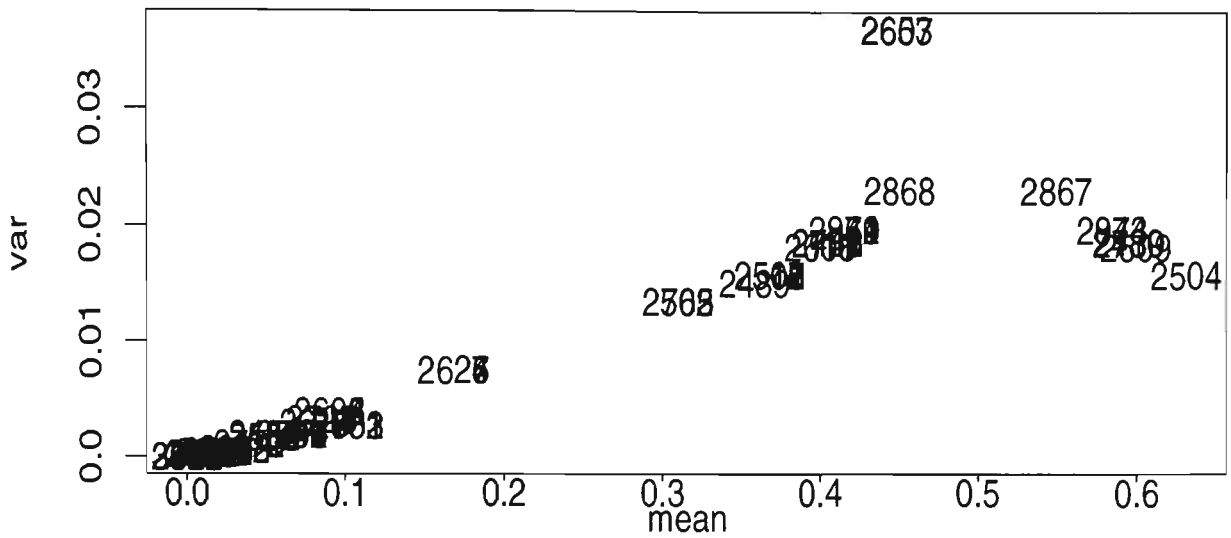

Figure 5.12: Plot of $\hat{Var}[edge(10,i)]$ vs. $\hat{E}[edge(10,i)]$: *BHPsmall* data.

A new variable, $edgeflag$ was appended to the existing *BHPsmall* dataset, creating a new dataset named *BHPsmallflag*. The $edgeflag$ variable is assigned a value of 1 if an observation falls in the band of observations exhibiting a negative relationship between $\hat{Var}[edge(10,i)]$ and $\hat{E}[edge(10,i)]$ and a 0 otherwise. A total of 2.03% of observations were assigned an $edgeflag$ value of 1. Refer to Table 5.5 for the distribution of $edgeflag$ across original response for the *BHPsmall* dataset. It can be seen that 62.5% of observations with original class defect=1 have $edgeflag = 1$ as opposed to 0.2% for those observations with no defect, implying defects are more difficult to predict or form the majority of the sub-context.

A new classification problem arises with the class variable being $edgeflag$ and predictor variables being the existing input predictors from the *BHPsmall* dataset.

Table 5.5: Distribution of *edgeflag* over initial response : *BHPsmall* data.

	edgeflag=0	**edgeflag=1**
defect=0	525(99.8%)	1(0.2%)
defect=1	6(37.5%)	10(62.5%)

Decision trees using *Splus' rpart* function and *c*4.5 both failed to classify any of the $edgeflag = 1$ observations correctly. Logistic regression and neural networks also failed to converge. These results indicate that the 10 points flagged in the *BHPsmall* dataset are most likely outliers, or that vital predictor variables have not been measured as no common properties of the $\mathbf{x}_i$'s for these observations can be determined. Interestingly, the percentage of observations flagged with $edgeflag = 1$ is 2.03% which is significantly lower than that for the *colic* and *heart-h* datasets, both of which have inverse models for a subset of the data. This is initial evidence of setting a threshold on the percentage of a dataset being flagged with $edgeflag = 1$ to indicate the presence of inverse models.

5.3.2 Edge Diagnostics and Boosting on the *BHPlarge* dataset

Both of the single tree results in Section 4.5 on the *BHPlarge* dataset point to an opportunity to apply boosting and edge diagnostics as there is the possibility of significant noise, or several independent operating regimes, all of which result in different defect prediction mechanisms.

Boosting with 10 iterations using *c*4.5 with default options as the base learner was applied to the *BHPlarge* dataset. Edge values for each observation were calculated and stored after each iteration. A plot of $\hat{Var}[edge(10, i)]$ versus $\hat{E}[edge(10, i)]$

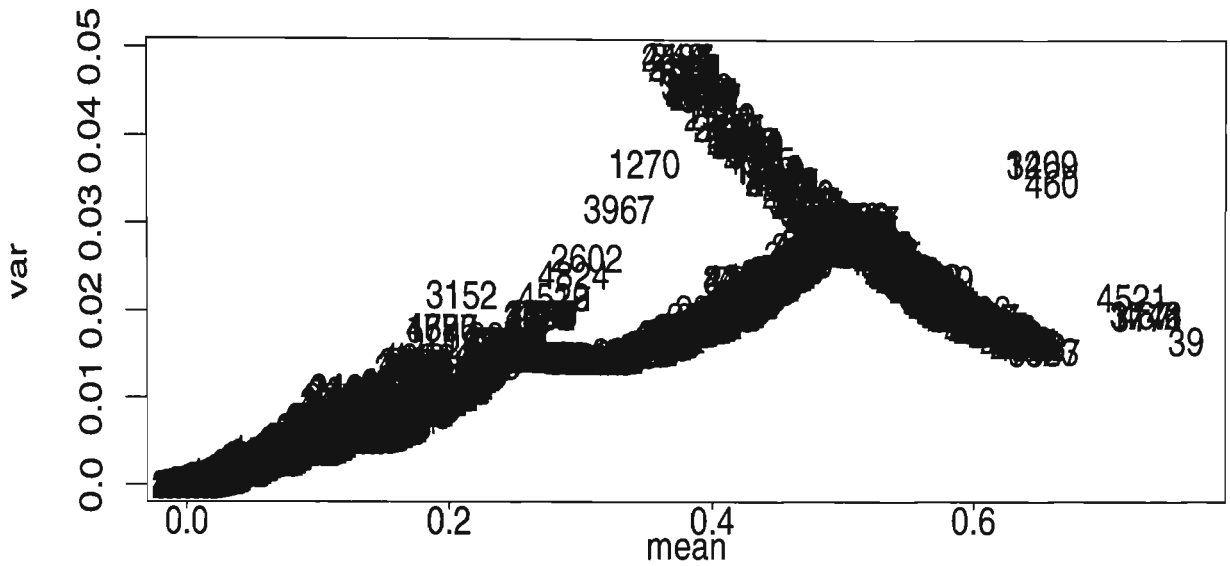

Figure 5.13: Plot of $\hat{Var}[edge(10,i)]$ vs. $\hat{E}[edge(10,i)]$: *BHPlarge* data.

is drawn as Figure 5.13, with the plotted symbols representing the observation number. Again, a smaller cluster of observations is evident on the far right of Figure 5.13, possibly indicating outliers, with the signature negative cluster relationship as seen for the *colic, heart-h* and *BHPsmall* datasets.

Analysis proceeds by separating the original *BHPlarge* dataset into the clusters as appearing in Figure 5.13. Cluster number may now be used as a target (class) variable in a new classification problem to enable us to determine possible differences in the input space for each cluster. This may illuminate disparity between the two contexts and improve classification accuracy as the entire *BHP* dataset may contain distinct regions operating under different conditions or process control. The separation is achieved by appending a new variable, *edgeflag* to the existing *BHPlarge* dataset. Where an observation falls into the cluster exhibiting a negative relationship between $\hat{Var}[edge(10,i)]$ and $\hat{E}[edge(10,i)]$, the value

of *edgeflag* is set at 1, and 0 otherwise. A total of 11.4% of observations were assigned *edgeflag* values of 1. Refer to Table 5.7 for the distribution of *edgeflag* values across original response for the *BHPlarge* dataset. As was seen for the *BHPsmall* dataset, the majority of *edgeflag=1* values occurred for original response=1. i.e. defects are more difficult to predict or form the majority class in the smaller sub-context.

Table 5.6: Distribution of $edgeflag$ over initial response : *BHPlarge* data.

	edgeflag=0	**edgeflag=1**
defect=0	3960(92.8%)	308(7.2%)
defect=1	126(19.8%)	511(80.2%)

A new classification problem now arises with the target variable being *edgeflag*. as with all datasets tested in this Chapter, there is the option of using only the original predictor variables available in order to classify *edgeflag*, or including the original response (defect=0/1) in the predictors to form a more robust model. After testing both methods, the only robust split between $edgeflag = 0/1$ was made when the original response (defect=0/1) was included in the predictors. The resulting tree was still very complex with high generalisation error as estimated via 10-fold cross-validation , indicative of the suspected high level of noise present in the *BHPlarge* dataset. As with other datasets tested, having cluster membership undeterminable given the information available will be futile when trying to predict the class value of future unseen cases.

The original *BHPlarge* dataset is partitioned into two smaller datasets depending on the value of the new *edgeflag* variable. *BHPlarge0* and *BHPlarge1* are the

new datasets, with the last digit in the dataset name referring to the value of *edgeflag* applicable to that particular dataset. As with other partitioned datasets in this study, single decision trees and 10 iteration boosting was applied to each sub-dataset. Cross-validation results for all trials on the *BHPlarge* dataset are given in Table 5.7. It should be noted that boosting improves generalisation performance on both the *BHPlarge0* and *BHPlarge1* datasets.

Table 5.7: Cross-validation results for *BHPlarge* dataset.

dataset	**response**	**method**	**mean**
BHPlarge	defect (0/1)	*c*4.5 single	19.60(5.29)
BHPlarge	defect (0/1)	*c*4.5 10 boost	21.16(5.33)
BHPlargeflag	edgeflag (0/1)	*c*4.5 single	11.63(3.62)
BHPlarge0	defect (0/1)	*c*4.5 single	9.62(5.74)
BHPlarge0	defect (0/1)	*c*4.5 10 boost	7.68(4.69)
BHPlarge1	defect (0/1)	*c*4.5 single	28.34(11.60)
BHPlarge1	defect (0/1)	*c*4.5 10 boost	21.97(7.92)

5.4 Signature Behaviour Observed on $\hat{Var}[edge(10,i)]$ versus $\hat{E}[edge(10,i)]$ Plots

The argument of using the mean and variance of the edge is purely notional and we may question whether the data is behaving in the manner we'd expect given the logic presented in Section 5.1. There are certainly the group of observations with low variance and mean which capture the majority of the data. Outliers tend to fall singularly to the high mean or high variance areas of the $\hat{Var}[edge(10,i)]$ versus $\hat{E}[edge(10,i)]$ plot. Additionally, there are groups with high variance, symptomatic of flipping between correct and incorrect classification as boosting

proceeds.

This Chapter has presented empirical edge statistics results for two UCI datasets and a large industrial dataset, all of which were subject to $m = 10$ boosting trials. For each dataset, a plot of $\hat{Var}[edge(10, i)]$ versus $\hat{E}[edge(10, i)]$ was drawn in order to detect border regions or flag clusters of observations with behaviour deviant form the main body of data. In all cases three features were present in the $\hat{Var}[edge(10, i)]$ versus $\hat{E}[edge(10, i)]$ plots:

1. a large dense cluster displaying a positive relationship between $\hat{Var}[edge(10, i)]$ and $\hat{E}[edge(10, i)]$. Many observations in this cluster had both mean and variance of their edge equal to zero, indicating correct classification in all boosting rounds.

2. a smaller cluster with a negative relationship between $\hat{Var}[edge(10, i)]$ and $\hat{E}[edge(10, i)]$.

3. symmetry about the line $\hat{E}[edge(10, i)] = 0.5$ for a selection of observations, with corresponding $\hat{Var}[edge(10, i)]$ values below 0.03.

The features listed above are best visualised by the sketch plot drawn below as Figure 5.14. The smaller black oval represent the groups of observations which have symmetry about the line $\hat{E}[edge(10, i)] = 0.5$.

This Thesis suggests the partitioning of each dataset according to cluster membership and training a new classifier on each partition. This process results

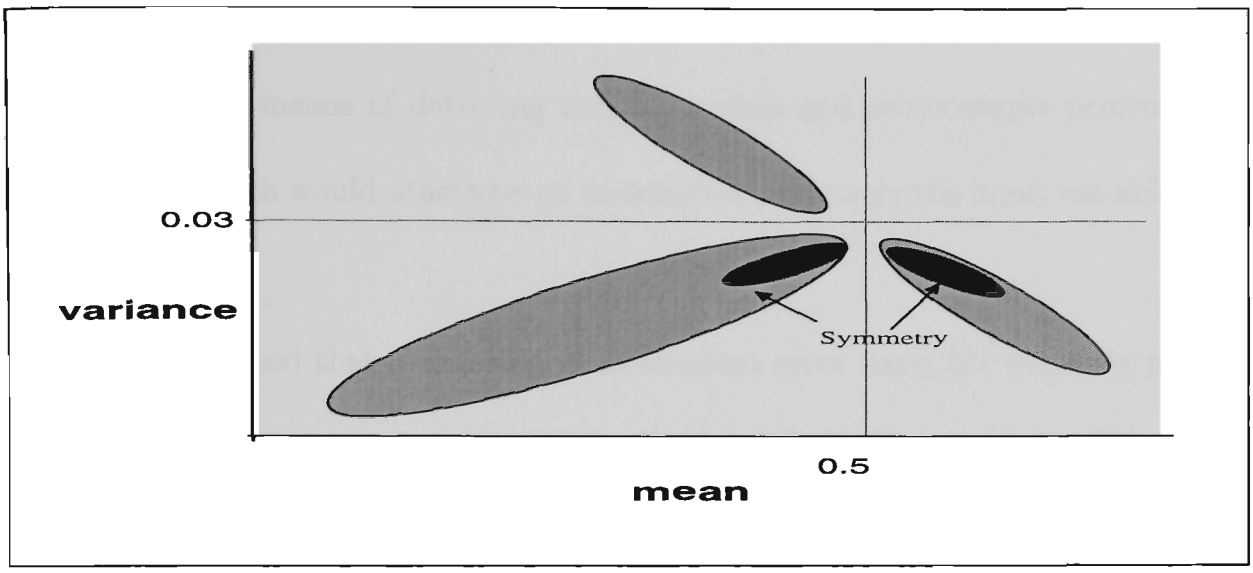

Figure 5.14: Example of features listed above for typical $\hat{Var}[edge(10, i)]$ versus $\hat{E}[edge(10, i)]$ plots.

in significantly lower generalisation error estimated over each partition as confounding on contexts or clusters is removed. The difficulty arises for unseen observations when the correct cluster is unknown. If the clusters identified via the $\hat{Var}[edge(10, i)]$ versus $\hat{E}[edge(10, i)]$ plots are able to be discriminated using only the predictor variables, it would be simple to use the resulting classifier to determine cluster membership for a new unseen observation. However, if the classification procedure fails to discriminate between clusters without using the original target variable, the optimal way to proceed is to assume all new observations are best modelled according to the robust classifier trained on the larger partition. We then proceed by applying the classifier built on the majority cluster to all unseen observations.

It seems ad-hoc to blindly apply this procedure to all unseen data but the empirical studies presented in Sections 5.2.1 - 5.2.6 have demonstrated this to be an effective means of detecting inverse models and sub-contexts present in the dataset which would otherwise go undetected using only the input variables available.

It is recognised that estimating generalisation error using the majority partition would give overly optimistic results. Chapter 6 introduces a mechanism whereby generalisation error is measured as the sum of two important components. When this more appropriate measure is applied, overall performance is not degraded.

At this stage it is salient to shift from empirical work and develop a theoretical basis for the inverse sub-context hypothesis. This work is carried out in Sections 5.7 and 5.7.1.

5.5 Discussion

In many cases, the $\hat{Var}[edge(10, i)]$ versus $\hat{E}[edge(10, i)]$ procedure identifies clusters no differently than simply selecting observations which were initially misclassified by a single decision tree on the entire dataset. Although discouraging, this phenomenon brings the following points to light:

- The usual practice of applying boosting to the entire dataset gives difficult observations an opportunity to be modelled correctly by the classifier. It was seen for the datasets tested, that even after 20 boosting iterations, the boosted ensemble was no closer to making consistent predictions on these

groups of observations. The variance of the edge also remains large as boosting flips between correct and incorrect classification on these observations.

- Closer inspection revealed these observations to occupy a similar region in predictor space to that of the 'core' data but the class labels are reversed. This could be attributed to mislabelling of the output or absence of a relevant predictor variable.

- In all datasets tested, the initial decision tree trained on the entire dataset was significantly more complex in its attempt to accommodate for the observations falling into the smaller cluster. A very simple robust tree with low generalisation error results when $c4.5$ is trained on the main cluster only. Using the tree trained on a subset of the data does not significantly deteriorate generalisation performance, as demonstrated in Table 6.3.

- Boosting is often successful within the 'core' data cluster, even when unsuccessful on the entire dataset.

5.6 Inverse Model Hypothesis for Binary Classification

It is now conjectured that inverse or closely inverse models are present and detectable within many datasets. The signature $\hat{Var}[edge(10,i)]$ versus $\hat{E}[edge(10,i)]$ and subsequent $edgeflag$ analysis are symptoms of this behaviour. It is suspected that the extent to which boosting degrades classification performance is a function of the percentage of the dataset contained in the inverse sub context. Also,

the higher the percentage of observations with $edgeflag = 1$, the more likely it is that an inverse model will be built. Refer to Table 5.8 for a summary of $edgeflag$ distribution for all datasets tested in this Thesis.

Table 5.8: Summary of datasets and presence of outliers and subsections of data modelled by inverses.

dataset	**% data with** $edgeflag = 1$	**inverse model?**
colic	14.13	yes
heart-h	12.03	yes
vote	3.4	no
BHPsmall	2.03	no
BHPlarge	12.99	no

Results to date on the *colic*, *hearth-h* and *BHP* datasets show a major cluster comprising approximately 90% of the data with a well defined, highly accurate classification rule, $h_1(\mathbf{x})$, say. The remaining 10% of the data can be modelled by an equally accurate classification rule which appears to be the inverse of $h_1(\mathbf{x})$. This may be the explanation of the negative $\hat{Var}[edge(10,i)]$ versus $\hat{E}[edge(10,i)]$ trends observed in the edge diagnostic plots to date. This inverse model hypothesis is simulated empirically in Section 5.6.2, with theoretical variance and mean expressions for inverse models being derived in Section 5.7.1.

5.6.1 The Formal Hypothesis

The inverse sub-context hypothesis is posed more formally below:

Let there be N training observations of the form $\{(\mathbf{x}_i, y_i), i = 1\ldots, N\}$. Assuming a 2-class problem, we have $y_i \in 0, 1$. Let N_1 of these observations be modelled

according to $h_1(\mathbf{x})$ and $(N - N_1)$ according to $1 - h_1(\mathbf{x})$.

Define $\mathbf{X}_1 = \{\mathbf{x}_i : i \in 1 \ldots N_1\}$ and $\mathbf{X}_2 = \{\mathbf{x}_i : i \in N_1 + 1, \ldots N\}$.

Let p_1 represent the probability of mis-classification according to $h_1(\mathbf{x})$ over $\mathbf{X}_1$. By assuming all unseen observations will be classified by the majority model, the probability of mis-classification for observations in $\mathbf{X}_2 = 1 - p_1$. Also assume $N_1 >> (N - N_1)$. But p_1 only holds for the N_1 observations correctly modelled under $h_1(\mathbf{x})$ and $1-p_1$ for the remaining $(N-N_1)$ observations correctly modelled according to $h_2(\mathbf{x})$. So,

$$\begin{aligned} P_{h_1(\mathbf{x}),\mathbf{X_1}}(misclass) &= p_1 \\ P_{h_1(\mathbf{x}),\mathbf{X_2}}(misclass) &= 1 - p_1 \end{aligned}$$

For *AdaBoost.M1*, p_1 will change at every iteration as the data is reweighted. Formulating theoretical $\hat{E}[edge(10, i)]$ and $\hat{Var}[edge(10, i)]$ may be intractable due to the non-independence of the p_j's. However, if we loosen the restrictions of reweighting placed by *AdaBoost* and formulate the problem as a bagging problem, this dependence problem can be overcome. At this stage it is worth noting that observations may not 'equalise' if there were different prior probabilities of mis-classification (refer to Chapter 4 as this notion forms the basis of the noise detection procedure).

5.6.2 Simulations on the Proposed Inverse Model

To check the hypothesis of datasets having an inverse sub-context, a simulation was carried out on a contrived dataset with *300* observations as per the *colic*

dataset. The 38 observations flagged with $edgeflag = 1$ in the *colic* dataset were marked with *context=1* and the remainder with *context=0*. At iteration 1, the probability of mis-classification of context 0 observations under the resulting hypothesis was randomly assigned to be e_1. Assuming a 2-class problem, the probability of mis-classification of context 1 observations would be equal to $1-e_1$ (since context 1 represents an inverse model to context 0). The $I_1(i)$'s were simulated using a simple Bernoulli Monte Carlo process and the resulting w_{i1}'s, ϵ_1 and β_1 calculated according to the *AdaBoost* (2.1) algorithm before proceeding to the next iteration. At iteration 2, the probability of misclassification, e_2, was assigned randomly to context 0 observations and $1-e_2$ to context 1 observations. This iterative process was repeated 10 times. The edge ($edge(m,i)$) for each observation was stored after each iteration and final calculations of $\hat{E}[edge(10,i)]$ and $\hat{Var}[edge(10,i)]$ edge were made after the 10 simulated iterations had been completed. This is certainly a simplified model since the $e_j/1-e_j$ relationship does not strictly hold due to noise components and if so, boosting would terminate due to ϵ_j becoming larger than 0.5. However, for the purposes of this exercise this model is retained as it has the same base structure as the *colic* data excluding noise. The aim of the simulation is to reproduce versions of the $\hat{Var}[edge(10,i)]$ versus $\hat{E}[edge(10,i)]$ plots for the *300* observations and confirm that 2 clusters are observed and that the observations falling into each cluster are those deliberately assigned to each context. This scenario will exaggerate the negative and positive slopes on the $\hat{Var}[edge(10,i)]$ versus $\hat{E}[edge(10,i)]$ plots as the model

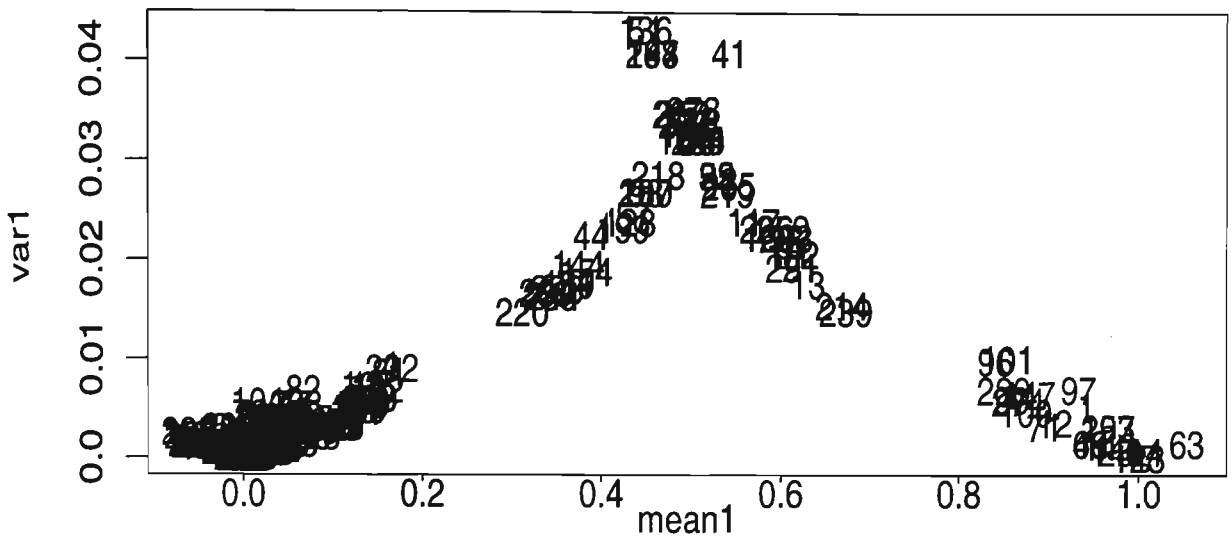

Figure 5.15: Plot of $\hat{Var}[edge(10,i)]$ vs. $\hat{E}[edge(10,i)]$ on simulated model using 300 observations (as per *colic* data) with pre-assigned clusters.

does not include any noise and is restrictive on the $e_j/1-e_j$ relationship. Refer to Figure 5.15 for a plot of the simulated $\hat{Var}[edge(10,i)]$ versus $\hat{E}[edge(10,i)]$ results. By definition, average edges below 0 and above 1 are impossible but due to the nature of the simulated model, and the looseness of assumptions, resulted in some unfeasible edge statistics. This does not, however, lessen the impact of the simulated results.

Examining Figure 5.15, distinct bands are clear but are exaggerated from the true $\hat{Var}[edge(10,i)]$ versus $\hat{E}[edge(10,i)]$ plots using the actual *colic* dataset (Figure 5.2). If thresholds are set to capture those observations in the negative sloping band, all bar 13.5% (5) of the observations set up as context 1 are contained within the negative sloping band. Figure 5.15 shows results for one simulation run only, but is typical of repeatable behaviour observed when more simulations

are executed. The $\hat{Var}[edge(10,i)]$ versus $\hat{E}[edge(10,i)]$ plot is typically triangular with those observations falling on the negatively sloping side belonging to the smaller sub-context (and inverse model) in almost 90% of cases.

The trial described above is repeated on the *hearth-h* dataset whereby 38 observations are assigned to context 1 and the remaining observations to context 0. The 38 observations selected are those which were assigned an $edgeflag$ value of 1 in the *hearthflag* dataset. Monte Carlo Simulation was used to simulate Bernoulli errors of $e_1, 1-e_1$ for contexts 0 and 1 respectively. Weights were updated according to the *AdaBoost* algorithm before proceeding to the next iteration. Edge values were stored for each observation after each iteration and the resulting $\hat{E}[edge(10,i)]$ and $\hat{Var}[edge(10,i)]$ statistics calculated after the 10 iterations had been completed. Similar reproducible $\hat{Var}[edge(10,i)]$ versus $\hat{E}[edge(10,i)]$ plots result from the simulation on the *heart-h* structure. With the $\hat{Var}[edge(10,i)]$ versus $\hat{E}[edge(10,i)]$ plots being comparable for the actual *heart-h* dataset plot (Figure 5.6) and the simulated *heart-h* dataset (Figure 5.16). Checking those observations with negative slope returns the original observations in the *heart-h* dataset which were flagged with $edgeflag = 1$. Although the negative trends are exaggerated, this demonstrates the point that observations which are classified according to an inverse rule to the main classification rule result in the negative trends when $\hat{Var}[edge(10,i)]$ is plotted against $\hat{E}[edge(10,i)]$.

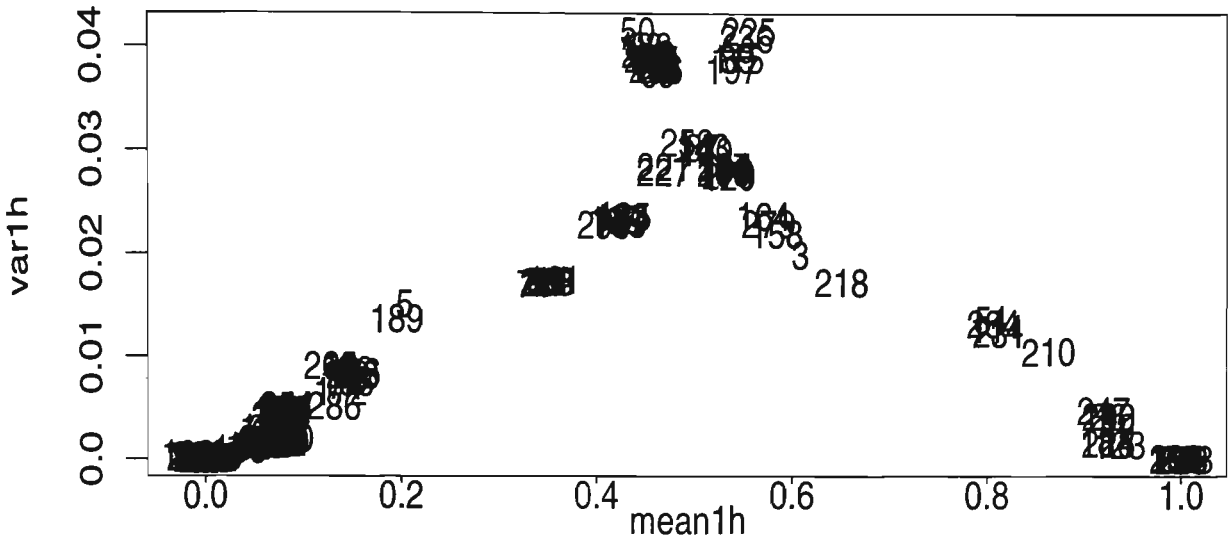

Figure 5.16: Plot of $\hat{Var}[edge(10,i)]$ vs $\hat{E}[edge(10,i)]$ on simulated model on 294 observations with pre-assigned clusters as per *hearth* data .

5.7 Theoretical Derivations for Inverse Models

Given that the same signature features appear in all $\hat{Var}[edge(10,i)]$ versus $\hat{E}[edge(10,i)]$ plots drawn in this Chapter, can it be proven theoretically that such features are expected when inverse sub-contexts are present within a dataset? The repeatable signature behaviour of symmetry about the line $\hat{E}[edge(10,i)] = 0.5$ is certainly worth investigating theoretically. This Section develops theoretical expressions for mean and variance of the edge and explores the assumptions regarding inverse contexts made in the empirical studies of Sections 5.2.1 - 5.2.6.

5.7.1 Theoretical expressions for $\hat{E}[edge(m,i)]$ and $\hat{Var}[edge(m,i)]$

Recall in Section 2.5, the edge of the i-th observation can be expressed in the form:

$$edge(m,i) = \frac{\sum_{j=1}^{m} a_j I_j[h_j(\mathbf{x}_i) \neq y_i]}{\sum_{j=1}^{m} a_j} \tag{5.7.1}$$

Now, $I_j(i) = I[h_j(\mathbf{x}_i) \neq y_i]$ and is equal to 0 if the j-th hypothesis is correct for the i-th observation and 1 otherwise. a_j is equal to the weighted vote of the j-hypothesis, which for *AdaBoost.M1* is equal to $\log(\frac{1-\epsilon_j}{\epsilon_j})$, where ϵ_j is the weighted error of the j-th hypothesis.

In matrix notation, the vector of edge statistics after $1, 2, \ldots, m$ boosting iterations can be expressed as $\mathbf{CZ}_i$, where $\mathbf{Z}_i = (I_1(i), I_2(i), \ldots, I_m(i))$. As the total votes for each iteration sum to 1, $\mathbf{C}$ is a lower triangular matrix which satisfies $\mathbf{C1} = \mathbf{1}$, where $\mathbf{1}$ is a column vector of ones.

In the case of equal voting weights, the edge of the i-th observation after m iterations is given by:

$$edge(m,i) = \frac{1}{m}\sum_{j=1}^{m} I_j(\mathbf{x}_i) \tag{5.7.2}$$

And for equal hypothesis vote weights, $edge(m,i)$ may be represented as the m-th row of the following vector:

$$\mathbf{e}_i = \begin{pmatrix} I_1(i) \\ \frac{1}{2}I_1(i) + \frac{1}{2}I_2(i) \\ \frac{1}{3}I_1(i) + \frac{1}{3}I_2(i) + \frac{1}{3}I_3(i) \\ \vdots \\ \frac{1}{m}\sum_{j=1}^{m} I_j(i) \end{pmatrix}$$

Or the m-th row of $\mathbf{CZ}_i$ where,

$$\mathbf{C} = \begin{pmatrix} 1 & 0 & 0 & 0 & \cdots & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 & 0 & \cdots & 0 & 0 \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{3} & 0 & \cdots & 0 & 0 \\ \vdots & \cdots & \cdots & \cdots & \cdots & \cdots & \vdots \\ \frac{1}{m} & \frac{1}{m} & \frac{1}{m} & \frac{1}{m} & \cdots & \frac{1}{m} & \frac{1}{m} \end{pmatrix}$$

And,

$$\mathbf{Z}_i = \begin{pmatrix} I_1(i) \\ I_2(i) \\ I_3(i) \\ \vdots \\ I_m(i) \end{pmatrix}$$

Empirically it is observed that vote weights for the m hypotheses do not stray significantly from uniformity. We seek expressions for the values of the average and variance of the edge for the i-th observation taken over m iterations. Denote these quantities by $E_m[edge(m,i)]$ and $Var_m[edge(m,i)]$ respectively. The expectations are taken over the m iterations for each observation as opposed to expectations being taken over the N observations for the m-th iteration as derived in Chapter 3.

The mean edge of the i-th observation over the m iterations is given by:

$$\begin{aligned} E_m[edge(m,i)] &= E_m[\mathbf{CZ}_i] \\ &= \frac{1}{m}\mathbf{1}^{\mathbf{T}}\mathbf{CZ}_i \end{aligned}$$

And the variance is:

$$\begin{aligned} Var_m[edge(m,i)] &= Var_m[\mathbf{CZ}_i] \\ &= \frac{1}{m}\mathbf{Z}_i^{\mathbf{T}}\mathbf{C}^{\mathbf{T}}\mathbf{CZ}_i - (\frac{1}{m}\mathbf{1}^{\mathbf{T}}\mathbf{CZ}_i)^2 \\ &= \mathbf{Z}_i^{\mathbf{T}}\mathbf{C}^{\mathbf{T}}(\frac{1}{m}\mathbf{I} - \frac{1}{m^2}\mathbf{J})\mathbf{CZ}_i \end{aligned}$$

Where $\mathbf{I}$ is an $m \times m$ identity matrix and $\mathbf{J}$ is an $m \times m$ matrix of ones.

It is possible to determine bounds for $Var_m[\mathbf{CZ}_i]$ as a function of $E_m[\mathbf{CZ}_i]$ which explain the nature of the lower rim of the bell shaped curve noted in Figure 5.17. We see that $Var_m[\mathbf{CZ}_i] \geq \frac{1}{(m-1)}\left(E_m[\mathbf{CZ}_i] - I_1(i)\right)^2$ as is proven below:

Let us re-express $E_m[\mathbf{CZ}_i]$ as $\boldsymbol{\mu}^T\mathbf{Z}_i$ where $\boldsymbol{\mu} = \frac{1}{m}\mathbf{C}^T\mathbf{1}$. Recall $I_j(i)$ is the misclassification indicator for observation i at the j-th iteration. Now:

$$\begin{aligned} Var_m[\mathbf{CZ}_i] &= \frac{1}{m}\mathbf{Z}_i^T\mathbf{C}^T\mathbf{CZ}_i - \left(\frac{1}{m}\mathbf{1}^T\mathbf{CZ}_i\right)^2 \\ &= \mathbf{Z}_i^T\boldsymbol{\Sigma}\mathbf{Z}_i \end{aligned}$$

where: $\boldsymbol{\Sigma} = \mathbf{C}^T\left(\frac{1}{m}\mathbf{I} - \frac{1}{m^2}\mathbf{J}\right)\mathbf{C}$

Since $\mathbf{C1} = \mathbf{1}$, it follows that:

$$\begin{aligned} \boldsymbol{\Sigma}\mathbf{1} &= \mathbf{C}^T\left(\frac{1}{m}\mathbf{I} - \frac{1}{m^2}\mathbf{J}\right)\mathbf{1} \\ &= \mathbf{C}^T\left(\frac{1}{m}\mathbf{1} - \frac{1}{m}\mathbf{1}\right) \\ &= \mathbf{0} \end{aligned}$$

Therefore $Var_m[\mathbf{CZ}_i]$ may be re-expressed as:

$$\begin{aligned} Var_m[\mathbf{CZ}_i] &= (\mathbf{Z}_i - I_1(i)\mathbf{1})^T\boldsymbol{\Sigma}(\mathbf{Z}_i - I_1(i)\mathbf{1}) \\ &= \mathbf{Y}_i^T\boldsymbol{\Sigma}_*\mathbf{Y}_i \end{aligned}$$

where $\mathbf{Y}_i = (I_2(i) - I_1(i), \ldots, I_m(i) - I_1(i))^T$ and the matrix $\mathbf{\Sigma}_*$ is obtained by omitting the first row and column of $\mathbf{\Sigma}$. A standard multivariate result gives:

$$\mathbf{Y}_i^T \mathbf{\Sigma}_* \mathbf{Y}_i \geq \frac{(\boldsymbol{\mu}_*^T \mathbf{Y}_i)^2}{(\boldsymbol{\mu}_*^T \mathbf{\Sigma}_*^{-1} \boldsymbol{\mu}_*)}$$

But

$$\begin{aligned} \boldsymbol{\mu}_*^T \mathbf{\Sigma}_*^{-1} \boldsymbol{\mu}_* &= \frac{1}{m} \mathbf{1}_{m-1}^T \mathbf{C}_* \mathbf{C}_*^{-1} (\mathbf{I}_{m-1} + \mathbf{J}_{m-1}) (\mathbf{C}_*^T)^{-1} \mathbf{C}_*^T \mathbf{1}_{m-1} \\ &= \frac{1}{m} \mathbf{1}^T (\mathbf{I}_{m-1} + \mathbf{J}_{m-1}) \mathbf{1} \\ &= m - 1 \end{aligned}$$

Where $\mathbf{C}_*$ is the matrix obtained by omitting the first row and column of $\mathbf{C}$, and $\boldsymbol{\mu}_*$ results from omitting the first row of the vector $\boldsymbol{\mu}$.

Now,

$$\begin{aligned} \boldsymbol{\mu}_*^T \mathbf{Y}_i &= \boldsymbol{\mu} (\mathbf{Z}_i - I_1(i)) \mathbf{1} \\ &= E_m[\mathbf{C}\mathbf{Z}_i] - I_1(i) \end{aligned}$$

Therefore:

$$Var_m[\mathbf{C}\mathbf{Z}_i] \geq \frac{1}{(m-1)} \left(E_m[\mathbf{C}\mathbf{Z}_i] - I_1(i)\right)^2 \tag{5.7.3}$$

For example if $m = 10$ and $E_m[\mathbf{C}\mathbf{Z}_i] = 0.5$ are substituted into equation (5.7.3), we obtain a lower bound for $Var_m[\mathbf{C}\mathbf{Z}_i]$ of 0.0278 which closely matches the values seen at the mode of the bell shaped curves observed in Figures 5.17 and 5.19. This bound is consistent when other values of $E_m[\mathbf{C}\mathbf{Z}_i]$ are substituted into the derived bound and compared to values from the theoretical mean/variance plots.

5.7.2 Inverse Observations

Consider two observations with $\mathbf{x}_k = \mathbf{x}_i$ but $y_k = 1 - y_i$ i.e. opposing class values. For each of the m classifiers the predicted class for the i-th and k-th observations will be the same, and therefore $\forall j = 1 \ldots m \; I_j(k) = 1 - I_j(i)$. Or, in vector notation: $\mathbf{Z}_k = \mathbf{1} - \mathbf{Z}_i$. We call the i-th and k-th observations *inverse observations* and from the results derived below in Equations (5.7.4) and (5.7.5) we see that they have inverse means and equal variances of their edges. More generally, this is also the case if $\mathbf{x}_i$ and $\mathbf{x}_k$ differ only in redundant input variables. This result has important practical implications which will be revealed in Chapter 6.

For any vector $\mathbf{Z}_i$. it follows that:

$$\begin{aligned} E_m[\mathbf{C}(\mathbf{1} - \mathbf{Z}_i)] &= \tfrac{1}{m}\mathbf{1}^{\mathbf{T}}\mathbf{C}(\mathbf{1} - \mathbf{Z}_i) \\ &= \tfrac{1}{m}\mathbf{1}^{\mathbf{T}}\mathbf{C}\mathbf{1} - \tfrac{1}{m}\mathbf{1}^{\mathbf{T}}\mathbf{C}\mathbf{Z}_i \\ &= 1 - E_m(\mathbf{C}\mathbf{Z}_i) \end{aligned} \tag{5.7.4}$$

And

$$\begin{aligned} Var_m[\mathbf{C}(\mathbf{1} - \mathbf{Z}_i)] &= (\mathbf{1} - \mathbf{Z}_i)^{\mathbf{T}}\mathbf{C}^{\mathbf{T}}(\tfrac{1}{m}\mathbf{I} - \tfrac{1}{m^2}\mathbf{J})\mathbf{C}(\mathbf{1} - \mathbf{Z}_i) \\ &= \mathbf{1}^{\mathbf{T}}\mathbf{C}^{\mathbf{T}}(\tfrac{1}{m}\mathbf{I} - \tfrac{1}{m^2}\mathbf{J})\mathbf{C}\mathbf{1} - \mathbf{1}^{\mathbf{T}}\mathbf{C}^{\mathbf{T}}(\tfrac{1}{m}\mathbf{I} - \tfrac{1}{m^2}\mathbf{J})\mathbf{C}\mathbf{Z}_i \\ &- \mathbf{Z}_i^{\mathbf{T}}\mathbf{C}^{\mathbf{T}}(\tfrac{1}{m}\mathbf{I} - \tfrac{1}{m^2}\mathbf{J})\mathbf{C}\mathbf{1} + \mathbf{Z}_i^{\mathbf{T}}\mathbf{C}^{\mathbf{T}}(\tfrac{1}{m}\mathbf{I} - \tfrac{1}{m^2}\mathbf{J})\mathbf{C}\mathbf{Z}_i \\ &= \mathbf{1}^{\mathbf{T}}(\tfrac{1}{m}\mathbf{1} - \tfrac{1}{m}\mathbf{1})\mathbf{1} - \mathbf{1}^{\mathbf{T}}(\tfrac{1}{m}\mathbf{I} - \tfrac{1}{m^2}\mathbf{J})\mathbf{C}\mathbf{Z}_i - \mathbf{Z}_i^{\mathbf{T}}\mathbf{C}^{\mathbf{T}}(\mathbf{0}) \\ &+ \mathbf{Z}_i^{\mathbf{T}}\mathbf{C}^{\mathbf{T}}(\tfrac{1}{m}\mathbf{I} - \tfrac{1}{m^2}\mathbf{J})\mathbf{C}\mathbf{Z}_i \\ &= \mathbf{Z}_i^{T}\mathbf{C}^{\mathbf{T}}(\tfrac{1}{m}\mathbf{I} - \tfrac{1}{m^2}\mathbf{J})\mathbf{C}\mathbf{Z}_i \\ &= Var_m(\mathbf{C}\mathbf{Z}_i) \end{aligned} \tag{5.7.5}$$

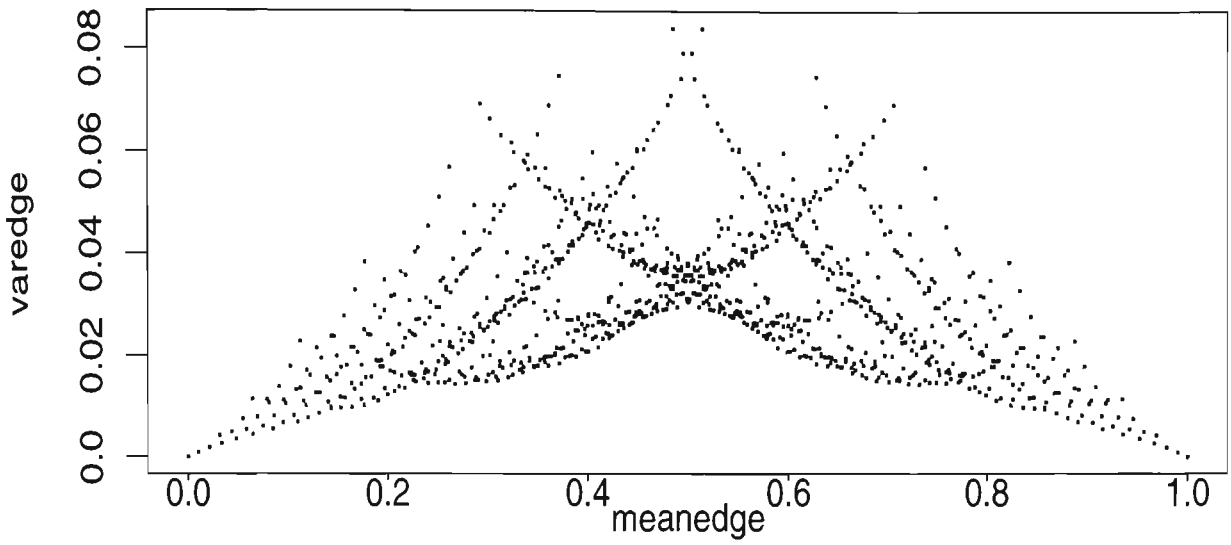

Figure 5.17: Plot of $\hat{Var}[edge(10,i)]$ vs. $\hat{E}[edge(10,i)]$: 2^{10} possibilities input space

As each component of $\mathbf{Z}_i$ has two possible values, there are 2^m possible outcomes of $\mathbf{Z}_i$. The edge variance can be plotted against the mean edge over the complete state space, in order to help interpret the variance/mean plots arising from actual boosting experiments on UCI and 'real life' datasets. This is drawn as Figure 5.17. For the above results involving $\mathbf{1} - \mathbf{Z}_i$, it can be seen that the plots of all possible combinations of mean and variance is symmetric about a mean of 0.5. Figure 5.17 is not dissimilar to Figures 5.2, 5.6 and 5.10 except for the latter Figures there are certain regions in mean/variance space which are not occupied. In Figures 5.2,5.6 and 5.10, we see a dense cluster in the lower left hand corner of the mean/variance plot. We also observe mirror imaging in the

mean=0.5 line, but only below the variance=0.03 line. We seem to be missing points with a negative relationship between mean and variance with a low mean. The mean and variance plots from the datasets tested seem to be hugging the lower border of a bell-shaped curve evident in Figure 5.17 and plots from empirical studies on actual datasets are not displaying as many points in the higher sections of the plot drawn for all 2^m possibilities. However, if this higher section of the $\hat{Var}[edge(10,i)]$ versus $\hat{E}[edge(10,i)]$ plot was filled, boosting would terminate due to the weighted error of the hypotheses becoming larger than 0.5. When mirror images are occurring they are often clustered around $\hat{E}[edge(10,i)] = 0.5$, $\hat{Var}[edge(10,i)] = 0.03$, the centre of the bell shape. Upper bounds on the variance of the edge were derived in Equation 5.7.3, which explains this phenomenon.

An interesting investigation would be to plot $\hat{Var}[edge(10,i)]$ and $\hat{E}[edge(10,i)]$ for observations where $I_1(\mathbf{x}_i) = 1$ i.e. those observations which were initially misclassified. This may shed some light on the effect of using unweighted means as opposed to weighted means since we will be able to detect where all initially misclassified observations fall in regards to the total possible state space. This plot is drawn as Figure 5.18 and we can see that only the right hand side of the plot in Figure 5.17 results. This points to the flaw in the previous argument of segregating <u>all</u> observations exhibiting a negative relationship between $\hat{Var}[edge(10,i)]$ and $\hat{E}[edge(10,i)]$. By doing this we are merely isolating all initially misclassified

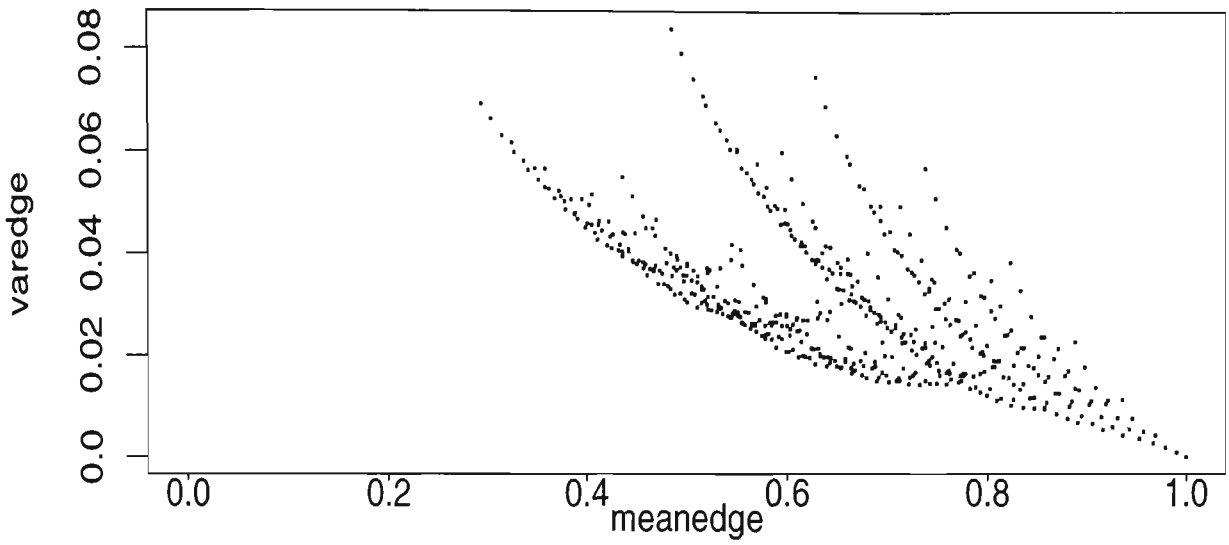

Figure 5.18: Typical plot of $\hat{Var}[edge(10,i)]$ vs. $\hat{E}[edge(10,i)]$: initially misclassified observations

observations and losing vital information about the nature of the data. This explains the dramatic decreases in generalisation error noted in Sections 5.2.1-5.2.6 and although the power of the results in Sections 5.2.1-5.2.6 is now diminished, we have gained knowledge of the flaws in the inverse model conjecture and subsequent edge analysis. In knowing this we seek to combine the logic behind the inverse model conjecture and the results derived in Equations (5.7.4) and (5.7.5) to detect *inverse observations* based on sound theoretical derivations.

For comparison, if we plot the mean and variance state space for observations misclassified at iteration 10, a consistent spread appears unlike the $\hat{Var}[edge(10,i)]$ versus $\hat{E}[edge(10,i)]$ plot for observations initially misclassified. Refer to Figure 5.19 for the $\hat{Var}[edge(10,i)]$ versus $\hat{E}[edge(10,i)]$ plot for observations misclassified at iteration 10. We see no distinct location as in Figure 5.18, rather an

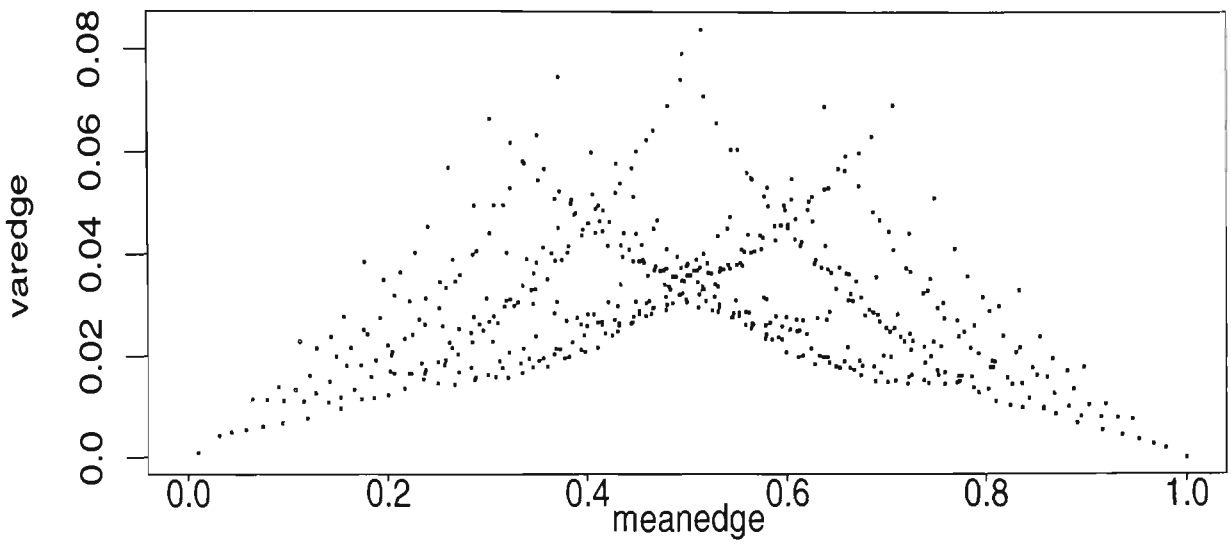

Figure 5.19: Typical plot of $\hat{Var}[edge(10,i)]$ vs. $\hat{E}[edge(10,i)]$: observations misclassified at iteration 10.

'equalisation' with points falling in a similar shape to Figure 5.17 but less dense. This is a possible indicator that we should be using weighted mean and variance as the misclassifications in later iterations have significantly less influence on the overall edge mean and variance. This notion is investigated in this Chapter as equation (5.7.9).

The plots of mean/variance space shown as Figures 5.17, 5.18 and 5.19 may be redrawn for weighted voting. Since there are infinite possibilities for the weight vector, the weights obtained empirically for the *colic* dataset are applied. Instead of each of the ten hypotheses having equal vote weights of $\frac{1}{10}$, the vote weights vector is given by:

$$\mathbf{c}_{colic} = (0.1735, 0.1857, 0.0825, 0.1170, 0.0747, 0.0980, 0.0728, 0.0845, 0.0814, 0.0299)^T$$

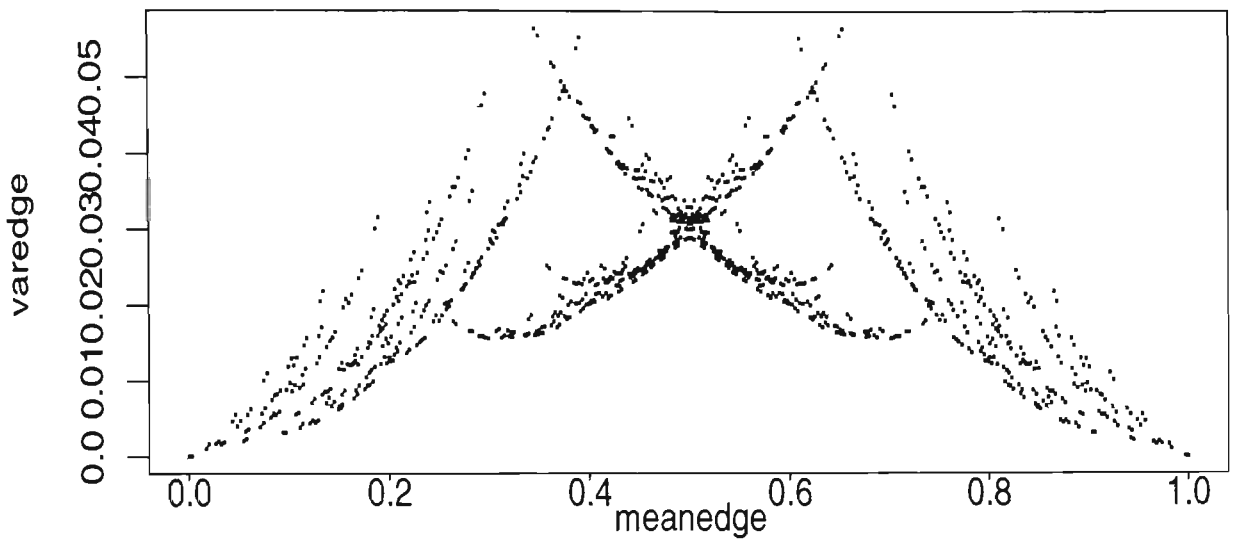

Figure 5.20: Typical plot of $\hat{Var}[edge(10,i)]$ vs. $\hat{E}[edge(10,i)]$: weighted voting according to $\mathbf{C}_{colic}$

The resulting mean/variance plots are given as Figures 5.20, 5.21 and 5.22. The overall shape and space occupation of the plots is unchanged from that observed for the unweighted voting case. We see minor perturbations in the shape of the dominant bell and initially misclassified observations still form a negative relationship when edge variance is plotted against mean edge. In Figure 5.22, where the space is plotted for observations misclassified at iteration 10, the plot is very similar to that plotted for the unweighted case where a more even distribution of points is plotted. Hence weighted voting and edge formation has no impact on the key features of a theoretical mean/variance plot and methodologies suggested from the features of such plots will be robust in the more general case of weighted voting.

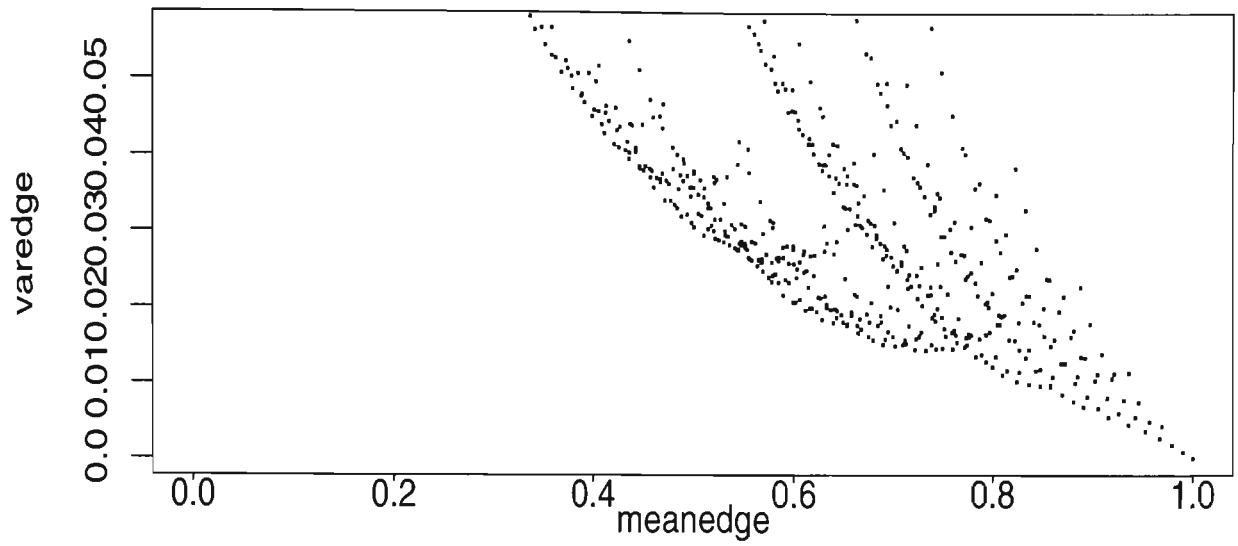

Figure 5.21: Typical plot of $\hat{Var}[edge(10,i)]$ vs. $\hat{E}[edge(10,i)]$: observations misclassified at iteration 1, weighted voting according to $\mathbf{c}_{colic}$.

It seems intuitive that the initial edge value ($edge(1,i)$) will have a strong influence on any edge statistics calculated across m iterations. This influence may mask behaviour and disadvantage later misclassifications. It now seems appropriate to apply a weighted mean and variance to the edge diagnostic procedure. More generally, if the mean of the edge is calculated as a weighted mean as opposed to an unweighted mean, we may adjust the weighting of each iteration so as to reduce the impact of earlier iterations and give later iterations a greater opportunity to influence edge statistics. Weighted mean and variance expressions are derived below in Equations (5.7.6) and (5.7.7) but the relationship between mean and variance for perfect inverses does not change if the means and variances are weighted. So even if we seek to lessen the impact of the initial misclassification on the final mean and variance calculations, we will still observe the perfect

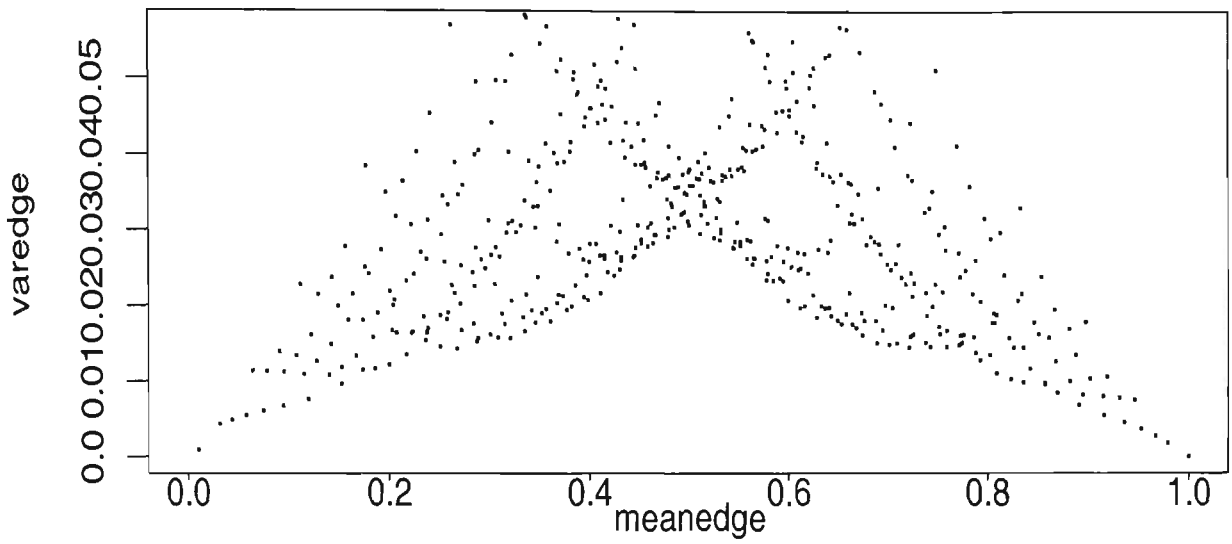

Figure 5.22: Typical plot of $\hat{Var}[edge(10,i)]$ vs. $\hat{E}[edge(10,i)]$: observations misclassified at iteration 10, weighted voting according to $\mathbf{c}_{colic}$.

symmetry in the line $\hat{E}[edge(10,i)] = 0.5$ for inverse observations, where means are inverse and variances are equal.

Letting the weights for $edge(1,i), edge(2,i), \ldots, edge(m,i)$ be $w_1, w_2, \ldots, w_m$, then the weighted mean and variance of the edge for the i-th observation are given by:

$$E_m[edge(m,i)] = E_m[\mathbf{CZ}_i] = \mathbf{w}^{\mathbf{T}}\mathbf{CZ}_i \tag{5.7.6}$$

And

$$\begin{aligned} Var_m[edge(m,i)] &= Var_m[\mathbf{CZ}_i] \\ &= \mathbf{Z}_i\mathbf{T}\mathbf{C}^{\mathbf{T}}\mathbf{W}\mathbf{C}\mathbf{Z}_i - (\mathbf{w}^{\mathbf{T}}\mathbf{C}\mathbf{Z}_i)^2 \\ &= \mathbf{Z}_i^{\mathbf{T}}\mathbf{C}^{\mathbf{T}}(\mathbf{W} - \mathbf{w}\mathbf{w}^{\mathbf{T}})\mathbf{C}\mathbf{Z}_i \end{aligned} \tag{5.7.7}$$

where $\mathbf{w} = (w_1, w_2, \ldots, w_m)^T$ and $\mathbf{W} = diag(\mathbf{w})$. The fact that $\sum w_j = 1$ implies that:

$$\mathbf{w^T 1} = 1$$

And

$$\begin{aligned} (\mathbf{W} - \mathbf{w}\mathbf{w^T})\mathbf{1} &= \mathbf{w} - \mathbf{w} \\ &= \mathbf{0} \end{aligned}$$

For any vector $\mathbf{Z}_i$, it follows that:

$$\begin{aligned} E_m[\mathbf{C}(\mathbf{1} - \mathbf{Z}_i)] &= \mathbf{w^T C}(\mathbf{1} - \mathbf{Z}_i) \\ &= \mathbf{w^T C 1} - \mathbf{w^T C Z}_i \\ &= 1 - \mathbf{w^T C Z}_i \\ &= 1 - E_m[\mathbf{C Z}_i] \end{aligned} \tag{5.7.8}$$

And:

$$\begin{aligned} Var_m[\mathbf{C}(1 - \mathbf{Z}_i)] &= (\mathbf{1} - \mathbf{Z}_i)^T \mathbf{C}^T (\mathbf{W} - \mathbf{w}\mathbf{w}^T)(\tfrac{1}{m}\mathbf{I} - \tfrac{1}{m^2}\mathbf{J})\mathbf{C}(\mathbf{1} - \mathbf{Z}_i) \\ &= \mathbf{1^T}(\mathbf{W} - \mathbf{w}\mathbf{w^T})\mathbf{1} - 2\mathbf{1^T}(\mathbf{W} - \mathbf{w}\mathbf{w^T})\mathbf{C Z}_i \\ &+ \mathbf{Z}_i^\mathbf{T}\mathbf{C^T}(\mathbf{W} - \mathbf{w}\mathbf{w^T})\mathbf{C Z}_i \\ &= \mathbf{Z}_i^\mathbf{T}\mathbf{C^T}(\mathbf{W} - \mathbf{w}\mathbf{w^T})\mathbf{C Z}_i \\ &= Var_m[\mathbf{C Z}_i] \end{aligned} \tag{5.7.9}$$

Again, as for the unweighted case, *inverse observations* will have inverse weighted means and equal weighted variances. The symmetry properties proven in Equations (5.7.4) and (5.7.5) will hold irrespective of the weighting given to each edge

value in calculating the means and variances across iterations. Equations (5.7.4), (5.7.5), (5.7.8) and (5.7.9) demonstrate a key factor in being able to detect *inverse observations*. (i.e. observations where $\mathbf{x}_k = \mathbf{x}_i$ but $y_k = 1 - y_i$). The practical implications of these relationships are demonstrated in Chapter 6, whereby the symmetry observed in the $\hat{Var}[edge(10, i)]$ versus $\hat{E}[edge(10, i)]$ plots is exploited to isolate the inverse observations which cause conflict when a classifier attempts to reconcile opposing class values for equivalent input values.

5.7.3 Refining the Inverse Model Hypothesis

In Section 5.7.1, we have seen that plots of the mean edge versus the edge variance will always detect the initially misclassified observations if we isolate all observations with a negative relationship between mean and variance of their edges. This may seem intuitive since the initial edge will be 1 for all initially misclassified observations and this initial misclassification will follow through impacting all future edge calculations. However, we now know from the preceding theory that observations with opposing class labels for equal input values will fall as mirror images about the mean =0.5 line with equal variances. The above theory also confirms that there may be more clusters than simply the negative and positive relationships between $\hat{Var}[edge(10, i)]$ and $\hat{E}[edge(10, i)]$ as shown in Figure 5.23.

The dense cluster captures the majority of observations, the majority of which are correctly classified in all iterations. A classifier built on this cluster will be simpler and more robust than a classifier built on the entire dataset. It has been

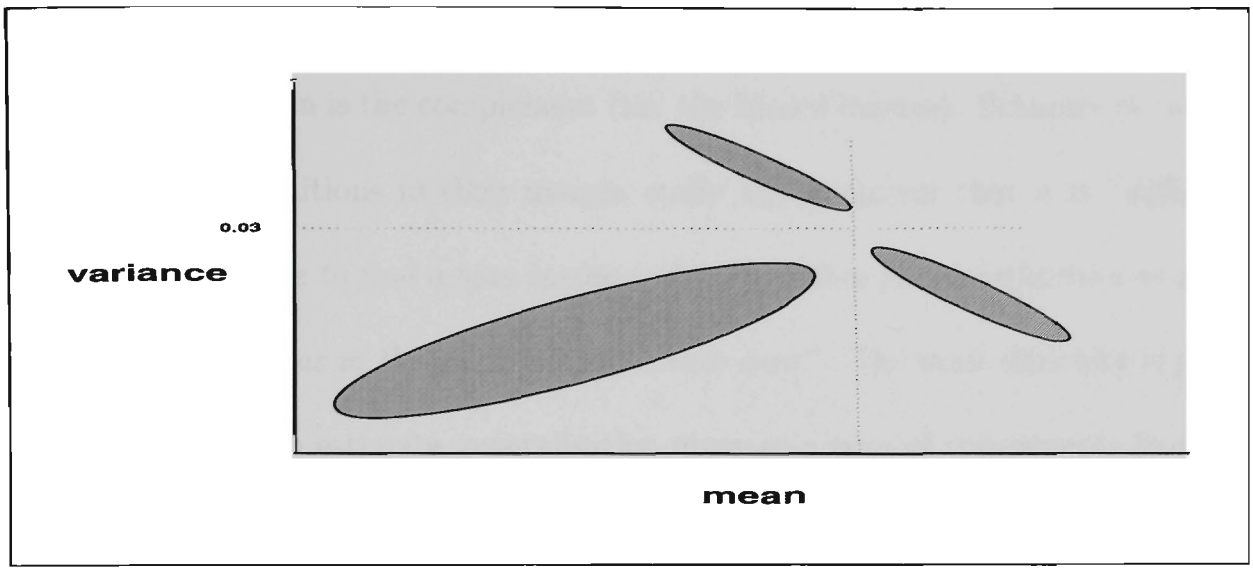

Figure 5.23: Typical plot of $\hat{Var}[edge(10,i)]$ vs. $\hat{E}[edge(10,i)]$

proven that observations falling below the $\hat{Var}[edge(10,i)] = 0.03$ line with *mirrored* observations in the line $\hat{E}[edge(10,i)] = 0.5$ should be examined as they will have opposing class values for equivalent input values. Modelling cannot properly account for such observations as the classifier is forced to make a decision between two opposing class values. As observations weights are adjusted during the boosting process, the classifier will change its decision on class based on the class with the majority weight. We therefore expect such inverse observations to alternate between the left and right hand sides of the $\hat{E}[edge(10,i)] = 0.5$ line as boosting proceeds. Detection and removal of the minority class in this situation will lead to significantly more robust classifiers on the majority cluster. Of course, the measure of generalisation error must be redefined as removal of observations leads to optimistic, misrepresentative test set errors. Breiman [5] defines bias and variance as partitioned quantities. His definitions are akin to the notion of

inverse models identified in this Thesis whereby one partition includes the $\mathbf{x}_i$'s for which the average classifier is unbiased for the optimal Bayes prediction rule. The second partition is the complement (i.e. the biased inverse). Schapire et. al. [42] use these definitions in their margin study and comment that it is "*difficult or even impossible to find a bias-variance decomposition for classification as natural and satisfying as in the quadratic regression case.*". The same difficulty is present when trying to estimate generalisation error as a sum of components from each partition of the dataset.

Chapter 6 applies this new discovery to the *colic*, *heart-h* and *BHP* datasets with encouraging results.

Chapter 6

Inverse Detection and Partitioned Generalisation Error

In this chapter we explore the practical implications of the theoretical expressions derived for $E_m[edge(m,i)]$ and $Var_m[edge(m,i)]$ in Chapter 5. In Section 5.7.1 we considered two observations with $\mathbf{x}_k = \mathbf{x}_i$ but $y_k = 1 - y_i$ i.e. opposing class values. For each of the m classifiers the predicted class for the i-th and k-th observations will be the same, and therefore $\forall j = 1 \ldots m$, $I_j(k) = 1 - I_j(i)$. We called the i-th and k-th observations *inverse observations*. In Chapter 5 it was proven that observations with inverse means and equal variances of their edges are *inverse observations*. More generally, this is also the case if $\mathbf{x}_i$ and $\mathbf{x}_k$ differ only in redundant input variables.

This forces the classifier to make difficult decisions as is it torn between two opposing forces. Depending on the weights of each observation, the classifier will swing between its class prediction. Incorrectly labelled observations will appear on the right hand side of the $\hat{E}[edge(m,i)] = 0.5$ line but will alternate depending on weighting as boosting proceeds. Figures 5.2, 5.6, 5.12 and 5.13 appeared to

have this feature.

It is now proposed to use this knowledge to improve classifier accuracy and simplicity. Observations with equal edge variance exhibiting a $\hat{E}[edge(10,i)]/1 - \hat{E}[edge(10,i)]$ relationship are said to be *mirrored* and are key in deteriorated classifier performance. The presence of a significant number of such observations results in unnecessarily complex classifiers due to the conflict of fitting inverse observations whose relative weightings change at each boosting iteration. By dealing with such observations, the resulting classifier is able to concentrate on the 'pure' nature of the underlying model which is certain to be simpler in structure than that over-fitted when inverse observations are present.

6.1 New results with mirroring

Further examination of the $\hat{Var}[edge(10,i)]$ versus $\hat{E}[edge(10,i)]$ plots for the *colic*, *heart-h* and *BHP* dataset shows a number of observations with identical input $\mathbf{x}_i$'s and opposite class labels. This is detected visually from the $\hat{Var}[edge(10,i)]$ versus $\hat{E}[edge(10,i)]$ plots via observations having identical variances and an $\hat{E}[edge(10,i)]$ / $1 - \hat{E}[edge(10,i)]$ relationship as formulated in Section 5.7.1, and checked via a basic sorting algorithm. In all cases there was a majority class label, given the identical input values. Do we simply remove the 'spurious' observations with apparent class mislabels or do we change the value of their class label? The answer to this is unclear but we must consider the change of weights which would result in the dataset for the particular combination of

input values. For example, a group of 3 observations with equal edge variance having one observation mislabelled will have greater impact than a group of 8 observations with one mislabelled. It is vital to ensure any resulting dataset is still representative of the distribution across the input values. In this study, we chose to simply remove the mislabelled minority and train and test on this altered dataset. This differs from Sections 5.2.1 - 5.2.6 as we are not simply siphoning off the observations with negative relationship between mean and variance, rather carefully examining those with exact inverse relationships. This procedure has a theoretical basis as developed in Section 5.7.1. By removing any data which we may consider mislabelled, the generalisation error estimated on the new pruned dataset should certainly improve, along with the benefit of achieving a simpler classification model less prone to overfitting. However, the generalisation error as measured on this reduced dataset is misleading and optimistic. But we now have the opportunity of devising a more informed generalisation error estimate: one which combines two components. Such an estimate will be able to quantify both the proportion of mislabelled data and the amount of error attributed to unmodellable noise and model inadequacies.

6.1.1 Generalisation Error Calculations

Using the *mirroring* property to detect and remove minority mislabelled observations will result in a misrepresentative generalisation error. The observations removed from the dataset are responsible for overfitting and classification conflict when the classifier is torn between opposing class labels given identical input

values. We would expect generalisation error based on the reduced dataset to be smaller than that obtained when training on the entire dataset and we expect future unseen observations to have the same distribution and same level of mislabelling as was seen in the training set. The amount of noise which is present in incoming unseen data is unmeasurable. At best we can only assume that unseen data will follow the same distribution as the training data and potential changes in distribution are uncalculable at the time of testing.

Assuming that unseen data will be distributed in a similar fashion to the training data and contain the same level of mislabels and noise, we may use this information as a means of segregating generalisation error into a component pertaining to observation noise and a component for modelling noise. Combining the two components does not significantly alter the magnitude of the existing generalisation error based on the entire dataset but we are gaining valuable insight into the relative break-up of errors present when faced with an existing noisy dataset.

Because it is proposed to use the simpler model trained on a subset of the original dataset, generalisation error must be re-defined.

Assume the proportion of observations removed due to *mirroring* is equal to p. $(0 < p < 1)$. Let e_1 represent the estimated generalisation error for the simple classifier trained on the new training data, estimated via 10-fold cross-validation. Now, since the remainder of the data is assumed to be mislabelled, the generalisation error for this section of the data will at worst be estimated at 100%.

Denoting the new overall generalisation error estimate as e_{mirror}, we have:

$$e_{mirror} \approx (1 - p) * e_1 + p \tag{6.1.1}$$

Although we see in later empirical work that the magnitude of this generalisation error is unchanged from published results for boosting [34], we are gaining more information as the generalisation error (e_{mirror}) is now reported in terms of an error due to mislabelling and error due to noise and classifier robustness. This places the classification tools in a better light and also gives an indication of how much influence the mislabelling has on the overall error expected on unseen data. In knowing this, the analyst can decide whether or not the mislabelling is worth remedying or seek to measure a key input variable which has been not measured or omitted from the dataset. Including the *mirrored* observations in a training dataset results not only in a more pessimistic estimate of generalisation error as the generalisation error is forced to encompass these observations, but causes the classifier to be overly complex. It makes sense to segregate the generalisation error when such observations are detectable using the edge diagnostics and *mirroring* introduced in this Thesis.

Now, the question of how to appropriately split the reduced dataset in the cross validation phase. Two methods may be applied in order to obtain an estimate of generalisation error on the reduced dataset. These are tabulated in Table 6.1 We see the two methods differing only in the order of the first two steps. In this Thesis, Method B is applied when determining the presence or otherwise of inverse observations and estimating generalisation error. Method B was chosen because

Table 6.1: Possible methods for estimating generalisation error on reduced datasets.

Step	**Method A**	**Method B**
1	Remove fold	Assess and remove *mirror* points
2	Assess and remove *mirror* points	Remove fold
3	Train on reduced set	Train on reduced set
4	Test	Test

it has advantages in terms of computational intensity. Further investigation into the differences between Method A and Method B could be the subject of future research.

Denote the generalisation error estimate using Methods A and B by e_A and e_B respectively, and the proportion of mirrored observations by p_{mirror} . Now,

$$\begin{aligned} e_A &= (1 - p_{mirror}) \times \text{P(misclass using A} \mid \text{not a mirror point using B)} \\ &+ p_{mirror} \times \text{P(misclass using A} \mid \text{mirror point in B)} \\ &\approx (1 - p_{mirror}) e_B + p_{mirror} \times 1 \end{aligned}$$

An empirical study is undertaken on a selection of UCI [2] datasets to test the process outlined in Section 5.7.3. For all datasets, the term 'boosting' refers to the *AdaBoost.M1* algorithm presented in Chapter 2 (Table 2.1). Unless otherwise described for a particular dataset, 10 boosting iterations were applied using *c*4.5 with default options as the base learner.

6.1.2 Results on *colic* data

Recall the plot of $\hat{Var}[edge(10, i)]$ versus $\hat{E}[edge(10, i)]$ for the *colic* data plotted as Figure 5.2 in Chapter 5. Analysis in Section 5.2.1 involved the partitioning

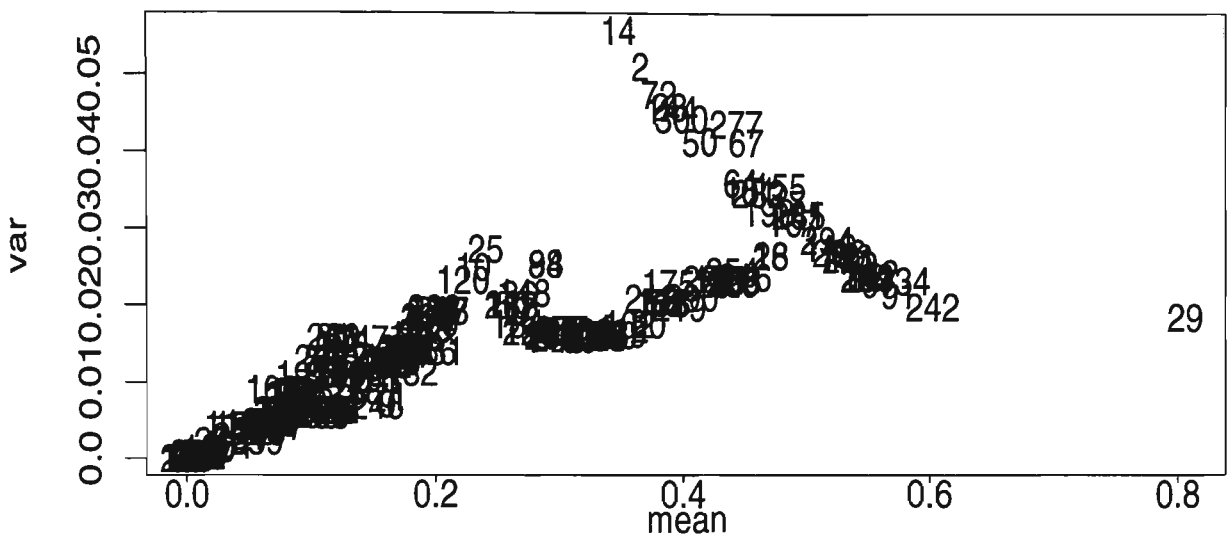

Figure 6.1: Plot of $\hat{Var}[edge(10,i)]$ vs. $\hat{E}[edge(10,i)]$: *colic* data.

of the data according to whether or not an observation fell into the negative cluster in Figure 5.2. We have now seen that this method will only detect observations which were initially misclassified. The theory developed in Section 5.7 showed that an important relationship exists between groups of observations with an equal edge variance and inverse mean edge relationship. By isolating groups of observations which are *mirrored* in the $\hat{E}[edge(m,i)] = 0.5$ line, we can remove (after examination) those falling to the right hand side. These observations are exact inverses and no classification tool will be able to deal with them perfectly as they are directly opposing observations to the left hand side of the $\hat{E}[edge(m,i)] = 0.5$ line. Figure 5.2 is redrawn as Figure 6.1 for ease of comparison in this discussion.

In identifying all observations with equal $\hat{Var}[edge(10,i)]$ and a $\hat{E}[edge(10,i)]$ /

$1 - \hat{E}[edge(10, i)]$ relationship, we may filter the original data and remove observations with inverse class labels. For the *colic* data, a total of 18 observations are flagged as having equal variance to at least one other observation. Closer inspection reveals 8 of these observations lie on the right hand side of the mean edge=0.5 line and have identical input value to their *mirror* observations, but opposite class labels. The next stage in the analysis sees these observations removed and the classifier retrained on a smaller dataset. The *colicrem* dataset now contains 292 observations. As discussed in Section 6.1.1, the method of generalisation error estimation will also need to be re-established. Using 10-fold cross-validation on *colicrem*, a generalisation error of 12.24% is obtained and the resulting trees for each fold are simpler than those obtained using the entire *colic* dataset (49 leaf nodes reduced to 38 leaf nodes). The 12.24% estimate of generalisation error is not representative of what would be expected for unseen data following the same distribution as the *colic* dataset. The level of measurable noise (in this case *inverse observations*) in the dataset has been unaccounted for. A more appropriate estimate of generalisation error is estimated via the e_{mirror} expression given in equation (6.1.1). For the *colic* dataset,

$$\begin{aligned} e_{mirror} &= \left(\frac{8}{300}\right) + \left(\frac{292}{300} \times 0.1224\right) \\ &= 0.1458 \end{aligned}$$

The new estimate of generalisation error is not dissimilar to the generalisation estimate of 14.99% obtained from training a single classifier on the entire *colic* dataset. However, new light has been shed on the *colic* dataset in partitioning the

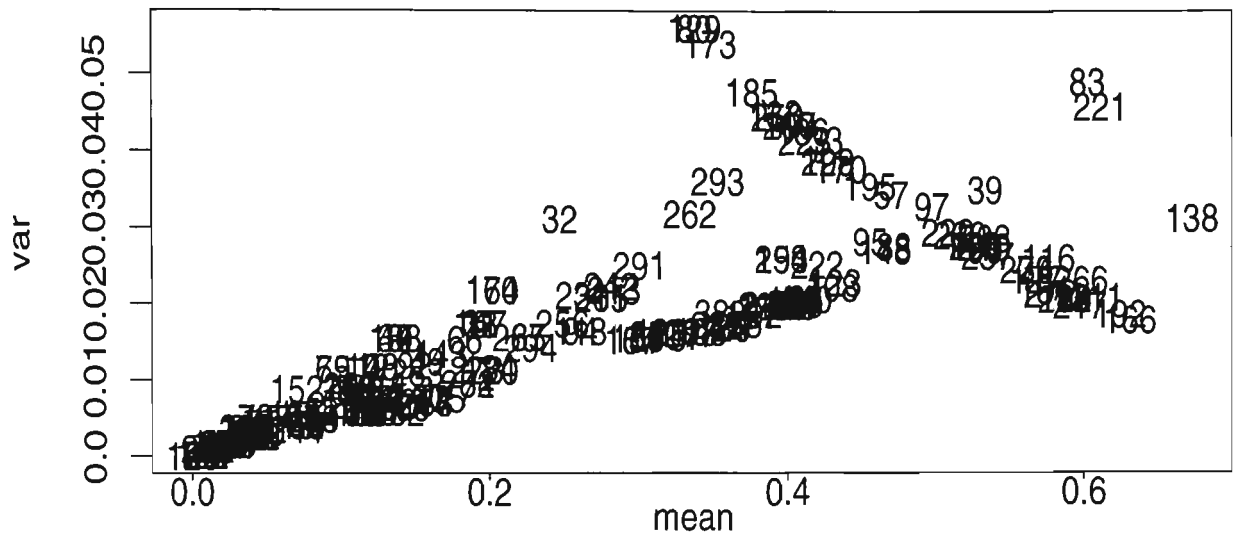

Figure 6.2: Plot of $\hat{Var}[edge(10,i)]$ vs. $\hat{E}[edge(10,i)]$: *heart-h* data.

error. i.e. 2.7% may be attributed to the presence of observations which oppose each other, and a further 11.9% to model inadequacies and inherent data noise.

6.1.3 Results on *heart-h* data

The plot of $\hat{Var}[edge(10,i)]$ versus $\hat{E}[edge(10,i)]$ for the *heart-h* data shown as Figure 5.6 exhibited positive and negative clusters of observations. In Section 5.2.4, the data was partitioned according to whether or not an observation fell into the negative cluster in Figure 5.6. The subsequent theory developed in Section 5.7 shows this method to be flawed as it will only detect observations which were initially misclassified. As with the *colic* data, symmetry in the $\hat{Var}[edge(10,i)]$ versus $\hat{E}[edge(10,i)]$ plot is evident and Figure 5.6 is redrawn as Figure 6.2 for ease of reference.

All observations with equal $\hat{Var}[edge(10,i)]$ and a $\hat{E}[edge(10,i)]$ / $1-\hat{E}[edge(10,i)]$

relationship we identified using a basic sorting and matching algorithm. For the *heart-h* data, a total of 46 observations are flagged as having equal variance to at least one other observation. On inspection, 11 of these observations were seen to lie to the right hand side of the $\hat{E}[edge(10, i)] = 0.5$ line and have identical input value to their *'mirror'* observations with opposite class labels. A smaller dataset named *hearthrem* was formed by removing *mirrored* observations to the right of the $\hat{E}[edge(10, i)] = 0.5$ line, resulting in a new training set containing 283 observations. Using 10-fold cross-validation on the *hearthrem* dataset, a generalisation error of 19.48% is obtained and the resulting trees are simpler in structure than those obtained after training on the entire *hearth* dataset (29 leaf nodes reduced to 24 leaf nodes) . The estimate of generalisation error for *hearthrem* is equal to 19.48% but this is not representative of what would be expected in future as the level of measurable noise has not been considered. Applying the method introduced in Section 6.1.1, a new estimate of generalisation error is given by:

$$\begin{aligned} e_{mirror} &= \left(\frac{11}{294}\right) + \left(\frac{283}{294} \times 0.1948\right) \\ &= 0.2249 \end{aligned}$$

As was noted for the *colic* dataset, the new e_{mirror} estimate of generalisation error is not substantially different to that obtained for a single tree trained on the entire dataset. Yet new information has been gleaned about the *heart-h* dataset in partitioning the generalisation error in that $\frac{11}{294} = 3.7\%$ of the data are *inverse observations* and result in deterioration of classifier performance. In any unseen data, this level of opposing observations is considered to be the same and

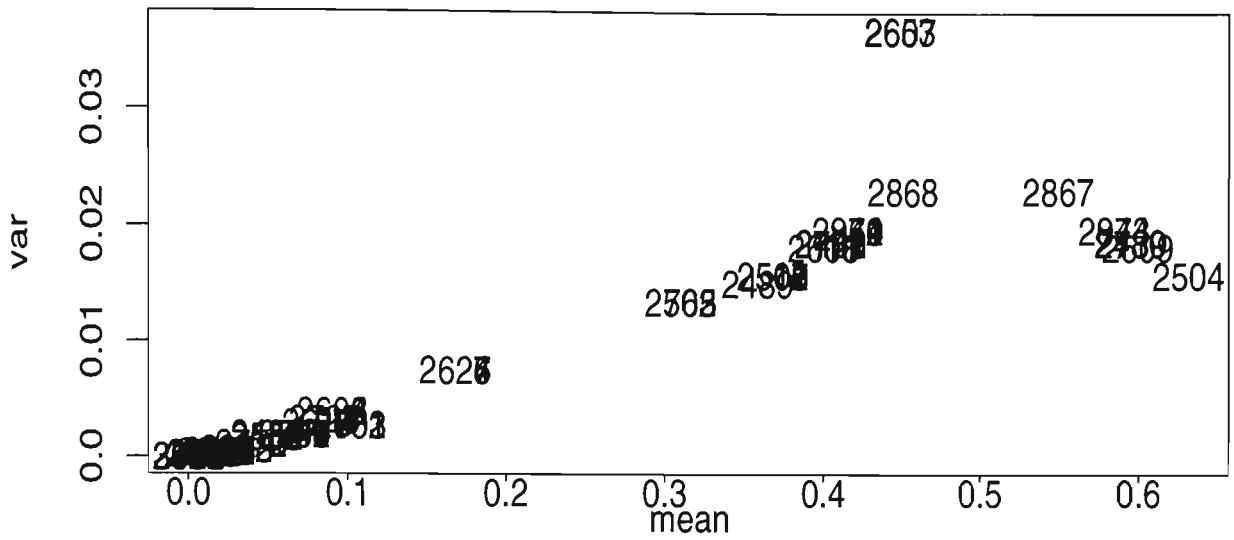

Figure 6.3: Plot of $\hat{Var}[edge(10,i)]$ vs. $\hat{E}[edge(10,i)]$: *BHPsmall* data.

an analyst may seek to remedy this situation.

6.1.4 Results on *BHPsmall* data

As outlined in Section 4.3.1, the dataset *BHPsmall* contains all observations which were predicted to have $obsflag = 1$ based on the decision tree drawn as Figure 4.4. Now that this subset of the *BHP* dataset is free of context differences, the process reverts to the original classification problem. After boosting, the $\hat{Var}[edge(10,i)]$ versus $\hat{E}[edge(10,i)]$ plot reveals the signature behaviour of *mirroring* in the $\hat{E}[edge(10,i)] = 0.5$ line. This is indicative of inverse data being present in the dataset which will deteriorate any model's performance. Figure 5.12 is shown again as Figure 6.3.

In identifying all observations with equal $\hat{Var}[edge(10,i)]$ and $\hat{E}[edge(10,i)]$ / 1 –

$\hat{E}[edge(10,i)]$ relationship, a total of 32 observations are flagged. Closer inspection reveals 8 of these observations lie on the left hand side of the $\hat{E}[edge(10,i)] = 0.5$ line. An example of such a group of observations is shown below, with the final column representing the values of the target class variable. From Table 6.2 it is clear to see that this value varies for identical input values given in the preceding columns, and depending on the current weightings the classifier is forced to make some incorrect predictions.

Table 6.2: Example of a cluster of *inverse observations* : *BHPsmall* dataset.

1	H	1	1	L	1	543	912	0	4	74	285	16.9	0.101	-36	81	107	120	505	566	496	1
1	H	1	1	L	1	543	912	0	4	74	285	16.9	0.101	-36	81	107	120	505	566	496	1
1	H	1	1	L	1	543	912	0	4	74	285	16.9	0.101	-36	81	107	120	505	566	496	0
1	H	1	1	L	1	543	912	0	4	74	285	16.9	0.101	-36	81	107	120	505	566	496	0
1	H	1	1	L	1	543	912	0	4	74	285	16.9	0.101	-36	81	107	120	505	566	496	1
1	H	1	1	L	1	543	912	0	4	74	285	16.9	0.101	-36	81	107	120	505	566	496	1
1	H	1	1	L	1	543	912	0	4	74	285	16.9	0.101	-36	81	107	120	505	566	496	1

These 8 observations are removed and the classifier retrained on a reduced *BHPsmall* dataset, renamed to *BHPsmallrem* containing 534 observations. Using 10-fold cross-validation on the *BHPsmallrem* dataset results in a generalisation error of 1.49% being obtained, and the resulting tree is identical to that obtained using the entire *BHPsmall* dataset. This is not discouraging since the decision tree formed for the *BHPsmall* dataset (Figure 4.5) had a single split and is itself extremely simple in structure. Despite the decision trees being identical for the two methods, an important dataset feature has been identified and future classification on this dataset will be more informed as to the level of opposing observations expected to be present in unseen observations.

Now, generalisation error as estimated using 10-fold cross-validation on the *BHPsmallrem* dataset is equal to 1.49%. Applying the expression given in equation

(6.1.1) we obtain:

$$\begin{aligned} e_{mirror} &= \left(\frac{8}{542}\right) + \left(\frac{534}{542} \times 0.0149\right) \\ &= 0.029 \end{aligned}$$

Again the e_{mirror} estimate closely matches the generalisation error estimate for a single decision tree on the entire *BHPsmall* dataset, yet we now know that this 2.9% comprises 1.49% inverse (possibly mislabelled) observations.

6.2 Results on *BHPlarge*

As outlined in Section 4.3.1, the dataset *BHPlarge* contains all observations which were predicted to have $obsflag = 0$ based on the decision tree drawn as Figure 4.4. Now that this subset of the *BHP* dataset is free of context differences, the process reverts to the original classification problem. After boosting, the $\hat{Var}[edge(10,i)]$ versus $\hat{E}[edge(10,i)]$ plot reveals the signature behaviour of *mirroring* in the $\hat{E}[edge(10,i)] = 0.5$ line. This is indicative of inverse data being present in the dataset which will deteriorate any model's performance. Figure 5.13 is shown again as Figure 6.4.

In identifying all observations with equal $\hat{Var}[edge(10,i)]$ and $\hat{E}[edge(10,i)]$ / $1 - \hat{E}[edge(10,i)]$ relationship, a total of 964 observations are flagged. Closer inspection reveals 348 of these observations lie on the left hand side of the $\hat{E}[edge(10,i)] = 0.5$ line. Examples of observations exhibiting this behaviour are given in Table 6.2 in the previous section, whereby we see varying class labels for identical input values.

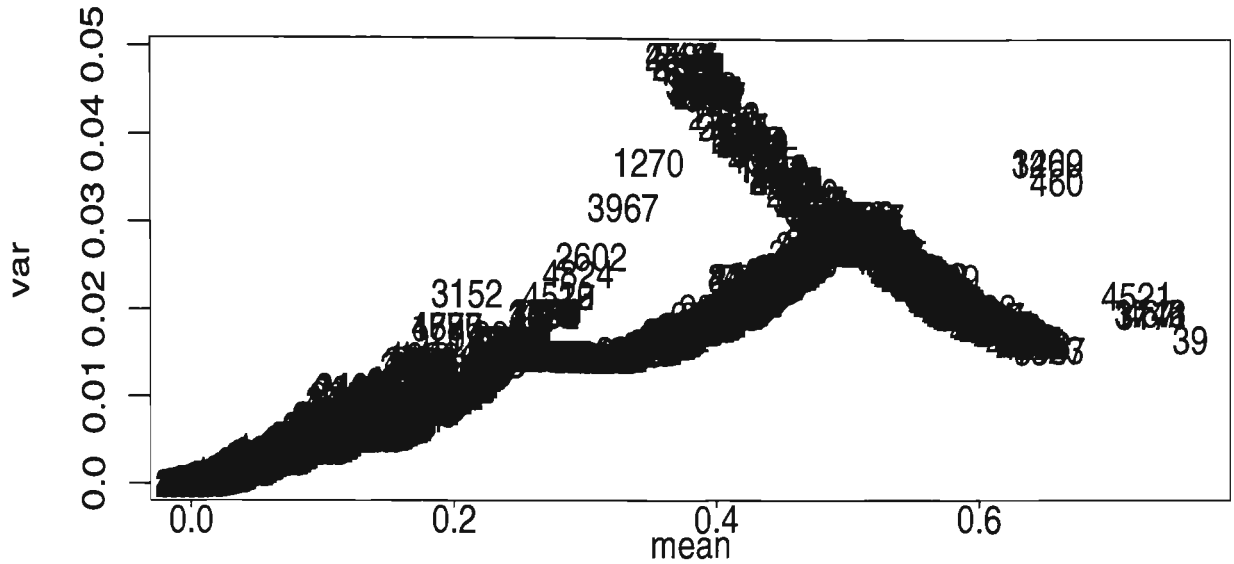

Figure 6.4: Plot of $\hat{Var}[edge(10,i)]$ vs. $\hat{E}[edge(10,i)]$: *BHPlarge* data.

The 348 *mirrored* observations are removed and the classifier retrained on a reduced *BHPlarge* dataset, renamed to *BHPlargerem* containing 534 observations. Using 10-fold cross-validation on the *BHPlargerem* dataset results in a generalisation error of 15.13% being obtained, and the resulting tree is simpler than that obtained using the entire *BHPlarge* dataset. Again, an important dataset feature has been identified and a significant number of inverse observations flagged as contributing to decreased generalisation performance.

Now, generalisation error as estimated using 10-fold cross-validation on the *BHPlargerem* dataset is equal to 15.13%. Applying the expression given in equation (6.1.1) we obtain:

$$\begin{aligned} e_{mirror} &= \left(\frac{348}{4905}\right) + \left(\frac{4557}{4905} \times 0.15143\right) \\ &= 0.2115 \end{aligned}$$

Again the e_{mirror} estimate closely matches the generalisation error estimate for a single decision tree on the entire *BHPlarge* dataset, yet we now know that this 21.15% comprises 7.10% inverse (possibly mislabelled) observations.

6.2.1 Summary of mirroring results

This Section provides a summary of results obtained after applying the new e_{mirror} methodology to the datasets tested in this Thesis. In all cases we see e_{mirror} is not significantly different from the initial generalisation error estimate based on the entire dataset (both generalisation error estimates are the result of fitting a single decision tree). However, the first column in Table 6.3 gives an indication of the proportion of the generalisation error which results from observations which oppose each other.

In any case, this process certainly doesn't deteriorate generalisation error, and for all datasets tested we have gained model simplicity. The only exception is the *BHPsmall* dataset where the initial decision tree was stump and unable to be simplified further.

Difficulty in generalisation error estimation always arises as we assume the unseen data will follow the same distribution as the training data. Without additional information on new unseen data, this is the best we can hope for. Therefore, in removing noise and retraining we really need to assume that unseen data will contain the same level of noise and possibly mislabelled observations as the training set. Furthermore, by using this new generalisation error estimate we now have a 'partitioned estimate'.

In removing opposing observations we may seek to measure additional variables which are not present in the existing dataset but may be key in differentiating between classes and hence negating the observations' inverse status.

Table 6.3: New estimates of generalisation error for a selection of UCI datasets.

dataset	% data mirrored	e_{init}	e_{mirror}
colic	2.7	14.99	14.58
heart-h	3.7	22.61	22.49
BHPsmall	1.41	2.98	2.9
BHPlarge	7.09	21.16	21.15

6.3 A Proposed Process

The analysis process developed and applied in this Thesis may be represented as a series of unique tasks. This representation is given in Figure 6.5. Where the term 'boosting' appears in a rectangle, the *AdaBoost* algorithm is applied. In this thesis, 10 boosting trials were applied with c4.5 using default options as the base learner. It is important to note, however, that the theory developed in Chapters 5 and 6 is not restrictive on the exact form of the classifier to be used, nor the number of iterations applied. Also worthy of noting is the final retraining step resulting in a simpler classifier than would otherwise have been obtained had the noise and *inverse observations* gone undetected by the analyst. Manual detection of inverse and noisy observations is tedious and time consuming, particularly in the case of large, high dimension datasets.

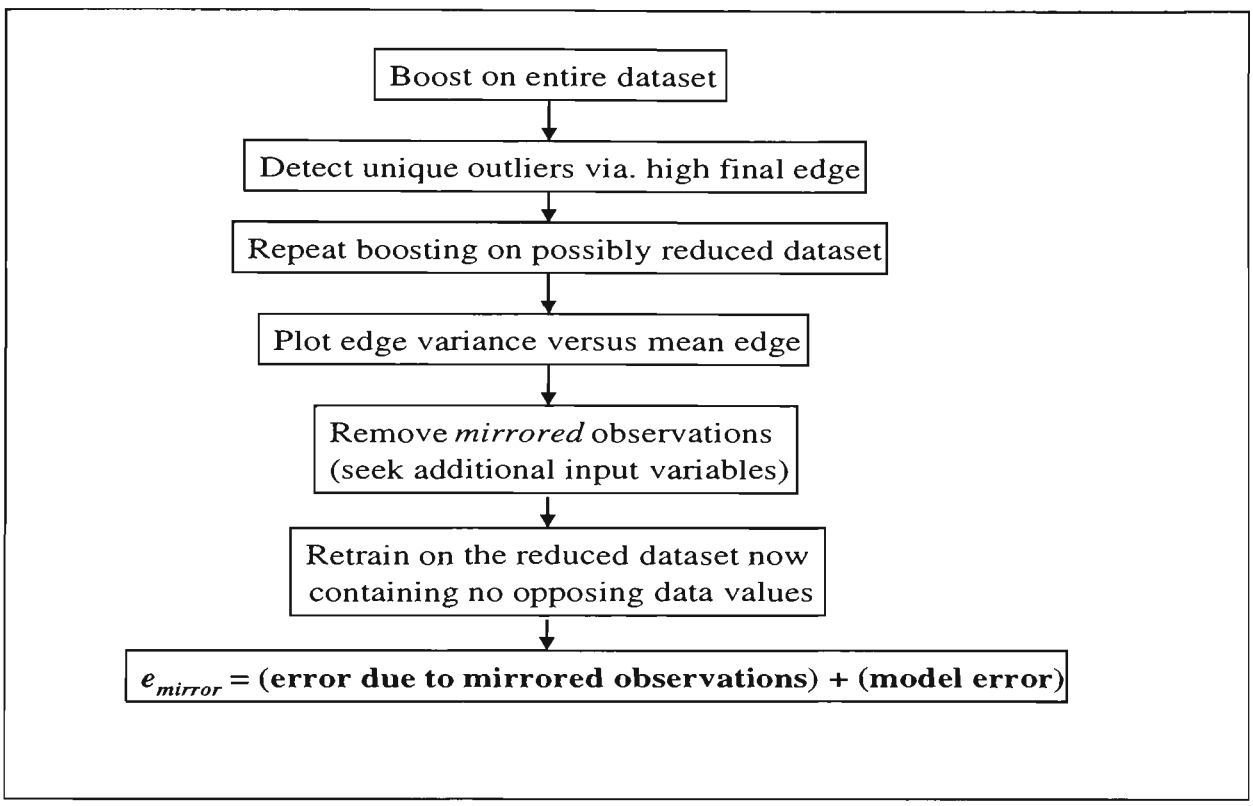

Figure 6.5: Process flow for inverse detection and generalisation error partitioning.

Chapter 7

Conclusions and Further Work

7.1 Conclusion

This Thesis has described an empirical and theoretical journey in extending boosting to solve a variety of problems and devise a more informative measure of generalisation error. After demonstrating boosting's tendency to balance the number of misclassifications per observation as the number of trials proceeds, a method for detecting noisy observations was introduced and tested on a variety of datasets. It is widely known that boosting fails on noisy data, often terminating with errors too high for the algorithm to proceed. Therefore the ability to detect and account for noisy data is highly desirable, so that misleading information does not hinder a classifier.

An extension to the notion of detecting unique noisy observations is to detect noisy clusters or subgroups of observations which behave differently to the remainder of the data. An extreme example of this is when two sections of the data are the inverse of each other. In the binary case, this implies opposing

class values for identical input values. Undetected, this results in increased generalisation error and more complex models prone to overfitting as the classifier is torn between observations which are contradictory. If such situations can be detected, the dataset partitioned and new classifiers trained on each partition, simple robust models should be apparent for each partition with an improved generalisation error. This concept was noted empirically and sound theoretical derivations ensued to substantiate the empirical claims and methodologies.

Futher exploration was carried out on the inverse observation scenario i.e. where $\mathbf{x}_k = \mathbf{x}_i$ but $y_k = 1 - y_i$, resulting in opposing class values for equivalent input values. Multivariate statistics formed the basis of a proof that such observations will have equal edge variance and inverse mean edges. Such a result is significant in practice as it is the foundation of a simple algorithm which is invaluable in detecting inverse observations in what may otherwise have been a tedious and time consuming task in high dimensional data. Again, if such observations go undetected the resulting classifiers are unnecessarily complex and generalisation error pessimistic.

The final result of this Thesis was to suggest the partitioning of generalisation error according to the identification or otherwise of inverse observations. A new measure of generalisation error was suggested whereby components pertaining to the proportion of inverse observations and a more realistic generalisation error based on the remainder of the data were combined. Although this results in equivalent generalisation error estimates to those estimated using standard cross

validation techniques, we have a more informative gauge which gives quantitative measures of expected unseen data noise versus model repeatability.

In conclusion we see that the methods introduced in this Thesis do not make any adjustments to Freund and Schapire's standard *AdaBoost* algorithm [20], but rather exploit the existing algorithm's output to gain insight into data structure prior to classification. As a result some difficulties with noisy data may be overcome and the analyst gains more succinct information about the makeup of the dataset one is working with.

7.2 Further Work

Many avenues remain open for research into the methodologies and ideas introduced in this Thesis. The area of data mining as a whole continues to remain a fertile ground for research into new algorithms and improvements to existing algorithms. Whilst only dealing with the classification problem, this Thesis has demonstrated the breadth and depth of issues arising even when we are concerned with the simplest binary case. The following list suggests further extensions from the noise methodologies and inverse observation detection covered in this Thesis.

1. In this Thesis, unweighted edge means and variances were used as the basis of new methodologies. Whilst theory on unweighted mean and variances was developed, it remains an open problem to formulate new methods of weighting to have a measure of mean edge and edge variance which better encompass contributions made from misclassifications across all iterations

instead of biasing the measures to over-represent earlier misclassifications.

2. It may be possible that plots shown as Figures 5.17, 5.17 and 5.18 be represented by a general mathematical expression. Bounds for the edge variance were derived in Chapter 5, yet the exact mathematical form of the bell shaped curves is elusive.

3. The distribution of inverse observations would certainly have an effect on the partitioning of generalisation error and resulting data noise estimates. For example a group of 10 observations having 1 of these observations as an inverse would have less impact on any classification system than a group of 10 observations with 4 observations being inverse. This issue also leads to the decision as to which inverse observations to remove - and in fact, should they be removed or relabelled to match the majority class?

4. It would be of value to determine the level of inverse observations permissable within a dataset before boosting or other classifiers simply breakdown and are unable to deal with the contradictions present.

5. The bound developed in equation (5.7.3) may be tightened by developing a bound in terms of misclassification at later iterations i.e. $I_2(i), \ldots, I_m(i)$.

6. Two publications are included as appendices in this Thesis. Another publication titled "Variance Reduction Trends on Boosted Classifiers" has been accepted pending revision for the Journal of Applied Mathematics and Decision Sciences.

Bibliography

[1] K. Ali and M. Pazzani, *Error reduction through learning multiple descrptions*, Machine Learning **24:3** (1996).

[2] C.L. Blake and C.J. Merz, *UCI repository of machine learning databases*, 1998.

[3] Leo Breiman, *Arcing the edge*, Tech. Report 486, University of California, Berkeley CA. 94720, 1995.

[4] ———, *Bagging predictors*, Machine Learning **26** (1996), no. 2, 123–140.

[5] ———, *Bias, variance and arcing classifiers*, Tech. Report 460, University of California, Berkeley, 1996.

[6] ———, *Prediction games and arcing classifiers*, Tech. Report 504, University of California, Berkeley, 1997.

[7] ———, *Random forests - random features*, Tech. Report 567, University of California, Berkeley, 1999.

[8] Leo Brieman, Jerome.H. Friedman, J.Olshen, and C.Stone, *Classification and regression trees*, Wadsworth and Brooks, 1994.

[9] William Cohen, *Fast effective rule induction*, Proceedings of the Twelfth International Conference on Machine Learning, 1995, pp. 115–123.

[10] Mark Devaney and Ashwin Ram, *Dynamically adjusting concepts to accommodate changing contexts*, Workshop on Learning in context Sensitive domains ICML (Bari, Italy), 1996, pp. 6–14.

[11] Thomas Dietterich and Ghulum Bakhiri, *Error-correcting output codes: A general method for improving multiclass inductive learning programs*, Proceedings American Association of Articifial Inteligence 1991, AAAI Press / MIT Press, 1991.

[12] Thomas.G. Dietterich, *Machine learning research: Four current directions*, AI Magazine **18** (1997), no. 4, 99–137.

[13] Thomas.G. Dietterich and Eun Bae Kong, *Machine learning,bias, statistical bias, and statistical variance of decision tree algorithms*, Tech. report, Department of Computer Science, Oregon State University, 1995.

[14] Bradley Efron, *The jacknife, the booststrap and other resampling plans*, CBMS-NSF - Regional Conference Series in Applied Mathematics **38** (1982).

[15] Bradley Efron and Robert.J. Tibshirani, *An introduction to the bootstrap*, Chapman and Hall, 1993.

[16] John.F. Elder and Dean W. Abbott, *A comparison of leading data mining tools*, Fourth International Conference on Knowledge Discovery and Data Mining (New York), 1998.

[17] John.F. Elder and Daryl Pregibon, *A statistical perspective on kdd*, Proceedings KDD'95, 1995, pp. 87–93.

[18] Usama Fayyad, *Editorial*, Data Mining and Knowledge Discovery **2** (1998), 5–7.

[19] Usama Fayyad, Gregory Piatesky-Shapiro, and Padhriac Smyth, *From data mining to knowledge discovery in databases*, AI Magazine **17** (1997), no. 3, 37–54.

[20] Yoav Freund and Robert.E. Schapire, *A decision-theoretic generalisation to on-line learning and an application to boosting*, Journal of Computer and System Sciences **55(1)** (1997), 119–139.

[21] Jerome.H. Friedman, *On bias, variance, 0/1 - loss, and the curse-of-dimensionality*, Data Mining and Knowledge Discovery **1(1)** (1997), 55–77.

[22] ———, *Statistics 315b: Statistical aspects of data mining (winter 1998)*, 1998.

[23] ———, *Greedy function approximation:a gradient boosting machine*, 1999.

[24] Jerome.H. Friedman, Trevor Hastie, and Robert Tibshirani, *Additive logistic regression: a statistical perspective on boosting*, Tech. report, Department of Statistics, Stanford University, 1998.

[25] Clark Glymour, David Madigan, Daryl Pregibon, and Padhriac Smyth, *Statistical themes and lessons for data mining*, Data Mining and Knowledge Discovery **1** (1997), 11–28.

[26] L. Hansen and P. Salamon, *Neural network ensembles*, IEEE Transactions on Pattern Analysis and Machine Intelligence **12** (1990), 993–1001.

[27] Peter.J. Huber, *From large to huge: A statistician's reactions to kdd and dm*, Proceedings KDD'97, 1997, pp. 304–308.

[28] Maurice.G. Kendall and Alan Stuart, *The advanced theory of statistics*, Oxford University Press, Oxford , UK, 1963.

[29] Eun Bae Kong and Thomas G. Dieterrich, *Error-correcting output coding corrects bias and variance*, Proceedings of the Twelfth International Conference On Machine Learning, 1995, pp. 313–321.

[30] John Maindonald, *New approaches ot using scientific data statistics, data mining and related technologies in research and research training*, Tech. report, The Australian National University (The Graduate School), 1999.

[31] L. Mason, Peter Bartlett, and Jonathon Baxter, *Improved generalisation error through explicit optimization of margins*, Machine Learning **38** (1999), no. 3, 243–255.

[32] Donald Michie, David. J. Spiegelhalter, and Charles. C. Taylor, *Machine learning, neural and statistical classification*, Ellis Horwood Series on Artificial Intelligence, 1994.

[33] J.Ross Quinlan, *C4.5: Programs for machine learning*, Morgan Kaufmann, San Fransisco, California, 1993.

[34] ———, *Bagging, boosting and c4.5*, Proceedings of the Thirteenth National Conference on Artificial Intelligence (Menlo Park, California), American Association for Artificial Intelligence, 1996, pp. 725–730.

[35] G. Rätsch, T. Onoda, K-R. Mueller, and Pedro Domingos, *Soft margins for adaboost*, Machine Learning **42(3)** (2001), 287–320.

[36] G. Rätsch, B. Schölkpof, A.J. Smola, K-R. Mueller, and S. Mika, *ν-arc:ensemble learning in the presence of outliers*, NIPS 12, 2000, pp. 561–567.

[37] Stuart Russell and Peter Norvig, *Artificial intelligence a modern approach*, Prentice Hall, Upper Saddle River, New Jersey 07458, 1995.

[38] Steven.L. Salzberg, *On comparing classifiers: A critique of current research and methods*, Data Mining and Knowledge Discovery **1** (1999), 1–12.

[39] Claude Sammut, S. Hurst, D. Kedzier, and Donald Michie, *Learning to fly*, Proceedings of the Ninth International Confernce on Machine Learning, Morgan Kaufmann, 1992.

[40] Robert.E. Schapire, *Using output codes to boost multiclass learning problems*, Fourteenth International Conference on Machine Learning (San Fransisco, California), Morgan Kaufmann, 1997, pp. 313–321.

[41] Robert.E. Schapire, Yoav Freund, Peter Bartlett, and Wee Sun Lee, *Boosting the margin:a new explanation for the effectiveness of voting methods*, Annals of Statistics **26(5)** (1998).

[42] Robert.E. Schapire and Yoram Singer, *Improved boosting algorithms using confidence rated predictions*, 11th COLT : Computational Learning Theory, 1998, pp. 80–91.

[43] Terry.M. Therneau and Elizabeth Atkinson, *An introduction to recursive partitioning using the rpart routine*, Tech. Report 61, Mayo Clinic, Section of Statistics, 1997.

[44] John Tukey, *Bias and confidence in not quite large samples*, Annals of Mathematical Statistics **29** (1958), 614.

[45] E. Wegmann, *Huge data sets and the frontiers of computational feasibility*, Journal of Computational and Graphical Statistics **4** (1995), 281–295.

[46] Sholom.M. Weiss and Kulikowski, *Computer systems that learn*, Morgan Kaufmann, San Mateo, CA, 1991.

Appendix A

Publication (Wheway, 2000)

Wheway, V. (2000) *Using boosting to detect noisy data.* In Lecture Notes in Computer Science Volume 2112 : AIApps 2000. Applications of AI in Industry.

Using boosting to detect noisy data

Virginia Wheway
virg@cse.unsw.edu.au

School of Computer Science and Engineering,
University of New South Wales
Sydney NSW 2052 Australia

Abstract

Noisy data is inherent in many real-life and industrial modelling situations. If prior knowledge of such data was available, it would be a simple process to remove or account for noise and improve model robustness. Unfortunately, in the majority of learning situations, the presence of underlying noise is suspected but difficult to detect.
Ensemble classification techniques such as *bagging*, (Breiman, 1996a), *boosting* (Freund & Schapire, 1997) and *arcing* algorithms (Breiman, 1997) have received much attention in recent literature. Such techniques have been shown to lead to reduced classification error on unseen cases, and this paper demonstrates that they may also be employed as noise detectors. Recently defined diagnostics such as *edge* and *margin* (Breiman, 1997; Freund & Schapire, 1997; Schapire et al., 1998) have been used to explain the improvements made in generalisation error when ensemble classifiers are built. The distributions of these measures are key in the noise detection process introduced in this study.
This paper presents some empirical results on edge distributions which confirm exisiting theories on boosting's tendency to 'balance' error rates. The results are then extended to introduce a methodology whereby boosting may be used to identify noise in training data by examining the changes in edge and margin distributions as boosting proceeds.

1 Introduction

This paper is concerned with the classification problem, whereby a learner is presented with a training set comprising of a series of n labelled training examples of the form $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)$, with $y_i \in (1, \ldots, k)$. The learner's task is to use these training examples to produce an hypothesis, $h(\mathbf{x})$, which is an estimate of the unknown relationship $y = f(\mathbf{x})$. This 'hypothesis' then allows future prediction of y_i given new input values of $\mathbf{x}$. A classifier built by combining individual $h(\mathbf{x})$'s to form a single classifier is known as an ensemble. Whilst there are many ensemble building methods in existence, this discussion focusses on the method of boosting which is based on a weighted subsampling of the training examples. Introduced by Freund and Schapire in 1997, boosting is recognised as being

one of the most significant recent advances in classification (Freund & Schapire, 1997). Since its introduction, boosting has been the subject of many theoretical and empirical studies (Breiman, 1996b; Quinlan, 1996; Schapire et al., 1998). Empirical studies have shown that ensembles grown from repeatedly applying a learning algorithm over different subsamples of the data result in improved generalisation error.

Boosting is an iterative procedure which trains a learner over the n wieghted observations. Boosting begins with all with all training examples being given equal weight (i.e. $\frac{1}{n}$) At the $m+1$-th iteration, examples which were classified incorrectly at the m-th iteration have their weight increased multiplicatively so that the total weight on incorrect observations is equal to 0.5. Hence, the learning algorithm will be given more opportunity to explore areas of the training set which are more difficult to classify. Hypotheses from these parts of the space make fewer mistakes on these areas and play an important role in prediction when all hypotheses are combined via weighted voting. At each iteration, the weighted error is stored and used in the final voting weight when individual classifiers are combined to form the ensemble. Accuracy of the final hypothesis depends on the accuracy of *all* the hypotheses returned at each iteration and the method exploits hypotheses that predict well in more difficult parts of the instance space. An advantage of boosting is that it does not require any background knowledge of the performance of the underlying weak learning algorithm. Refer to Table 1 for details of the boosting algorithm and its weight update methodology.

Table 1: AdaBoost:M1

AdaBoost

Input: n training instances x_i with labels y_i. Maximum trials, M. Base learner, H.

Initialization: All training instances begin with weight $w_i^0 = 1/n$.

Repeat for M trials:

- Induce classifier, h_m, using weighted training data and H.
- ϵ_m = weighted error for h_m on the training data. If $\epsilon_m > 1/2$, discard h_m and stop boosting. (If $\epsilon_m = 0$, then h_m gets infinite weight.)
- Classifier weight, $\beta_m = log\frac{\epsilon_m}{1-\epsilon_m}$
- Re-weight training instances:
  if $h_m(x_i) \neq y_i$ then,
  $w_i^{m+1} = w_i^m/(2\epsilon_m)$
  else, $w_i^{m+1} = w_i^m/2(1-\epsilon_m)$

Unseen instances are classified by voting the ensemble of classifiers h_m with weights w_m.

2 Current Explanations of the Boosting Mechanism

Ensemble classifiers and the reasons for their improved classification accuracy has provided a fertile ground for research in both the machine learning and statistical communities. In theory, as the combined classifier complexity increases, the gap between training and test set error should increase. However, this is not reflected in empirical

studies. There is strong empirical support for the view that overfitting is less of a problem (or perhaps a different problem) when boosting and other resampling methods are used to improve a learner. Some authors have addressed this issue via bias and variance decompostions in an attempt to understand the stability of a learner (Breiman, 1997; Breiman, 1996b; Friedman, 1997).
The following is a summary of the key comments and conclusions made on boosting and ensemble classification to date:

- Breiman (1996b) claims the main effect of the adaptive resampling when building ensembles is to reduce variance, where the reduction comes from adaptive resampling and not the specific form of the ensemble forming algorithm.
- A weighted algorithm in which the classifiers are built via weighted observations performs better than weighted resampling at each iteration, apparently due to removing the randomisation (Friedman, Hastie & Tibshirani, 1998).
- Confidence rated predictions outperform boosting algorithms where a 0/1 loss is applied to incorrect classification (Freund & Schapire, 1996; Schapire & Singer, 1998).
- Successful ensemble classification is due to the non-overlap of errors (Dietterich, 1997) i.e. observations which are classified correctly by one hypothesis are classified incorrectly by others and vice versa.
- Margin and edge analysis are recent explantions (Breiman, 1997; Schapire et al., 1998). More detail on these measures and related studies is provided in the next section.

3 Edge and Margin

Recent explanations as to the success of boosting algorithms have their foundations in margin and edge analysis. These two measures are defined for the ith training observation at trial m as follows:
Assume we have a base learner which produces hypothesis $h_m(\mathbf{x})$ at the m-th iteration, and an error indicator function, $I_m(\mathbf{x}_i) = I(h_m(\mathbf{x}_i) \neq y_i)$. Let c_m represent the vote for the m-th hypothesis with $\sum_m c_m = 1$. Then,

- $edge_i(m, \mathbf{c})$ = total weight assigned to all incorrect classes.
  Breiman (1997) defines the edge as:

$$edge_i(m, \mathbf{c}) = \sum_{j=1}^{m} c_j I_j(\mathbf{x}_i) \quad (1)$$

- $margin_i(m, \mathbf{c})$ = total weight assigned to the correct class minus the maximal weight assigned to any incorrect class.

For the 2 class case $margin_i(m, \mathbf{c}) = 1 - 2\ edge_i(m, \mathbf{c})$ and in general, $margin_i(m, \mathbf{c}) \geq 1 - 2\ edge_i(m, \mathbf{c})$.
Whilst more difficult to compute, the value of the margin is relatively simple to interpret. Margin values will always fall in the range $[-1, 1]$, with high positive margins indicating confidence of correct classification. An example is classified incorrectly if it has a negative margin. The edge on the other hand cannot be used as an indicator variable for correct classification (except in the

2-class case). Whilst the margin is a useful measure due to its interpretability, mathematically it is perhaps not as robust and tractable as the edge.
Schapire et al. (1998) claim that boosting is successful because it creates a higher margin distribution and hence increases the confidence of correct classification. Breiman, however, claims the high margin explanation is incomplete and introduces new ensemble techniques which actively improve margin distributions but do not result in improved generalisation error (Brieman, 1997, 1999).
This study demonstrates the tendency for boosting to 'balance' the edge (or margin) in its quest to classify more difficult observations. Using this property, a method of detecting noise in training data is presented. Methodology for the study is discussed in detail in the next section.

4 Empirical Results

In all experiments, the decision tree learner C4.5 with default values and pruning was used as the base classifier, with a boosted ensemble being built from $M = 50$ iterations. Datasets used are a selection from the UCI[1] Machine Learning Repository. Results from only four of these datasets are presented in this paper. These four datasets were chosen to provide a representative mixture of dataset size and boosting performance previously reported. All edge distribution results are certainly replicable for other UCI datasets. 10-fold crossvalidation was applied whereby the original training data was shuffled randomly and split into 10 equal-sized partitions. Each of the ten partitions was used in turn as a test set for the ensemble generated using the remaining 90% of the original data as a training set.
At each iteration, the values for edge were calculated for each observation in the training set. The average and variance of $edge_i(m, \mathbf{c})$ were calculated as follows:

$$\hat{E}[edge_i(m,\mathbf{c})] = \frac{1}{n}\sum_{i=1}^{n} edge_i(m,\mathbf{c})$$

$$\hat{Var}[edge_i(m,\mathbf{c})] = \frac{1}{n}\sum_{i=1}^{n}(edge_i(m,\mathbf{c}) - \hat{E}[edge_i(m,\mathbf{c})])^2$$

The graphical results of these trials for the colic, glass and letter datasets appear overleaf.

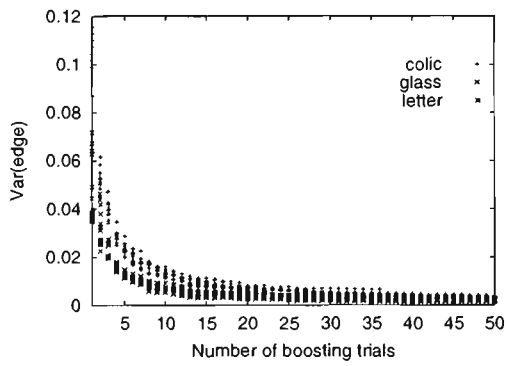

Figure 1: Var edge vs number of boosting trials

Refer to Figure 1 to note an apparent exponential decrease in variance perhaps indicating an asymptote of zero or some small value, ϵ. Figure 2 shows the average edge increasing as the number of boosting trials increases. These results show a homogenisation of the edge, implying that the observation error rate is becoming more uniform. i.e observations which were initially clas-

[1] http://www.ics.uci.edu/~mlearn/MLRepository.html

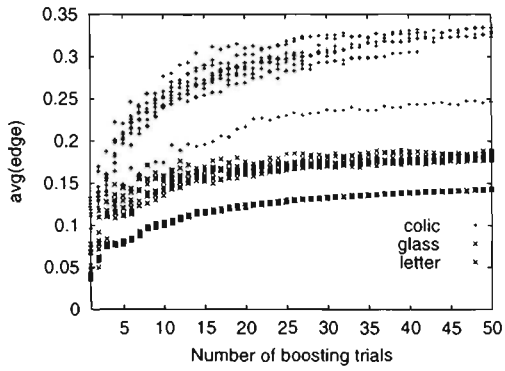

Figure 2: Average edge vs number of boosting trials

sified correctly are classified incorrectly in later rounds in order to classify 'harder' observations correctly. This results in the percentage of incorrect classifications for each observation becoming more uniform as the number of boosting trials increases. This notion is consistent with margin distribution results presented by Schapire et al. (1998) and edge and margin results by Breiman (1997, 1999).
The most dramatic variance decay is seen in boosting trials $m \leq 5$ i.e. most of the 'hard' work appears to be done in the first few trials. This observation is consistent with several authors noting in earlier published empirical studies that little additional benefit is gained after 10 boosting trials when a relatively strong learner is used as the base learner.

5 A method for detecting noise

The notion of 'balancing' was discussed in the previous section. This property may now be exploited to detect noise in the training data. Since we expect the distribution of the edge values to become more uniform as the number of boosting trials increases, we may assume deviations from this distribution to be caused by noisy or incorrect data. Noisy data is certainly difficult to classify and some observations may be too difficult to classify correctly in all but a few boosting iterations. Such observations would have edge values which remain high due to persistent misclassification. If these deviations from the overall edge distribution can be detected at say, m=10-20 iterations, then the associated observations may be deemed to be noisy data. Removal of such observations should lead to improved classficiation accuracy on training and test data. The choice of optimal m is still unclear but at m=10-20, computing time is still relatively small and deviations from distributions should already be apparent.

To test this hypothesis, and check whether 'offending' noisy data could be identified, noise was injected into the letter and census datasets by assigning random class labels to 5% of the data. These datasets were chosen because of their size (20,000 and 32,000 observations respectively). To perform this randomisation, the data was shuffled, then the first 5% of observations were assigned a random class label before the data was reshuffled again. This ensured no systematic bias in the noise while still retaining the observation number for later comparison. The edge values were captured after 15 iterations and plotted against observation number below.

Referring to Figure 3 below, a clear distinction between edge values can be

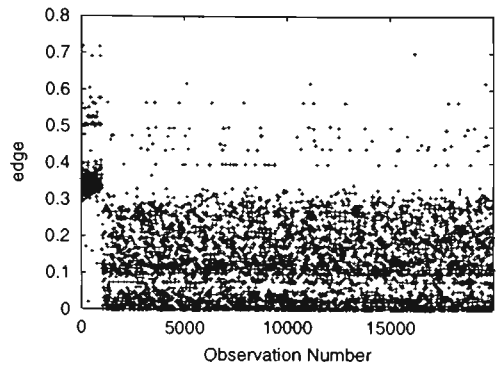

Figure 3: edge vs observation number: letter data with class labels randomly assigned for obs. 1-1000

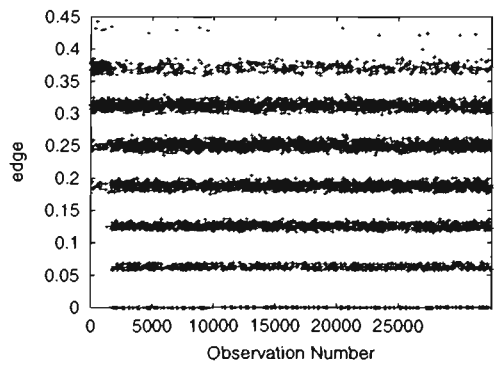

Figure 4: edge vs observation number: census data with class labels randomly assigned for obs. 1-1600

seen around observation 1000. For the letter data, observations 0-1000 were deliberately relabelled with random class values to simulate noise. The remainder of the observations were untouched, yet it is still possible that some of these observations may be noisy but as yet undetected in the raw UCI dataset. For the census data, another clear boundary is evident, this time with earlier observations (i.e. observations 0-1500) showing failure to reach lower edge values - note the gap in the plot on the lower left hand corner. Although this example is contrived with known noise being introduced, it demonstrates an important result in being able to identify and eliminate noise. These results indicate that truncating the edge distribution at the top say, 5% of edge values or setting a threshold on edge values and relearning could be an effective way of reducing noise and improving model performance. It would also seem that this technique would be best applied to larger datasets where 5% of observations is not an insignificant number.

Conclusion

This study has presented some interesting results on the variance of the edge when a boosted ensemble is formed. These results confirmed existing research on margin distrubtions. An empirical study on 2 larger datasets demonstrated that is is possible to track deviations in the edge (or margin) distribution when trying to identify the existence or otherwise of noisy data. Although this concept was only tested on known datasets from the UCI repository, initial results are certainly encouraging and further application and

research on large scale industrial datasets should lead to higher confidence in the models built from improved training data.

References

Breiman, L. (1996a). Bagging predictors. *Machine Learning, 26(2)*, 123–140.

Breiman, L. (1996b). *Bias, Variance and Arcing Classifiers* (Technical Report 460). Statistics Department, University of California, Berkeley.

Breiman, L. (1997). *Arcing the edge* (Technical Report 486). Statistics Department, University of California, Berkeley.

Breiman, L. (1999). *Random Forests - Random Features* (Technical Report 567). Statistics Department, University of California, Berkeley.

Dietterich, T. G. (1997). Machine learning research: Four current directions. *AI Magazine, 18(4)*, 99–137.

Freund, Y., & Schapire, R. E. (1996). Experiments with a new boosting algorithm. *Proceedings of the Thirteenth International Conference on Machine Learning* (pp. 148–156). Morgan Kaufmann.

Freund, Y., & Schapire, R. E. (1997). A decision-theoretic generalisation to on-line learning and an application to boosting. *Journal of Computer and System Sciences , 55(1)*, 119–139.

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, J. H. , Hastie, T. & Tibshirani, R.(1998). *Additive logisitic regression: a statistical perspective on boosting.* (Technical Report 199). Department of Statistics, Stanford Univeristy.

Kendall, M. G. & Stuart, A. (1963). *The advanced theory of statistics.* Oxford University Press, Oxford, UK.

Quinlan, J. R. (1993). *C4.5: Programs for Machine Learning.* Morgan Kaufmann.

Quinlan, J. R. (1996). Bagging, boosting and C4.5. *Proceedings of the Thirteenth National Conference on Articifical Intelligence.* (pp. 725–730). Menlo Park California, American Association for Artificial Intelligence.

Schapire, R. E. , Freund, Y., Bartlett, P. & Lee, W. S. (1998). Boosting the margin:a new explanation for the effectiveness of voting methods. *Annals of Statistics, 26(5)*, 1651–1686.

Schapire, R. E. & Singer, Y. (1998) Improved boosting algorithms using confidence rated predictions. *Proceedings of the Eleventh Computational Learning Theory* (pp. 80–91).

Appendix B

Publication (Wheway, 2001)

Wheway, V. (2001) *Using boosting to Simplify Classification Models.* In Proceedings IEEE ICDM'01 - International Conference on Data Mining 2001, San Jose USA, November 2001.

Using Boosting to Simplify Classification Models

Virginia Wheway
vlw04@uow.edu.au

School of Mathematics and Applied Statistics,
School of Information Technology and Computer Science
University of Wollongong
NSW 2522 Australia

Abstract

Ensemble classification techniques such as *bagging*, *boosting* and *arcing* algorithms have been shown to lead to reduced classification error on unseen cases and seem immune to the problem of overfitting. Several explanations for the reduction in generalisation error have been presented, with authors more recently defining and applying diagnostics such as *edge* and *margin* [2,3,7]. These measures provide insight into the behaviour of ensemble classifiers but can they be exploited further?

In this paper a four stage classification procedure in introduced, which is based on an extension of edge and margin analysis. This new procedure allows inverse sub-contexts and difficult border regions to be detected using properties of the edge distribution. Classification is made more robust as confounding within a dataset is removed.

The majority of classification techniques have not been adapted to detect contexts within a dataset and the generalisation error reported in studies to date is based on the entire dataset and can be improved by partitioning the dataset in question. The aim of this study is to move towards interpretability, and it is shown that by training on a subset of the original training data we gain simplicity of models and reduced generalisation error.

1 Introduction

The ability to classify unseen observations efficiently is a desirable property of many data mining algorithms. Numerous barriers may be present when an optimal classification model is being sought. In particular unavoidable dataset symptoms such as noisy data, outliers and 'fuzzy' boundaries inhibit a model's performance. Another such inhibitor is the presence of more than one context within a dataset. i.e. different sections of the dataset being classified according to significantly different models. If undetected, this results in increased generalisation error and a more complex model prone to overfitting. The undetected

Table 1: AdaBoost:M1 [6]

AdaBoost:M1 *Input:* n training instances $\mathbf{x}_i$ with labels y_i. M trials. Base learner, H.
Initialization: All training instances begin with weight $w_i^0 = 1/n$.
Repeat for M trials:

- Induce classifier, h_m, using weighted training data and H.
- ϵ_m = weighted error for h_m on the training data. If $\epsilon_m > 1/2$, discard h_m and stop boosting. (If $\epsilon_m = 0$, then h_m gets infinite weight.)
- Classifier weight, $\beta_m = \log(\frac{1-\epsilon_m}{\epsilon_m})$
- Re-weight training instances:
  if $h_m(x_i) \neq y_i$ then, $w_i^{m+1} = w_i^m/(2\epsilon_m)$
  else, $w_i^{m+1} = w_i^m/2(1-\epsilon_m)$

Unseen instances are classified by voting the ensemble of classifiers h_m with weights β_m.

presence of such subcontexts has previously been attributed to noise and deterioration in generalisation error. If such situations can be detected and either removed or accounted for in the final modelling stage, significant gains may be made in both model simplification and generalisation ability.
A classifier built by combining individual hypotheses to form a single classifier is known as an ensemble. Whilst there are many ensemble building methods in existence, this discussion focusses on the method of boosting which is based on a weighted subsampling of the training examples.
This paper presents the results from an empirical study on boosting and edge analysis. Section 2 introduces the theory behind the study, which is tested in Sections 3 and 4 on a selection of UCI [1] [1] datasets.

2 Boosting, Margin and Edge

Boosting is an iterative procedure which trains a learner over n weighted observations, beginning with each observation having weight $\frac{1}{n}$. At the $m+1$-th iteration, examples which were classified incorrectly at the m-th iteration have their weight increased multiplicatively so that the total weight on incorrect observations is equal to 0.5. The learning algorithm will be given more opportunity to explore areas of the training set which are more difficult to classify. Accuracy of the final hypothesis depends on the accuracy of *all* the hypotheses returned at each iteration and the method exploits hypotheses that predict well in more difficult parts of the instance space. Table 1 gives the details of the boosting algorithm and its weight update methodology.

2.1 Margin and Edge

Recent explanations as to the success of boosting algorithms have their foundations in margin and edge analysis [2,3,7]. The edge is defined for the i-th

[1] URL = http://www.ics.uci.edu/~mlearn/MLRepository.html

training observation as the total weight assigned to incorrect classes, whereas the margin is defined as the total weight assigned to the correct class minus the maximal weight assigned to any incorrect class.
More formally, assume the ensemble comprises a combination of base learners, each of which produce $h_m(\mathbf{x})$ at the m-th iteration. From $h_m(\mathbf{x})$, an error indicator function $I(h_m(\mathbf{x}_i) \neq y_i)$ may be determined. Let c_m represent the vote for the m-th hypothesis with $\sum_m c_m = 1$. Then, for the i-th observation:

$$edge(m,i) = \sum_{j=1}^{m} c_j I(h_j(\mathbf{x}_i) \neq y_i) \tag{1}$$

In its pursuit of correctly classifying 'harder' sections of the data space, boosting tends to 'balance' the edge i.e. the proportion of misclassifications for each observation becomes uniform as the number of iterations increases [3,8]. By utilising this property, a method for detecting noise has been introduced in [8] whereby noisy or difficult observations are detected via deviations in the overall edge distribution.
In Section 3, the notion of deviation from the main edge distribution is used as the basis of a procedure to detect clusters of observations behaving differently to the rest of the data (as opposed to detecting single outlying observations). By isolating these clusters and attempting to determine their collective structure, we may gain insight into areas of the input space which should be segregated or at the least, treated with caution.

3 A Four Stage Classification Process

3.1 Stage 1: Detect Obvious Outliers

In the noise study discussed in [8], plots of $edge(10,i)$ versus observation number are drawn as the first step in detecting noise or anomolies in the data. Identified via a significant deviation from a 'balanced' edge distribution, such observations can be examined and if justified, removed. If observations are removed from the original dataset, boosting trials must be re-run as changes in the dataset results in adjustment of the edge values for all other observations.

3.2 Stage 2: Detect Clusters via Edge Diagnostic Plots

The best edge measure for the task of detecting clusters of observations with unique properties is unclear. Measures are sought which will capture and differentiate behaviour deviant from the main 'core' of the data. Observations which are consistently classified correctly will have a low average edge and a low variance of the edge. Observations which are persistently misclassified will also have a low edge variance but high average edge. Groups of difficult observations which differ in structure from the majority of observations will have a high edge variance as boosting will alternate between correct and incorrect

classification. Observations which collectively behave in the same manner have similar mean and variance of their edge values, and will fall in a common location when edge variance is plotted against average edge.
More formally, the edge measures for $m = 10$ boosting trials are calculated as:

- $\hat{E}[edge(10,i)] = \frac{1}{10}\sum_{m=1}^{10} edge(m,i)$
- $\hat{Var}[edge(10,i)] = \frac{1}{10}\sum_{m=1}^{10}(edge(m,i) - \hat{E}[edge(m,i)])^2$

3.3 Stage 3 - Define a new Classification Problem to Differentiate Between Clusters

Analysis proceeds by separating the dataset into the clusters as appearing on the $\hat{Var}[edge(10,i)]$ versus $\hat{E}[edge(10,i)]$ plot. Cluster number is used as the class variable in a new classification problem. This may illuminate differences between the clusters and improve classification accuracy as confounding on clusters will be removed. This process goes beyond simply selecting observations which were initially misclassified or by choosing observations with a high proportion of misclassifications. It is highlighting observations which have a certain variance and mean structure, symptomatic of flipping between correct and incorrect classification as boosting proceeds.

3.4 Stage 4 - Retrain the Classifier on Subsets

Once clusters have been identified, training classifiers on individual clusters will result in more generalisable models over these partitions. The difficulty arises for unseen observations when cluster membership is unknown. If the classification procedure in Section 3.3 fails to discriminate between clusters, we proceed by assuming all new observations are best modelled according to the classifier trained on the larger partition. This notion is shown to lead to reduced classification error as demonstrated in Section 5.

4 Demonstration on the UCI *colic* dataset

The four-stage edge based procedure is now demonstrated on the *colic* dataset. (For more detail on this and other UCI datasets tested in this study, refer to Table 3.) Decreased generalisation performance has been observed on the *colic* dataset when boosting is applied [8]. The four stage classification process suggested in this paper sheds some light on this phenomenon.

4.1 Stages 2 and 3 : Formulation of a new Classification Problem

After detecting and removing an obvious outlier using the methodology from [8], a plot of $\hat{Var}[edge(10,i)]$ versus $\hat{E}[edge(10,i)]$ is drawn and shown as Figure 1.

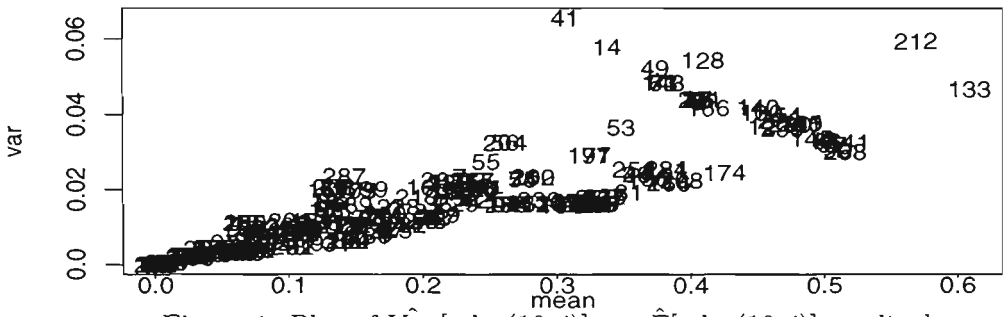

Figure 1: Plot of $\hat{Var}[edge(10,i)]$ vs. $\hat{E}[edge(10,i)]$: *colic* data.

Two clusters are evident in Figure 1: one cluster being those observations exhibiting a positive relationship between $\hat{Var}[edge(10,i)]$ and $\hat{E}[edge(10,i)]$ and the others a negative one.

A new dataset named *colicflag* is created for which a new response variable is appended to the *colic* dataset. This new response, $edgeflag$ is assigned a value of 1 if an observation lies in the upper right hand cluster of Figure 1 and 0 otherwise. This creates a new classification problem with $edgeflag = 0/1$ as the new response variable. If a classifier can find repeatable structure in differentiating $edgeflag = 0$ from $edgeflag = 1$ using the available predictor variables, then it may be concluded that such structure forms boundary regions of uncertainty, or differentiates between two contexts within the dataset. Because an aim of this study is interpretability, single trees are fitted as opposed to boosted ensembles, since boosted models become un-interpretable very quickly.

Several unsuccessful attempts to classify $edgeflag$ using only the initial predictor variables were made. Only when the original response is included as a predictor did a concise, highly accurate classifier result. The original response was returned as the initial split on the decision tree, indicating different mechanisms occurring for different values of the initial response, given identical input contributions. The resulting decision tree had subtrees beyond the initial split which were inverse to each other.

This may seem intuitive as observations which are difficult to classify will naturally be predicted oppositely the main 'core' of data. However, in this and other UCI datasets tested we see that each cluster has equivalent input ($\mathbf{x}_i$) values but reversed y_i (class) values.

4.2 Stage 4 - Retrain the Classifier on a Majority Subset of the *colic* Dataset

Using the $edgeflag$ variable defined in Section 4.1, the *colic* dataset is split into two distinct datasets (*horsec0* and *horsec1*). Although we require the original target variable to do this, the important property if inverse sub-contexts is demonstrated.

The initial binary classification problem may then be re-estimated for each sub-dataset (*horsec0, horsec1)* using the input predictors. For interpretability, a *c*4.5 decision tree using default values was used as the classification method. The results are encouraging with training error for the *horsec0* dataset being only 0.38% (as opposed to 14.99% on the entire *colic* dataset). Refer to Table 2 for detail of generalisation performance on the *colic* dataset. The figures presented in the final column refer to the average generalisation error estimated via 10-fold cross-validation , with bracketted figures giving the standard error of this estimate. The large standard errors evident for the *horsec1* dataset are based on test sets of 3-4 observations and may not be representative. Because

Table 2: Cross-validation results for *colic* dataset.

dataset	**response**	**method**	**gen. error**
colic	Had surgery (y/n)	*c*4.5 single	14.99(8.76)
colic	Had surgery (y/n)	*c*4.5 10 boost	21.01(7.21)
colicflag	edgeflag (0/1)	*c*4.5 single with original response	0.34(1.08)
horsec0	Had Surgery (y/n)	*c*4.5 single	1.53(2.68)
horsec0	Had surgery (y/n)	*c*4.5 10 boost	0.38(1.20)
horsec1	Had surgery (y/n)	*c*4.5 single	3.33(10.25)
horsec1	Had surgery (y/n)	*c*4.5 10 boost	3.33(10.25)

the decision trees are fitted on a reduced set of training data, the generalisation error on these partitions will not be representative of the generalisation error for the entire dataset. Section 6 outlines logic for a new generalisation error estimate which encompasses error from both partitions of the dataset.

5 Generalisation Error Calculations

Because it is proposed to use the simpler model trained on the majority partition for future prediction, generalisation error must be re-defined. Assume the proportion of observations falling into the main cluster is given as p. $(0 < p < 1)$. Let e_1 represent the estimated generalisation error for the simple classifier trained on this cluster. Now, since the remainder of the data is modelled in an inverse fashion to the main cluster, the generalisation error for this section of the data will at worst be estimated at $1 - e_1$. Denoting the new overall generalisation error estimate as $e_{cluster}$, we have:

$$e_{cluster} \approx (p * e_1) + (1 - p) * (1 - e_1) \tag{2}$$

Table 3 gives a description of the UCI datasets used in this study. Unless otherwise described for a particular dataset, 10 boosting iterations were applied using *c*4.5 with default options as the base learner. Calculations of this generalisation error after applying the four-stage classification process from Section 3 for a selection of UCI datasets are presented in Table 3. Note: $\mathbf{e_{init}}$ refers to the generalisation error as estimated on a 10 iteration boosted classifier trained

Table 3: Summary of UCI datasets used in this study - as per [9] plus new generalisation error calculations

Dataset	Cases	Classes	Continuous Attributes	Discrete Attributes	e_{init}	$e_{cluster}$
colic	368	2	10	12	21.01	14.40
credit	690	2	7	13	18.24	12.17
heart-h	294	2	8	5	22.61	15.99
heart-c	303	2	8	5	18.41	16.64
hepatitis	155	2	6	13	16.00	14.98

on the entire dataset.
For the *colic, credit* and *hearth-h* datasets, this simplification of models results in an impressive reduction in generalisation error estimates. For the *heart-c* and *hepatitis* datasets, reduction of generalisation error occurs but is less dramatic.

6 Discussion

In many cases, the $\hat{Var}[edge(10,i)]$ versus $\hat{E}[edge(10,i)]$ procedure identifies only those observations which were initially misclassified. Although discouraging, this phenomenon brings the following points to light:

- The usual practice of applying boosting to the entire dataset gives difficult observations an opportunity to be modelled correctly. It was seen for the datasets tested, that even after 20 boosting iterations, the boosted ensemble was no closer to making consistent predictions on these groups of observations. The variance of the edge also remains large as boosting flips between correct and incorrect classification on these observations.

- Closer inspection revealed these observations to occupy a similar region in predictor space to that of the 'core' data but the class labels are reversed. This could be attributed to mislabelling of the output or the absence of a relevant predictor variable.

- In all datasets tested, the initial decision tree trained on the entire dataset was significantly more complex in its attempt to accommodate for the observations falling into the smaller cluster. A very simple robust tree with low generalisation error results when $c4.5$ is trained on the main cluster only. Using the tree trained on a subset of the data does not significantly deteriorate generalisation performance.

- Boosting is often successful within the 'core' data cluster, even when unsuccessful on the entire dataset.

7 Conclusion

This study has proposed a new multi-stage classification process which not only improves generalisation error but also results in easily interpretable classification models. In all datasets tested it was seen that a simple, highly accurate model built on the majority of the data was preferable to a more complex model trained on the entire dataset. The more complex model fitted to the entire dataset was a result of the need to accommodate noise and difficult observations. This additional detail was seen to cloud the underlying core classification model. It was also proposed that the UCI datasets tested contain sections of mislabelled data, or are lacking the measurement of a key predictor variable.
The clusters identified from the edge diagnostics are usually unable to be discriminated without the use of the initial class variable. However, if the simple tree trained on the majority cluster is applied to the entire dataset, the overall misclassification rate is reduced, sometimes substantially.
Further work will concentrate on automatic cluster detection and testing other possible edge measures across a varying number of boosting iterations and datasets.

References

[1] Blake, C.L. & Merz, C.J. (1988). UCI Repository of Machine Learning Databases. University of California, Irvine, Dept. of Information and Computer Sciences.

[2] Breiman, L. (1997). *Arcing the edge* (Technical Report 486). Statistics Department, University of California, Berkeley.

[3] Breiman, L. (1999). *Random Forests - Random Features* (Technical Report 567). Statistics Department, University of California, Berkeley.

[4] Freund, Y., & Schapire, R. E. (1997). A decision-theoretic generalisation to on-line learning and an application to boosting. *Journal of Computer and System Sciences , 55(1)*, 119–139.

[5] Quinlan, J. R. (1993). *C4.5: Programs for Machine Learning*. Morgan Kaufmann.

[6] Quinlan, J. R. (1996). Bagging, boosting and C4.5. *Proceedings of the Thirteenth National Conference on Artificial Intelligence*. (pp. 725–730). Menlo Park California, American Association for Artificial Intelligence.

[7] Schapire, R. E. , Freund, Y., Bartlett, P. & Lee, W. S. (1998). Boosting the margin:a new explanation for the effectiveness of voting methods. *Annals of Statistics, 26(5)*, 1651–1686.

[8] Wheway, V. L. (2000). Using Boosting to Detect Noisy Data *To appear in Lecture Notes in Computer Science 2112:AIApps 2000* Springer