Learning From Large Data: Bias, Variance, Sampling, and Learning Curves

Damien Brain, B.Sc.

Submitted in fulfilment of the requirements for the degree of
Doctor of Philosophy

Deakin University
October, 2003

Abstract

One of the fundamental machine learning tasks is that of predictive classification. Given that organisations collect an ever increasing amount of data, predictive classification methods must be able to effectively and efficiently handle large amounts of data. However, it is understood that present requirements push existing algorithms to, and sometimes beyond, their limits since many classification prediction algorithms were designed when currently common data set sizes were beyond imagination.

This has led to a significant amount of research into ways of making classification learning algorithms more effective and efficient. Although substantial progress has been made, a number of key questions have not been answered.

This dissertation investigates two of these key questions. The first is whether different types of algorithms to those currently employed are required when using large data sets. This is answered by analysis of the way in which the bias plus variance decomposition of predictive classification error changes as training set size is increased. Experiments find that larger training sets require different types of algorithms to those currently used. Some insight into the characteristics of suitable algorithms is provided, and this may provide some direction for the development of future classification prediction algorithms which are specifically designed for use with large data sets.

The second question investigated is that of the role of sampling in machine learning with large data sets. Sampling has long been used as a means of avoiding the need to scale up algorithms to suit the size of the data set by scaling down the size of the data sets to suit the algorithm. However, the costs of performing sampling have not been widely explored. Two popular sampling methods are compared with learning from all available data in terms of predictive accuracy, model complexity, and execution time. The comparison shows that sub-sampling generally produces models with accuracy close to, and sometimes greater than, that obtainable from learning with all available data. This result suggests that it may be possible to develop algorithms that take advantage of the sub-sampling methodology to reduce the time required to infer a model while sacrificing little if any accuracy.

Methods of improving effective and efficient learning via sampling are also investigated, and new sampling methodologies proposed. These methodologies include using a varying proportion of instances to determine the next inference step and using a statistical calculation at each inference step to determine sufficient sample size. Experiments show that using a statistical calculation of sample size can not only substantially reduce execution time but can do so with only a small loss, and occasional gain, in accuracy.

One of the common uses of sampling is in the construction of learning curves. Learning curves are often used to attempt to determine the optimal training size which will maximally reduce execution time while not being detrimental to accuracy. An analysis of the performance of methods for detection of convergence of learning curves is performed, with the focus of the analysis on methods that calculate the gradient of the tangent to the curve. Given that such methods can be susceptible to local accuracy plateaus, an investigation into the frequency of local plateaus is also performed. It is shown that local accuracy plateaus are a common occurrence, and that ensuring a small loss of accuracy often results in greater computational cost than learning from all available data. These results cast doubt over the applicability of gradient of tangent methods for detecting convergence, and of the viability of learning curves for reducing execution time in general.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The Information Revolution has brought with it the ability for organisations to collect and store massive amounts of data. The justification for this collection is that the data can potentially be used to improve the organisation. However, data has little utility when kept in its raw form. For data to be usable it must be transformed into information.

Many methods of transforming data into information exist, ranging from simple analysis such as finding the mean of a collected measurement through to building complex models describing the data. A common way of extracting information from data is with machine learning. One of the more popular uses of machine learning is for predictive classification, in which models of the data are built such that an unmeasured discrete attribute of previously unseen data can be predicted based on values of measured attributes. Examples of uses of predictive classification include detection of credit card fraud, medical diagnosis, and cosmological classification.

The prevalence of such uses is increasing. It can be expected that data collection will become more widespread and that collected data set sizes will increase. Classification learning algorithms must, therefore, be able to effectively and efficiently deal with large data sets. However, many algorithms were designed when data set size was smaller than it currently is, and potentially much smaller than in the future. Techniques to minimise this mismatch in scale are required. This thesis investigates a number of methods by which this may be done.

1.1 Background to the Research

Much research has been performed into ways of dealing with large data sets. This research can be generally categorized into two sets of approaches. The first set contains those approaches that attempt to introduce new algorithms or modify existing algorithms to better suit large data sets. This is known as "scaling up" algorithms, with many of these approaches based on parallelising learning algorithms. The problem with this set of approaches is that scaling up an algorithm so that it can deal with data set sizes of today runs the risk of becoming obsolete tomorrow. Since data set size will continue to grow, algorithms will continue to require scaling up to deal with larger data sets.

The second set of approaches attempt to "scale down" data sets so they are more practical to use with existing algorithms. This is often done by reducing the size of the data set, resulting in an increase in the risk of overlooking valuable information. It may be argued that, in practice, scaling down only increases this risk for information that is not frequently represented in the data set, with the implication that such low-frequency information is relatively unimportant. However, information can be valuable precisely because it is rarely evidenced by the data. Thus, scaling down may preclude such information being found.

Methods for scaling up algorithms and scaling down data sets have certainly not been exhaustively explored. There is great scope for research in both areas. This aim of this thesis is to add to the body of research on ways of dealing with large data sets in classification learning.

1.2 Research Objectives

The objectives of the thesis are to investigate ways in which classification learning algorithms can be enhanced for use with large data sets. The objectives are to:

1. Investigate whether different types of algorithm are required when learning from large as opposed to small data sets. Since many classification learning algorithms were designed when data sets were much smaller than present, it is plausible that they are predisposed to perform better when learning from smaller rather than larger data sets.

2. Provide a rigorous comparison of popular sampling methodologies. Given the prevalence of sampling, it may be useful for practitioners to know how pre-sampling and sub-sampling compare in terms of accuracy, model complexity, and execution time. As yet, no such comparison exists.

3. Investigate alternative means of selecting sub-samples. Two strategies often used when selecting samples are disproportionate sampling and non-replacement. However, the effects of these within a sub-sampling context has not been researched. Excluding these strategies may potentially reduce execution time, without significant detriment to predictive accuracy.

4. Investigate alternative means of determining sub-sample size. Sub-samples are often taken with a fixed, pre-determined sample size. This implies that the number of instances from which a sub-sample is selected is unimportant. It is reasonable to believe that such an assumption is untrue. Methods of determining sub-sample size based on available data should therefore be explored.

5. Investigate the viability of learning curves. Learning curves are often used with large data sets to determine the optimal training set size with which to

learn. However, it is believed that the viability of learning curve methods has not been properly studied. Although learning curve methods are shown to often reduce execution time, the conditions under which this can be expected to occur are not known.

6. Investigate the reliability of learning curve methods that estimate the gradient of the tangent to the curve. Such methods may be susceptible to local accuracy plateaus. Much research has implicitly shown that these plateaus exist, but their effect on various learning methods has not been explored.

1.3 Principal Outcomes of the Thesis

The principal outcomes of the thesis are as follows:

1. As data set size increases, bias becomes the more dominant part of the error, regardless of the expected level of bias management and variance management of the algorithm. This suggests that algorithms that deal with large data sets should focus on bias management rather than variance management if the potential information in the data is to be fully realised. (Section 3.1)

2. An analysis is provided on the causes of bias and variance. (Section 3.2.1)

3. It is shown that decision tree grafting acts primarily by reducing variance, and becomes less effective for reducing bias as training set size increases. (Section 3.2.3)

4. It is shown that reducing representational bias results in a decrease in bias, but does not consistently lower error. (Section 3.2.3)

5. Sub-sampling is shown to be a much more accurate method of sampling than pre-sampling. Sub-sampling generally produces models with accuracy close to that obtainable from the full training set with small sample sizes, and sometimes induces models with accuracy greater than no sampling. However, sub-sampling results in much longer execution time than pre-sampling. (Chapter 4)

6. Disproportionate sampling and sampling with replacement affect the accuracy of induced models and the time required for induction when sub-sampling. Non-disproportionate sampling is shown to result in a greater loss of accuracy and smaller decrease in execution time than sampling with replacement. (Section 5.2)

7. Determining sub-sample size by using a variable proportion of instances is shown to reduce accuracy and execution time. (Section 5.3)

8. Determining sub-sample size using a statistical measure of sample quality can result in substantial time savings without loss of accuracy. (Section 5.4)

9. Evaluation of the statistical quality of a sample has been extended to continuous attributes. (Section 5.4.4)

10. Learning curves are often not a viable method for determining optimal sample size, as they can be less efficient than learning from the full training set. (Section 6.2)

11. Learning curve methods that detect convergence by analysing the gradient of the tangent to the curve can lead to large loss of accuracy by finding local accuracy plateaus, rather than the global accuracy plateau. (Section 6.3)

1.4 Software Developed

C4.5 Release 8 was the basis of most of the software used in this research. The following modifications have been implemented to enable the experiments detailed in this research to be performed:

1. Extended to enable pre-sampling.
2. Extended to enable sub-sampling.
3. Extended to allow different types of sub-sampling to be performed. This included options for sampling with and without replacement, and with and without disproportionate sampling.
4. Extended to evaluate Variable Proportion Sampling and Statistical Quality (SQ) sampling.

The OC1 package was also extended to enable collection of results regarding the bias plus variance decomposition of error.

1.5 Research Methodology

The nature of this research demands the use of empirical evaluation of hypotheses. Experiments are performed, and the results analysed. Statistical hypothesis tests are used where appropriate.

1.6 Structure of the Thesis

The thesis is structured as follows.

Chapter 2 is a review of literature relevant to the thesis. Problems with applying classification learning algorithms to large data sets are outlined. Approaches to remedy these problems are discussed, and reasons why they may not meet expectations shown.

Chapter 3 details an investigation into how the bias plus variance decomposition of classification error is affected by increased training set size. Experiments regarding this decomposition of error are performed, as are experiments into how the performance of two bias reduction methods varies with larger training sets.

Chapter 4 presents experiments investigating the effects of different types of sampling on the performance of classification learning algorithms. Performance is compared in terms of predictive accuracy, model complexity, and execution time.

Chapter 5 investigates modifications to the sub-sampling methodology used in Chapter 4. These modifications are designed to reduce execution time of sub-sampling without detrimentally affecting accuracy. Two new sub-sampling methodologies are proposed and evaluated.

Chapter 6 discusses the viability of using learning curves with large data sets. An analysis is made of the computational requirements of the current best learning curve algorithm. Experiments into the reliability of the accuracy achievable by this and similar algorithms are performed.

Chapter 7 presents conclusions and re-iterates the main findings of the thesis.

Chapter 2

Review of Relevant Literature

This chapter presents a review of literature relevant to the thesis. The focus of this review is on the impact of large data sets on supervised induction of classification prediction models.

The review is structured as follows. Section 2.1 outlines problems that induction methods encounter when faced with large data sets. Section 2.2 reviews research into ways these problems can be mitigated.

2.1 The Problem With Large Data Sets

Classification prediction algorithms are often applied to large data sets (e.g. [37, 18, 23]). However, since many classification learning algorithms were designed when common data set sizes were much smaller than they are now, there is often a mismatch in scale. For example, the Nearest Neighbor algorithm has existed since at least 1967 [24], and the CART decision tree induction algorithm was published in 1984 [13]. At such times, data sets of terabyte size (e.g. [37]) were not only well out of the range of technical possibilities, but were likely not taken into account in the design of algorithms. The machine learning community has since found that this mismatch of scale can lead to significant problems in the application of classification learning algorithms to large data sets. These problems are discussed below.

2.1.1 Execution Time and Computational Complexity

Possibly the most obvious problem that can occur when applying classification learning algorithms to large data sets is that of execution time. Increasing data set size will likely require an increase in the amount of processing required, and hence lead to longer execution time. However, the problem is compounded by the fact that the computational complexity of classification learning algorithms is generally greater than linear in the number of training instances [29].

For example, the computational complexity of C4.5 has been measured at between $\mathrm{O}(n^{1.22})$ and $\mathrm{O}(n^{1.38})$, where n is the number of instances in the training set [84]. The variation in complexity is due to the data set used. However, even with the lower of these complexities, a large increase in training set size can be expected to have a severe impact on execution time.

2.1.2 Memory Requirements

A second problem that can occur when using very large data sets is the excessive memory requirements of some algorithms. Many algorithms require all training instances be retained in main memory (e.g. [88, 64, 15]). Therefore, as data set size increases, so does the required memory size. Classification learning algorithms also require memory for conducting searches, holding partial solutions, bookkeeping, etc. These ancillary requirements alone can amount to a substantial memory requirement.

The problem arises when the total amount of memory required by an algorithm exceeds the primary memory available. If this occurs, the operating system starts swapping — moving some of the memory allocated to a process from main memory to disk, thus freeing space in main memory. When memory that has previously been swapped to disk is again required, it is read from disk back into main memory. Although swapping increases the effective memory size of computers, it comes at a cost of degraded performance, as swapping is comparatively a very slow process. Access times times for main memory are measured in nanoseconds, while disk access times are measured in milliseconds — a factor of one million times slower. Thus, once swapping starts to occur, execution times can increase greatly.

Other algorithms require only a portion of the data to be resident in memory at any one time (e.g. [13]). However, as data set size increases, the proportion of data residing in memory relative to the total data set size must become smaller given a fixed maximum memory capacity. This may lead to difficulty in detecting rare concepts in the data.

2.1.3 Efficient Data Access

A third aspect important to effective handling of large data sets is the timely and efficient retrieval of data [93]. An algorithm that cannot obtain data when required will certainly not execute as quickly as otherwise possible. This is especially true for incremental algorithms, as they may continually request more data during the induction process. Therefore, to minimise execution time of induction algorithms, data should be organised and stored in a manner such that timely access by an algorithm is likely. However, although this problem affects the use of learning algorithms, it falls outside the scope of this research.

2.1.4 Model Complexity

The final problem of large data sets is the effect they have on the complexity of inferred models. It is often desirable for induced models to be humanly understandable, and thus less complex models may be preferred. However, increasing training set size often results in an increase in the complexity of the model without a corresponding increase in model accuracy [80, 81]. Therefore, learning from large data sets may result in models that are more complex than necessary and overfit the data.

This is counteracted by the finding that the more general a model, the less likely it is to classify unseen instances with accuracy similar to that observed on the training set [102]. When comparing models inferred using the same algorithm with the same data set, it can likely be thought that the more specific the model, the more complex it is. For example, within a decision tree context, a model is made more specific by adding nodes, which in turn increases complexity. Thus, to have confidence in the performance of a model on unseen data, a practitioner may prefer a more complex model. However, this complexity should not increase to a point where overfitting begins to detriment performance.

2.2 Approaches to Managing Large Data Sets

Many attempts to counteract the problems discussed in the previous section have been made. Collectively, they fall into two categories: those that "scale up" algorithms to handle large data sets, and those that "scale down" data sets to a practical size. Good general surveys of such methods are provided by Provost and Aronis [86] and Provost and Kolluri [85].

2.2.1 Scaling Up

Parallelising

The purpose of parallelising methods is to split the computational requirements of a learning algorithm between many processors. An algorithm can be parallelised in two ways — data parallel or task parallel [19]. In a data parallel scheme, data is partitioned so that multiple processors perform the same task on different data subsets. The results generated by each processor are then combined. Two implementations of data parallel schemes are discussed below. Task parallel schemes distribute different parts of the learning algorithm across processors. In this situation, each processor has a copy of the data, but performs a different task. A number of methods claim to have implemented task parallel schemes for decision trees by inducing subtrees on different processors (e.g. [19, 25, 106]). However, this is a data parallel, not a task parallel approach, as the same procedure is being performed by each processor [56].

Provost and Aronis' parallel approach [86] is a data parallel scheme. In these experiments, sequential and parallel versions of the same algorithm are compared. A model is developed, and then evaluated over the training set. The result of the evaluation is used to accept or reject changes to the model. A change is accepted if it reaches a minimum threshold of support and is below a maximum threshold of counter-support. In the sequential approach, all evaluations are performed using the whole training set on a single processor. The parallel approach distributes the training set across a number of processors: a single processor is used to develop and modify the model, but multiple processors are used to evaluate the model.

The speed improvement reported using the parallel approach is substantial. Execution times are reported for different training set sizes using both approaches, but results for the sequential approach are given to only approximately 7% of the total data set size as execution times became so large that experiments with larger training set sizes were infeasible. Execution times for the parallel approach remained very small, even with the full training set. Although these results are impressive, they are hardly surprising since the parallel approach used 8,192 processors. Using so many processors can clearly have huge advantages, but may not always be practical.

Others have attempted data parallel schemes using agent architectures [62]. Agents offer the advantage of using multiple processors to perform a task, but also allow for a more coordinated approach to the problem through a certain amount of “intelligence” built into each agent.

However, agents still suffer from the same problems as standard data parallel approaches, plus an extra, often overlooked problem. One of the benefits of agents is that they are designed to run over a network, and are therefore distributed by nature, but must therefore also incur the communication overheads and other associated problems involved with networks. This means that, due to the volatile nature of networks, agent architectures cannot rely on timely, or even eventual communication between agents. Thus, although an agent may find the “perfect” solution, it is possible the agent will not be able to communicate this fact to the rest of the agent architecture.

Chattratichat et al. [19] have investigated which of task and data parallel schemes are more applicable for classification algorithms. Their results suggest that the best strategy depends highly on properties of the particular data set used. In some instances, the results show that parallel methods can even have longer execution times than standard sequential algorithms, illustrating that parallelised algorithms are not always appropriate for machine learning with large data sets. When data and task parallel schemes should be employed has been further explored [25], with the conclusion drawn that the decision is difficult and dependent on the task at hand and the hardware available.

Simple and Fast Algorithms

As mentioned previously, algorithmic speed is a major factor in the usefulness of an algorithm when dealing with very large data sets, and common machine learning algorithms can take prohibitively long to execute [29]. However, this is not always the case. Algorithms such as the Naïve-Bayes classifier [70] can be very computationally efficient, and hence quite quick. The drawback of such methods, though, is that they usually sacrifice accuracy to obtain this efficiency

[86].

Kohavi's NBTree algorithm [64] attempts to find a middle ground between efficiency and accuracy by combining accurate but inefficient decision trees with Naïve-Bayes classifiers. This could be expected to produce an algorithm that retains the Naïve-Bayes efficiency, but produce accuracy comparable to the more complex decision trees. Unfortunately this is not the case. Also, the results presented in this paper do not show an improvement in execution time against normal decision tree algorithms.

Lazy techniques (e.g. [24, 44, 107]) can significantly reduce the amount of time required to build a model. Rather than requiring a generalised model to be built from a training set, lazy techniques build a specialised model for each test instance. This has the advantage that paths that are irrelevant to the current test instance do not need to be explored, and can significantly reduce execution time required when the number of test instances is small However, since each test instance requires the induction of a new model, classifying a large number of test instances can be computationally expensive. Studies have shown that classifying the same large number of test instances with a lazy technique generally takes substantially longer than with an equivalent non-lazy technique. Although lazy techniques are affected little by large training sets, it is not unreasonable to expect that a large training set will be accompanied by a large test set. This precludes lazy techniques from use with large data sets.

Optimisation

The objective of algorithm optimisation is to minimise the resource and computational costs of a specific algorithm. This may include using efficient data structures, optimising code, or performing a task in a more efficient way.

The third of these approaches is taken in the CWS algorithm [29]. Instead of testing the performance of separate parts of a model (in this case, classification rules), CWS tests the performance of the model as a whole. Changes to parts of the model can be explored, but are only accepted if they improve the performance of the whole model. The reported predictive performance of CWS is comparable to that of other systems (i.e. C4.5rules [88] and CN2 [21]), and execution times given show that CWS is much quicker than both C4.5rules and CN2. Unfortunately, these results appear unreliable as the environment in which each algorithm was tested was inconsistent. For example, footnote 5 states that “the program crashed due to lack of memory. This may be due to other jobs running concurrently.” However, CWS may be fastest regardless, as the difference in execution times between CWS and CN2 was quite substantial. Also, an analysis of computational complexity shows CWS to have complexity O(n), compared to O(n^4) for C4.5rules, raising the expectation that CWS should be fast compared to other classification rule induction algorithms. CWS has not yet been shown as applicable for very large data sets, though, as the largest data set employed contained only 50,000 instances. However, the aforementioned computational complexity suggests that execution time should not be a significant negative factor on larger data sets.

Methods such as SPRINT [93] and RainForest [46] eliminate the need for instances to be stored in main memory by maintaining summary statistics of attributes. However, the maintenance of these statistics runs into problems when a split is performed, as the statistics must also be partitioned between child nodes. SPRINT does this by associating record identifiers with the statistics, so they can be partitioned correctly after a split. A hash table is required for this step, and can itself be too large to fit in main memory [93]. RainForest proposes that statistics for child nodes be collected as data is partitioned, but does so by reading all instances from disk at each node. This is clearly inefficient. The ADTree method [75] also maintains summary statistics, but requires that both the ADTree data structure and training instances are kept in main memory.

Efficient C4.5 [90] implements the RainForest framework as one of three optimisations to C4.5. However, results of reported experiments show that these optimisations can actually increase memory requirements.

Other optimisations speed up the way in which specialisation of hypotheses is performed. Breadth-first Marker Propagation [3] reduces complexity by the average number of values per attribute by generating counts of the number of instances that match all specialisations of a rule in one pass through the data. However, the method may not be beneficial for algorithms, such as decision trees, that separate data into different sub-sets based on the current model. For example, the first three steps of Breadth-first Marker Propagation are irrelevant in a decision tree context as they have already been effectively performed. The fourth step is also, in effect, the same as is then performed in a decision tree to create tallies of attribute values.

2.2.2 Scaling Down

An alternative to modifying algorithms so they can better handle large data sets is to make large data sets more usable by existing algorithms. This is the premise behind methods discussed in this section. These methods involve relaxing constraints on induced models, reducing the complexity of attributes, and reducing the number of instances in the training set.

Reducing Attribute Complexity

One of the more computationally expensive tasks of a learning algorithm is sorting of continuous values. Removing this requirement can increase efficiency greatly, and can be performed by discretising continuous attributes. This has been studied often [89, 39, 65, 33, 105, 6, 5, 55, 14], with the method of Fayyad and Irani [38] often shown to result in models with the lowest predictive error.

Although substantial execution time can be saved by discretising continuous attributes prior to invoking the learning algorithm, there is also a cost to be paid in the loss of information involved in discretisation. Many of the above mentioned discretisation methods determine discretisation intervals by analysing one attribute at a time. This can result in important inter-relations between attributes being hidden or removed from the discretised data set [5]. This may not only harm potential accuracy of the model, but may also reduce the understandability of the model. Thus, pre-discretisation, while saving execution time, also has drawbacks.

Rather than discretising continuous attributes, an alternative is to eliminate unnecessary attributes [72]. Irrelevant attributes not only increase search space, and hence execution time, but can also harm predictive accuracy [69].

Such attribute selection methods fall into two categories — wrapper and filter methods. Wrapper methods create multiple models using differing subsets of features, selecting the model with the highest accuracy. Obviously, this method comes at the cost of increase in execution time. Filter methods analyse the data set to determine which attributes are least useful for inducing a model. This is performed using measures such as distance and dependence [72]. However, induction algorithms that do not necessarily include all features in learned models are filter methods in themselves [63]. These type of algorithms include decision trees and decision rules.

Sampling

The idea behind sampling methods is to use only a portion of the available data to extract the required information. This enables the learning algorithm to perform its job while only having to deal with a fraction of the data that would normally be used.

Sub-sampling [13] begins by selecting a portion of the available training instances into a special training set: the sub-sample. From the sub-sample, a decision about the next step for the machine learning algorithm to take is made. No instances are added to or deleted from the sub-sample. Therefore, to be confident of making a “good” decision, it is essential that the instances within the sub-sample are sufficiently representative of the whole data set, as no other instances are used to verify this assumption. In other words, sub-sampling does not use the rest of the training set to ensure that the decision made is indeed a valid decision. It would then seem logical that a good proportion of training instances should be included in the sub-sample to guarantee that satisfactory decisions are made. However, as sub-sample size increases, sub-sampling becomes less and less advantageous. This creates a trade-off between the utility of sub-sampling and the representativeness of the sub-sample. Establishing an appropriate balance between the two can be difficult. An advantage of sub-sampling, though, is that only instances in the sub-sample are kept in main memory — the rest can remain on disk. Thus, the memory requirements of sub-sampling are much smaller than many other machine learning algorithms.

CART uses a form of disproportionate sampling to select instances to include in a sub-sample. Instances are selected so that the number of instances representing each class in the sub-sample are as even as possible. Alternative methods of selecting samples include: selecting instances without replacement and without regard to class distribution, so that every instance has equal chance of being in the sample; stratification, so that the proportion of instances of each class in the data is maintained in the sample; undersampling, where fewer instances from well represented classes are selected; and oversampling, where instances from under-represented classes are replicated [74].

Windowing [88] begins, like sub-sampling, by selecting a portion of the training instances — the window. Using this window, a model of the data is built. The rest of the training data (i.e. instances not in the window) are used to validate the model. Unlike sub-sampling, instances can be added to the window. This is done by including a portion of the instances (at least half) that do not fit the model into the next window. The process then starts again. This continues until either all instances fit the model, or improvements are no longer being made. Windowing suffers a similar problem to sub-sampling, however. If the initial window is small, the model generated will likely not fit the data well. Therefore, a large proportion of instances will be included in the subsequent window. Using a large initial window defeats the purpose of windowing. Another problem with windowing is that multiple models are usually built. Although building many models using smaller sized data sets may offer speed advantages, it is far from optimal. Research has suggested that windowing may be more applicable to decision rule induction than decision tree induction [45].

Catlett's peepholing approach [16] is more complex than sub-sampling and windowing. Instead of just decreasing the number of instances to be searched when trying to make a decision, peepholing also attempts to decrease the number of attributes and the range of possible values of these attributes to search. These reductions are performed during induction, and are designed specifically for decision trees.

Peepholing is performed in two parts: short-listing and blinkering. Short-listing involves restricting the number of attributes to be searched by estimating the optimistic and pessimistic information gain [88, 16] of each attribute. All attributes whose optimistic gain is less than the best pessimistic gain are scratched, or removed from consideration. Subsets of data are used to estimate the information gain of attributes.

Blinkering restricts the range of an attribute's possible values to be searched. Again, a subset of data is used, with an initial size of 300 instances. Blinkering is performed in iterations, where each iteration eliminates a number of values to be searched by progressively narrowing a range in which the optimal value is presumed to lie. The subset size is increased every iteration by a factor of three.

There are a number of problems associated with peepholing. Like windowing, the sample of data used can grow. However, the subset is initially quite small. Given a training set of millions of instances, a subset of 300 instances is a very low proportion. It may be over-zealous to exclude attributes or ranges of values at such an early stage. A second problem is the computational complexity of peepholing. Catlett [16] claims that peepholing has the computational complexity O(n) *per instance*. When viewing the data set as a whole, this is a computational complexity of O(n^2).

John and Langley proposed a measure of determining when a sample is large enough to adequately represent a data set [60], describing the Probably Close Enough (PCE) framework for governing sample size. A sample is considered sufficiently large if the probability of a loss of accuracy greater than a user-specified limit is less than a user-specified probability bound. However, PCE requires a learning algorithm whose data set size can change dynamically. Not all machine learning algorithms meet this requirement.

One method of parallelising a machine learning algorithm, mentioned above, is to split the data set into many disjoint subsets, then analyse each subset on a different processor. Batching takes a similar approach, except instead of using many processors, partitions are analysed sequentially on a single processor.

Chan and Stolfo's partitioning approach [17] separates a data set into subsets. A base model is then induced from each subset. Results from each base model are combined in either of two ways suggested. The first way is to use the results produced by each model to generate a new and final model. This model is then used to produce the output of the system. The alternate method decides the system output by a majority vote. Ties are broken by using an arbitration rule, along with a trained arbiter. The arbiter is trained using "confusing" instances as its training set. Confusing instances are classed as those for which the number of base classifiers that agree on the output does not form a majority.

The evidence given to support the utility of these approaches is not entirely convincing, however. Many of the results presented show a substantial drop in performance when compared to a non-partitioned data set. Experiments were only performed on two data sets, with a maximum of 20,000 instances — not a size that could generally be considered a very large data set. No information about execution time is given, and hence a full comparison into the efficiency of the algorithm is not present. However, given that the computational complexity of the base model inducers is superlinear, it is reasonable to expect a substantial reduction in execution time. This reduction should also increase with division of data into more partitions.

Partitioning is also used with the RISE rule induction system [30], where it is shown to substantially reduce execution time. However, these experiments are again performed with relatively small training sets — the largest containing 43,500 instances.

Pasting [11] is a method similar to that of partitioning, in that both build a number of models with small training sets. There are two important differences between pasting and partitioning. The first is that, in pasting, models are combined in a committee, without further models trained specifically to provide arbitration or combination. The second difference is that instances are selected for training each new model based on correctness of classifications by previous models. As models are only built with small data sets, pasting is computationally efficient, and can produce a model at any time. Efficiency has been further improved by performing pasting in a distributed environment [20]. However, pasting suffers the same problem as other methods of combining models (e.g. [40, 10, 17]) in that the output models become exceedingly complex. This limits the application of these approaches if comprehensible models are sought.

Rather than using all training data in small subsets, Toivonen [96] uses a small sample of training instances to generate potential association rules with relaxed constraints, then evaluates the validity of those rules on the full training set. This results in a vast speed up of execution time, but comes at the detriment of certainty as it is possible that rules that would be supported by the full training set will not be selected as potential rules from the sample.

Sequential or adaptive sampling methods have recently become popular (e.g. [48, 77, 92]). The premise behind these is that the data is iteratively sampled with increasing size until a bound on the number of instances required for confidence in a decision is reached. However, such methods require constant re-evaluation of the decision metric (e.g. information gain, gini index). When discrete attributes are used, this is inexpensive as statistics can be easily and quickly updated with new data. Continuous attributes, though, must be ordered, requiring sorting each time new data is added. Some sequential algorithms [92, 28, 60] also require that the metric must be able to be measured in an incremental fashion, and sometimes rely on the metric being monotonic [99].

The VFDT [31] and CVFDT [57] incremental learning algorithms make the number of instances required arbitrary by deferring growth of a decision tree until enough instances have reached a leaf. This is done by ensuring bounds on the number of instances required to make a decision are met, and means that a decision will not be made with more data than necessary. However, this may then lead to a problem with complex concepts as the amount of data required grows with the number of leaves. These methods also work only with discrete attributes.

The CLOUDS system [1] takes a different approach by not sampling instances at a node, but by sampling potential splits. A list of all possible splits is generated, with only a sampled subset of these splits evaluated. This may be expected to result in a loss of accuracy while reducing execution time, as less search is performed. Results of the presented experiments show that CLOUDS generally produces less accurate models than CART and C4.5 on the four data

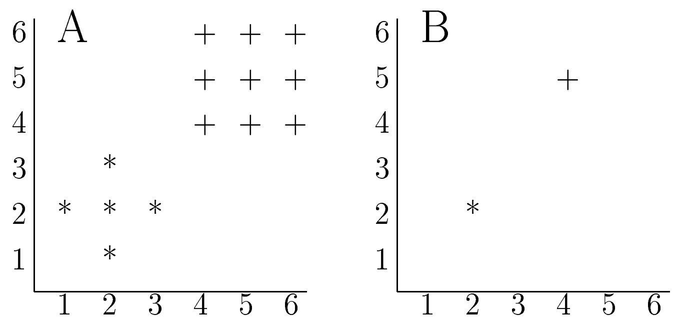

Figure 2.1: An example of clustering. Each class of instances in A has been replaced by a single instance in B. The instance of class + will have weight 9, and the instance of class * will have weight 5.

sets tested. No comparison of execution time is given.

Clustering and Squashing

Random or semi-random selection of instances for a training set runs the risk that not all instances will be represented in the sample. Ensuring representation of all instances requires taking large samples — an undesirable action. An alternative to random selection is to deliberately create a training set that represents every instance of the full training set. This can be achieved with clustering algorithms (e.g. [32, 79, 8, 26]).

Clustering algorithms analyse a set of instances with the aim of identifying groups of similar instances. Each group can be represented in a training set by a single weighted instance, where the weight represents the number of instances in the group. For example, consider Figure 2.1.

A common way of performing clustering is with the k-means method. This method groups instances into k clusters, where k is user-specified. The position of a cluster may change dynamically as more data is analysed, with instances being assigned to the cluster "closest" in the attribute space. Many clusters may exist for one class, and clusters may consist of only one instance.

Many clustering algorithms, such as k-means, are suitable for data sets that contain continuous attributes only. However, much research has been performed into clustering that incorporates discrete attributes [52, 47].

The benefit of clustering is obvious. If a data set contains distinct groups of instances, representing these groups as a single instance should save a considerable amount of execution time. However, there are a number of problems associated with clustering. First, clustering algorithms are expensive, often having computational complexity of $O(n^2)$ [34]. Research has been performed to reduce this cost, including approaches incorporating clustering into database management systems [36]. If data can be extracted from a database already clustered, the problem of computational complexity is reduced. Second, for an algorithm to be able to use clustered data, it must be able to take weights of instances as input. Thirdly, clustering is a form of lossy compression, and thus the information contained within a clustered data set is less than that of the unclustered data set. An example of this can be seen in Figure 2.1, where the boundaries of each clustered class have been lost. Variances of attributes and co-variances between attributes have also been lost. This may increase the possibility of inferring an erroneous decision boundary, leading to incorrect classification of future instances.

Data squashing [34, 82] can be viewed as a variant of clustering, designed to reduce information loss. Whereas clustering replaces a group of instances with a single weighted instance, data squashing replaces a group of instances with many weighted instances.

Data squashing explicitly attempts to reduce the number of instances of a data set while keeping many properties of the data intact. This is achieved by ensuring moments of the squashed data are similar to those of the original data set. Moments are calculated on a region-by-region basis, where regions are determined through the application of hyper-rectangles or data spheres to the instance space.

Other methods of reducing data set size exist, such as that presented by Vucetic and Obradovic [98]. Their method uses a compression algorithm to reduce the number of bits required to represent a data set, in turn reducing the resources required to store and transfer data. However, the method requires continuous attributes to be discretised.

Learning Curves

One way in which sampling is often applied is with learning curves. A learning curve is a plot of accuracy against training set size. Such a plot can be used to determine the smallest training set size for which adding extra instances does not improve accuracy.

Substantial research has been performed into estimating learning curves. Some methods involve fitting a function to a number of small sample sizes such that the curve can be extrapolated [42, 49]. Others attempt to determine a lower bound on the number of instances required [35], or an upper bound on potential accuracy [53].

Possibly the most promising method is that of progressive sampling [84], where iteratively larger samples are taken until confidence that accuracy will no longer improve is reached. This has been further studied to eliminate the need to build models below a data set dependent size [50].

However, the problem with learning curves is that they cannot guarantee accuracy will be reached that is sufficiently similar to that obtainable with the full training set. It has been suggested that this is not possible as the accuracy represented in learning curves does not plateau, but tapers [83, 71]. Therefore, one of the assumptions of many learning curve algorithms — that a point will be reached after which accuracy does not increase — may be violated.

2.2.3 Type of Algorithm

Rather than modifying algorithms to handle large data sets, it may be sensible to look at different types of algorithms that may be more suited to large data sets. This section discusses two ways in which algorithms may be more suited to large data sets.

Incremental Algorithms

One of the problems with many learning algorithms is that incorporating new data into a model requires discarding of the current model and starting again. Given that execution time can be large, this is certainly undesirable. Incremental algorithms can alleviate the problem.

An incremental algorithm is one that modifies an existing model when presented with new data. Kalles and Papagelis [61] discuss incremental learning within the context of decision trees, and show that incremental decision tree algorithms can have similar performance to algorithms which process all data at once. VFDT [31] and CVFDT [57] are also incremental decision tree algorithms, with the latter having the added advantage that previously learned concepts of the tree can change if the data changes over time.

Bias plus Variance Decomposition

Another potential way in which a different type of algorithm may be useful with large data sets is in the bias plus variance decomposition of error [59]. The bias plus variance decomposition provides an insight into how much of the error is due to randomness of the training data, and how much is due to systematic issues with the learning algorithm. It is plausible that as training set size increases, error due to randomness of the training data should decrease, leaving error dominated by systematic issues — or bias. However, no previous study has investigated this.

2.3 Summary

This chapter has presented a review of literature relevant to the thesis. Problems faced when using classification learning algorithms with large data sets have been discussed, as have methods that attempt to remedy these problems. Scaling up algorithms to handle large data sets does not work well. Scaling down data sets introduces risks that are often not addressed or not well investigated. Regardless of these drawbacks, such methods for handling large data sets are often used.

There has been little research performed into whether the type of algorithm required when learning from large data sets differs to that required when learning from small data sets. Specifically, the effect of large data sets on the way the bias plus variance decomposition of classification error changes has not been studied. Sampling is possibly the most commonly used method of scaling down data sets, but no study into the comparative effects of different types of sampling methods has been performed. Learning curves are often used in an attempt to reduce execution time. However, there is little evidence to suggest that they can be effectively used to reduce execution time while reliably producing models with accuracy comparable to that obtainable without reducing data set size.

Chapter 3

Bias and Variance

One of the many problems facing data mining is the abundance of data. Within the classification learning context, more data rarely leads to the creation of less accurate models [71, 83]. It can, however, cause classification learning algorithms to take a substantial time to create models.

This can be at least partially attributed to the fact that many popular classification learning algorithms (or algorithms on which they were directly based) were created when the average data set size was what is now considered quite small. For example, C4.5 [88] was published in 1993, and its predecessor ID3 [87] first appeared in 1986. Classification and Regression Trees [13] was published in 1984. Since then the amount of data collected and stored has grown enormously. Whereas data sets now often consist of millions of instances, at the time these algorithms were created, typical data set sizes ranged in the hundreds to thousands of instances.

It is reasonable to believe that this lack of data may have caused authors to unintentionally predispose their algorithms toward learning from small data sets. If larger data sets had been available, such algorithms may have been developed differently. Even so, if data sets available now had been available then, creating, testing, and adapting an algorithm would have been very difficult with the technology of the time.

However, execution time may not be the only way in which such influence manifests itself. It is plausible that one of the legacies of these circumstances is that the learning process of such algorithms is inclined toward small data sets. If this is true, it could then be expected that these algorithms do not perform as well on large data sets as they might if they had been developed with large data sets. This implicit and unrealised constraint on model-building performance could materialise as longer execution time, decreased model descriptiveness, or decrease in accuracy.

One way in which potential performance gains can be tested is to investigate the bias plus variance decomposition of classification error. This decomposition allows analysis of error attributable to randomness in the selection of the data and error attributable to systemic conflicts between the algorithm and the learning task. Such an investigation may also provide insight into which of these two types of error future development of algorithms should focus on minimising. A more detailed discussion of the decomposition appears below.

Experiments were performed to investigate two aspects of the bias plus variance decomposition: first, how decomposed error changes with larger training sets, and second, how changing the expected level of bias management affects models built with larger training sets.

3.1 Investigation of the Effect of Training Set Size on Bias and Variance

The machine learning field of classification prediction aims at producing a model that describes the concepts underlying a given data set, such that one of the attributes of new data (the "class") can be predicted through analysis of other attributes. Such a model is created by a classification learning algorithm.

When used to classify previously unseen instances, inferred models often have error. This section investigates how the nature of this error, in terms of the bias plus variance decomposition, is affected by the size of the training set used to infer a model.

3.1.1 The Bias plus Variance Decomposition of Classification Error

To fully describe the underlying concepts, a data set must contain enough combinations of values across all attributes such that there can be absolutely no ambiguity as to which class a particular combination of attribute values belongs. It is rarely the case that a data set fulfills this requirement, and it is often not possible to meet this requirement when continuous values form part of the data set. A data set is therefore usually only a sample of the underlying concept, and a particular data set one of many possible data sets derivable from the underlying concepts.

Statistical inference methodologies attempt to draw conclusions regarding a population from a sample of that population [76]. To mitigate the chance of obtaining an unrepresentative sample, statistical methodologies select the sample from the population randomly. Also, since only a sample of the population is used, error can be expected. Statistical methodologies define the notion that this error can be decomposed into two parts: error due to systematic inadequacies in the learning process, and error due to the fact the sample was taken randomly. When measured over all possible samples of a given size, these causes of error are known respectively as bias and variance.

These statistical properties of error can be closely related to error in predictive classification. Bias can be viewed as those instances consistently incorrectly classified by multiple models trained on different samples, whereas variance can be viewed as instances occasionally incorrectly classified. Obviously, instances consistently correctly classified do not contribute to error.

Two problems arise, however. First, it is often impractical within a classification prediction context (as it also is within a statistical context) to create a model from every possible sample of a given size. Therefore, bias and variance measurements can only estimate the true bias and variance. The more samples that are used, the more accurate the estimates become.

Second, bias and variance were originally defined for domains such as estimating means or evaluating regression models — domains in which the attribute to be estimated is continuous. Methods for calculating continuous bias and variance do not directly map to discrete classification. There is no single method to calculate bias and variance within a discrete context, and many methods for calculating these measures within a predictive classification context [67, 66, 9, 95, 43, 58, 101, 59] have been proposed. These methods are discussed in Section 3.1.3.

Bias and Variance Management

Levels of bias and variance are highly dependent on the data set and number of training instances used. It is therefore difficult to make predictions of expected levels of bias and variance. It is possible, however, to make predictions regarding the relative level of management of these factors when comparing algorithms. Whereas bias and variance refer to a decomposition of measured error, relative levels of bias and variance management refer to an expectation of the degree to which an algorithm focuses on reducing bias or variance with respect to another algorithm.

Take the following examples of bias management:

- Increasing expressive power allows an algorithm to describe more concepts. Thus, the potential to build a model that adequately represents concepts underlying the data is increased. It could be expected that an algorithm with greater expressive power has a greater level of bias management.

- Incorporating boosting [91] into an algorithm has been shown to reduce bias [4]. It can therefore be expected that a version of an algorithm that uses boosting employs greater bias management than a version that does not.

Determining relative levels of bias management may not always be as straightforward as the above cases. For example, comparing a decision tree algorithm to a decision rule algorithm is much more difficult. Such algorithms are often able to express the same concepts, but due to the way in which these concepts are inferred and employed, a decision tree algorithm and a decision rule algorithm may not necessarily produce precisely the same bias error.

The relative level of variance management is easier to determine than bias management. This can be found by comparing the susceptibility of algorithms to small changes in the data. For example, with default settings, C4.5 is relatively robust to small changes in data. Changing C4.5's settings so that fewer instances are required to branch a decision node makes the algorithm more susceptible to a small change in the data. Therefore, C4.5 modified to require fewer instances per branch can be considered as having less variance management than default C4.5. Similarly, Naïve-Bayes classifiers are known to be altered very little by small changes in the data, and can therefore be considered to have high variance management.

It is important to understand the distinction between bias and variance, and management of bias and variance. Bias and variance are measured quantities. Bias and variance management are expectations of relative performance. Comparing the levels of management of, for example, bias, does not necessarily imply that the algorithm with higher bias management will always have lower bias. Although it can be expected that the algorithm with higher bias management will tend to have lower bias, it may not have lower bias for all learning tasks. However, given measurements of bias, it is reasonable to conclude that the algorithm with lower bias is likely to have higher bias management.

3.1.2 Justification

The above discussion does not address why an investigation of bias and variance may be useful within a discrete predictive classification context. The reason is thus: within the classic statistical context of a continuous domain, variance can be expected to be a significant proportion of error given a small sample of the population [76]. As the size of the sample increases, variance can be expected to decrease. No assumption is made regarding the behaviour of bias as larger samples are used. Although expectations of bias and variance within continuous statistical domains may hold for continuous predictive classification, there is no evidence that they hold for discrete predictive classification. It would be useful to be able to characterise how discrete predictive classification algorithms behave in terms of both systematic and random error, especially with larger training sets.

An associated point is that many common classification prediction algorithms were created when the number of instances in training sets was much smaller than is common today. Since it is known that variance is a large component of error with small samples, it is reasonable to expect that such algorithms may focus on variance management. If variance becomes a negligible proportion of error with large training sets, these algorithms may be underperforming when learning from large training sets as they do not focus on the then more important task of bias management. Evidence of this may provide an impetus and justification for the design of algorithms that focus on bias management rather than variance management. Such algorithms may perform poorly on small training sets, but excel when tackling larger training sets.

The purpose of the research in this chapter is to investigate whether the statistical expectations of bias and variance do indeed apply to discrete predictive classification. Evidence can be collected by examining the bias plus variance profile of various algorithms at different training set sizes. If these expectations are confirmed, then this may have a significant impact on the approach algorithm designers take when trying to solve modern problems. Designing algorithms focusing on bias management would be a major paradigm shift from the current emphasis on variance management algorithms.

3.1.3 Experiments

Two hypotheses are investigated:

- as training set size increases, variance can be expected to decrease.
- as training set size increases, bias will become a dominating proportion of total error.

Experiments were performed to evaluate the hypotheses. Experimental design considerations are addressed below.

Methodology

All experiments were performed using ten times three-fold cross-validation. Thus, every instance was classified in a test set precisely ten times. This ensures that results are not influenced by disproportionate representation of selected instances, as every instance contributes equally to the overall result. Webb and Conilione [103] show cross-validation provides a more stable estimation of bias and variance than the “training pool” method introduced by Kohavi and Wolpert [66].

Sampling of training sets was designed to ensure that larger training set sizes included all instances in smaller training set sizes. This is desirable as any change in bias and especially variance can only be caused due to the addition of extra instances to the training set, and eliminates the possibility that results could reflect the effect of sampling a whole new training set. Since the aim of these experiments is to investigate the effect increasing training set size has on error, ensuring smaller training sets are subsets of larger training sets is appropriate.

Bias plus Variance Measure

Many definitions have been proposed for measuring bias and variance in a discrete classification learning context.

The most widely used measure is that of Kohavi and Wolpert [66]. Error is decomposed into bias, variance, and irreducible error, with bias and variance defined in terms of the squared error between induced classifiers and data. Variance is considered to always contribute to error, and therefore cannot be negative. Irreducible error is that which cannot be reduced, and occurs when instances with the same description have different classes.

The philosophy of this measure is similar to that used in statistical inference, and is defined as follows:

$$\text{bias}_x^2 = \frac{1}{2}\sum_{y\in Y}[P(Y_F = y|x) - P(Y_H = y|x)]^2$$

$$\text{variance}_x = \frac{1}{2}(1 - \sum_{y\in Y} P(Y_H = y|x)^2)$$

$$\text{irreducible error}_x^2 = \frac{1}{2}(1 - \sum_{y\in Y} P(Y_F = y|x)^2)$$

where Y is the set of output classes, Y_F is a fixed function that maps each instance x to a class y, and Y_H is a hypothesis estimating Y_F. Note that for noise-free functions, $P(Y_F = y|x)$ equals 1 for a single value of y, and 0 for all other values. Similarly, for classifiers such as decision trees, $P(Y_H = y|x)$ equals 1 for a single value of y, and 0 for all other values.

Then,

$$\text{bias}^2 = \sum_x \text{bias}_x^2$$

$$\text{variance} = \sum_x \text{variance}_x$$
$$\text{irreducible error}^2 = \sum_x \text{irreducible error}_x^2$$

Kong and Dietterich [67] define bias such that a test instance can only contribute to bias or variance, never both. An instance that is most often classified incorrectly is considered a bias error, with weight 1. Variance is defined as the difference between total error and bias, with no separate term for irreducible error. Thus, an instance can have negative variance if it is most often classified incorrectly, but sometimes classified correctly. Bias and variance are then determined by calculating the mean bias and variance of all test instances.

Allowing negative variance for an instance has the undesirable property that, when averaged over all instances, variance can be zero even though variance for individual instances is not zero. This can give a misleading impression of measured variance.

Other measures aim at calculating error attributable to bias and variance, rather than the actual bias and variance themself. James and Hastie [58] take this approach, defining systematic effect and variance effect, and show Tibshirani's definition of variance [95] to equate to theirs within the class of classification rules. This has been taken further [59], suggesting two measures of bias and variance should be used — one measuring bias and variance, the other measuring their effect.

Breiman [9] defines bias and variance such that they can be interpreted as error due to bias and error due to variance. Webb [101] takes the same approach, but whereas Breiman defines a separate term for irreducible error, Webb absorbs irreducible error into bias and variance. However, although these measures differ theoretically, in this research they are in practice identical, as no two instances in any data set employed have the same description with differing classes.

Friedman [43] investigates the interaction between bias and variance, and shows that rather than having an additive effect, they actually have a multiplicative effect. It is therefore possible that reducing variance will increase error. This can occur regardless of whether variance is permitted to be negative, and can be explained as follows.

Let bias and variance be viewed as regions of the attribute space in which error occurs. Bias is then the space in which the most commonly predicted class of instances in that space is incorrect, and variance is the space in which the most commonly predicted class is correct. Such areas cannot overlap. However, the context of a region of variance may determine whether it increases or decreases error. For example, consider a variance region that is totally surrounded by regions in which instances are always correctly classified. In this case, the variance region can be viewed as introducing error. On the other hand, consider a variance region that is totally surrounded by bias regions. Then, the variance region can be viewed as introducing correctness. Reducing variance in such a region will hence increase error.

Bias and variance were calculated using the Kohavi–Wolpert, Kong and Dietterich, and Webb measures. All measures recorded similar results. Therefore, only the Kohavi-Wolpert measure is reported as it is the most widely employed.

Algorithms

The experiments were aimed at investigating whether variance can be expected to decrease with increasing training set size. Therefore, multiple algorithms with differing levels of variance management are employed.

A Naïve-Bayes algorithm [70] was chosen as it is very robust with respect to small changes in data. It can therefore be considered as having high variance management. Naïve-Bayes classifiers can represent only a very strict subset of concepts — namely, those for which all attributes are independent. However, attributes frequently are not independent [107]. Naïve-Bayes is hence considered an algorithm with little bias management.

Three variants of C4.5 were used, since options for controlling the behaviour of C4.5 allow different levels of variance management to be investigated without having results affected by changes in the basic algorithm. As pruning has been shown to reduce variance [4], leaving trees unpruned is an obvious choice for decreasing variance management. Changing the minimum number of instances required to create a branch can also be expected to affect variance management — if fewer instances are required, small changes in data can have greater impact. This is a decrease in variance management. By default C4.5 requires that all branches of a node must contain at least two instances. If this condition is not satisfied, branches are not allowed and the node must therefore be a leaf. Hence, C4.5 without pruning and with only one instance required in each branch (hereafter referred to as *Unpruned C4.5 Min1)* should further reduce variance management.

MultiBoosting [101] has been shown to reduce both bias and variance. MultiBoost C4.5 was therefore included to provide an algorithm with more variance management than standard C4.5. This can be expected to still have less variance management than Naïve-Bayes, due to Naïve-Bayes' very high level of variance management.

The five algorithms employed and their expected bias plus variance profiles are summarised in Table 3.1.

Sample Size

While selection of sample size is arbitrary, it was deemed that using sample sizes that were powers of two would be both simple and natural within a computing context. The smallest sample size was 32. This was chosen as it is the smallest power of two greater than 30 — the minimum sample size from which it is recommended statistical inferences be drawn. Since three-fold cross-validation was used, only two-thirds of the total data was available for training. The largest

Table 3.1: Algorithms used and their bias plus variance profiles

Algorithm	Bias plus Variance Profile
Naïve-Bayes	Very high variance management, very little bias management
C4.5	Medium variance management, medium bias management
MultiBoost C4.5	More bias and variance management than C4.5
C4.5 without pruning	Less variance management than C4.5
C4.5 without pruning, minimum of 1 instance at leaf	Very little variance management

sample size was therefore the largest power of two less then two-thirds of the total data set. For example, the Connect-4 data set contains 67,557 instances. Using three-fold cross-validation reduces the maximum possible training set size to 45,038 instances. The largest power of two less than this is 32,768. This is then the largest sample size employed for this data set.

Data Sets

Data sets were drawn from the UCI Machine Learning repository [7]. Data sets were required to:

- contain a substantial number of instances. The smallest data set used was the Adult data set with 48,842 instances.
- be suitable for classification learning
- be publicly available

Table 3.2: Data sets used

Data Set	Number of Instances	Continuous Attributes	Discrete Attributes	Classes
Adult	48,842	6	8	2
Census Income	199,523	7	33	2
Connect-4	67,557	0	42	3
Cover Type	581,012	10	44	7
IPUMS	88,443	60	0	13
Shuttle	58,000	9	0	7
Waveform	1,600,000	21	0	3

A mixture of data sets containing only discrete attributes, only continuous attributes, and both discrete and continuous attributes were used. Table 3.2 presents a summary of the data sets employed. Although it is difficult to argue that the data sets used are exceptionally large, they are much larger than those used in the creation of many algorithms. Time constraints required consideration, and empirical evidence suggested that including data sets much larger than those used would require an unacceptable and infeasible amount of extra time. As this study investigates the effects of increasing data set size on bias and variance, it is the examination of variations in data set size rather than the absolute magnitude of data set size that is most critical. Also, in order to assess the relative performance of alternative methods, existing learning methods needed to be applied to the data. The computational demands of this imposed constraints on the size of the data sets that could be employed.

3.1.4 Results

Graphs show the relation between

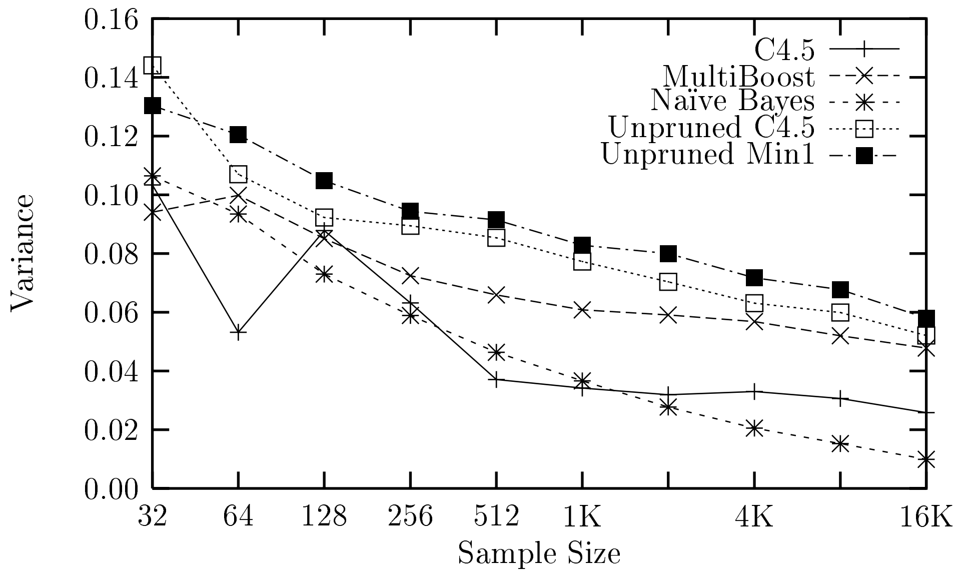

Figure 3.1: Variance of algorithms on the Adult data set

- training set size and bias,
- training set size and variance,
- training set size and the ratio of bias to variance.

Each graph contains results for the five algorithms used, and relates to one data set.

Note that none of the data sets used contained instances with identical descriptions and different classes. Irreducible error is therefore zero.

Variance

Figures 3.1–3.7 plot the variance component of error against training set size for the algorithms and data sets used. The graphs show that variance can generally be expected to decrease with larger training set size regardless of the bias plus variance profile of the algorithm, providing support for the first hypothesis.

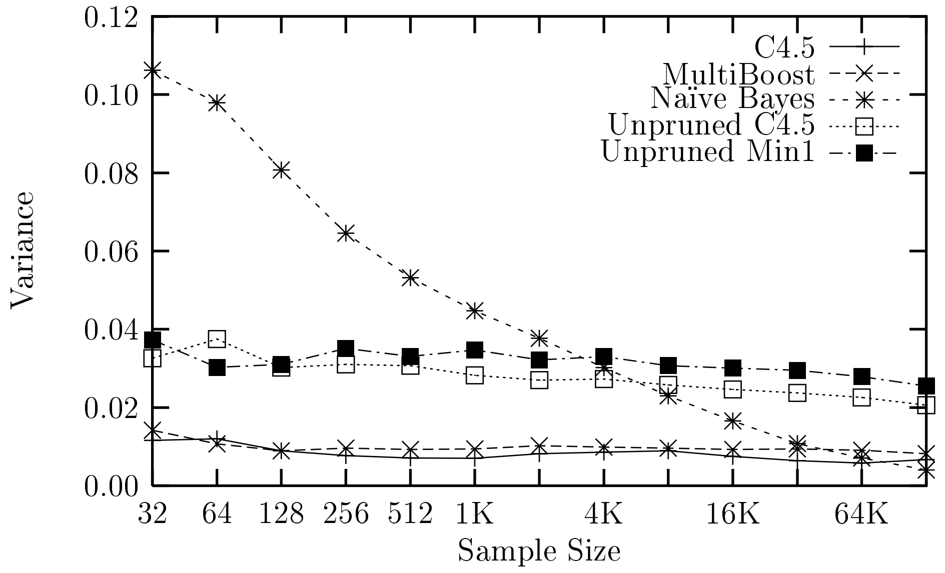

Figure 3.2: Variance of algorithms on the Census Income data set

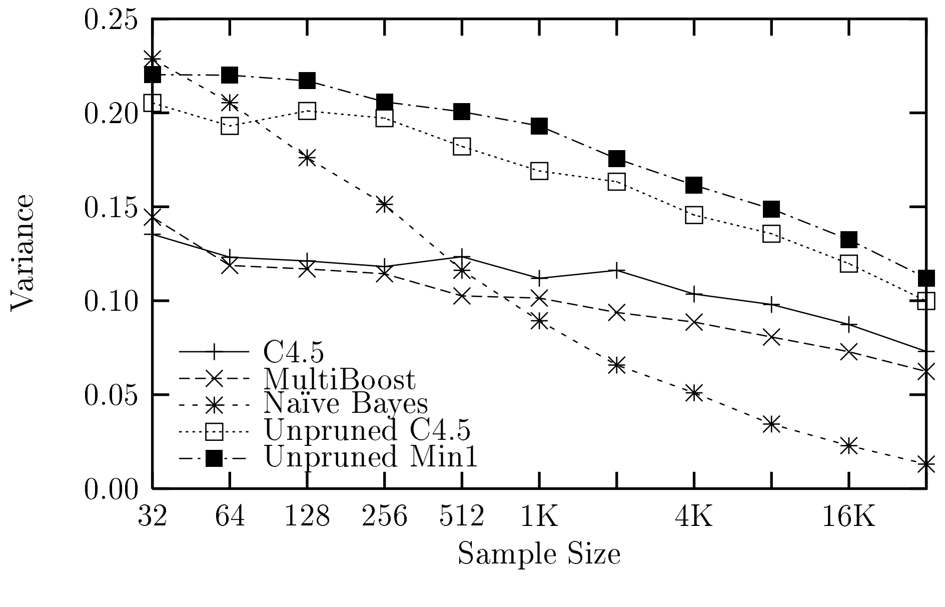

Figure 3.3: Variance of algorithms on the Connect-4 data set

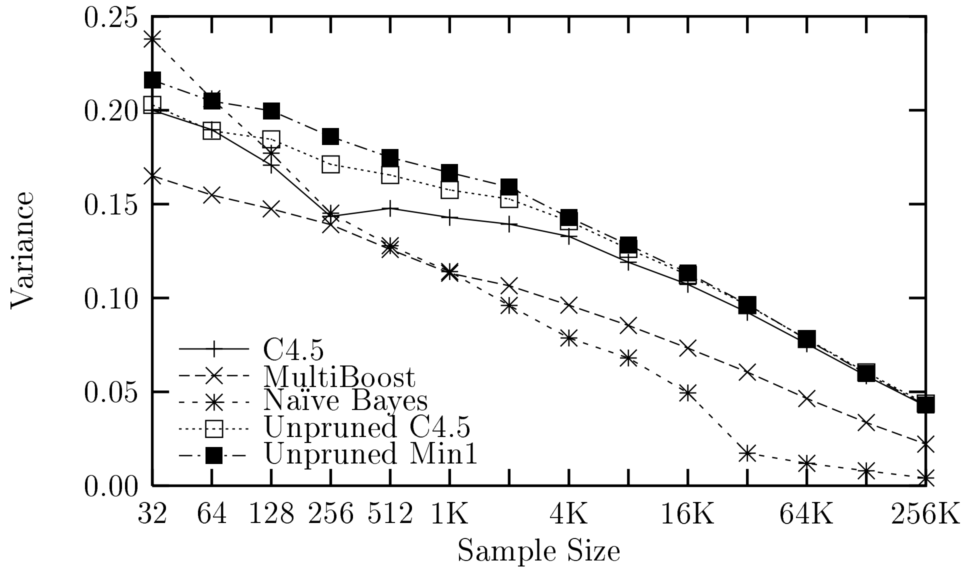

Figure 3.4: Variance of algorithms on the Cover Type data set

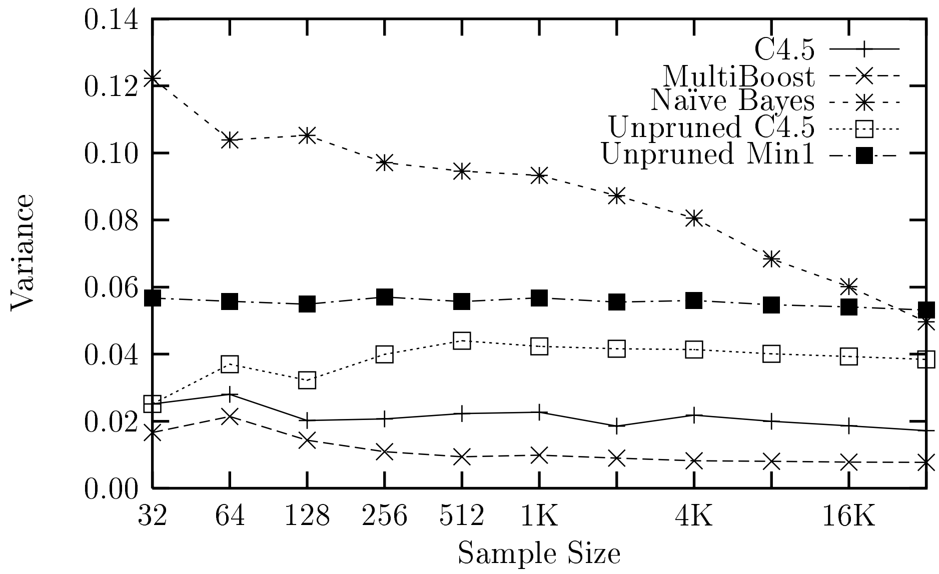

Figure 3.5: Variance of algorithms on the IPUMS data set

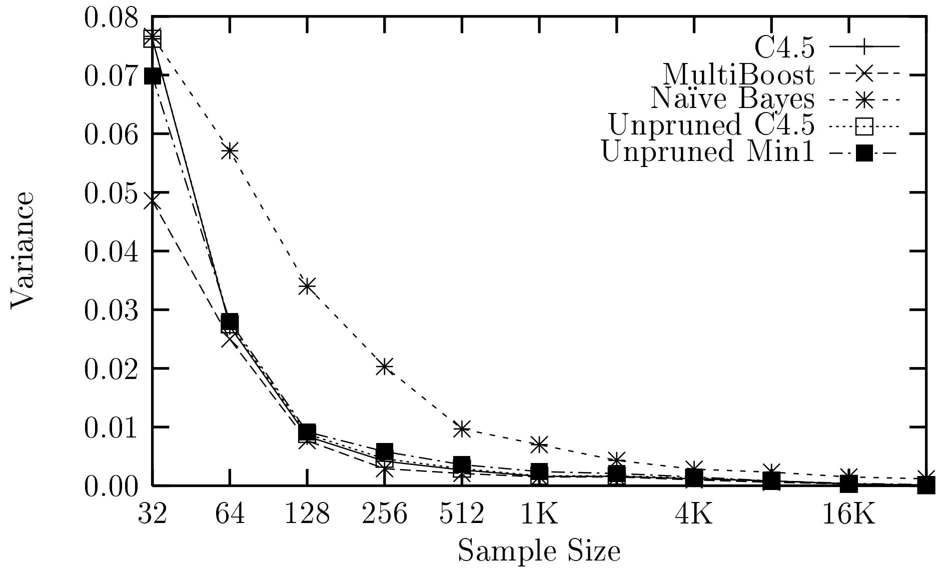

Figure 3.6: Variance of algorithms on the Shuttle data set

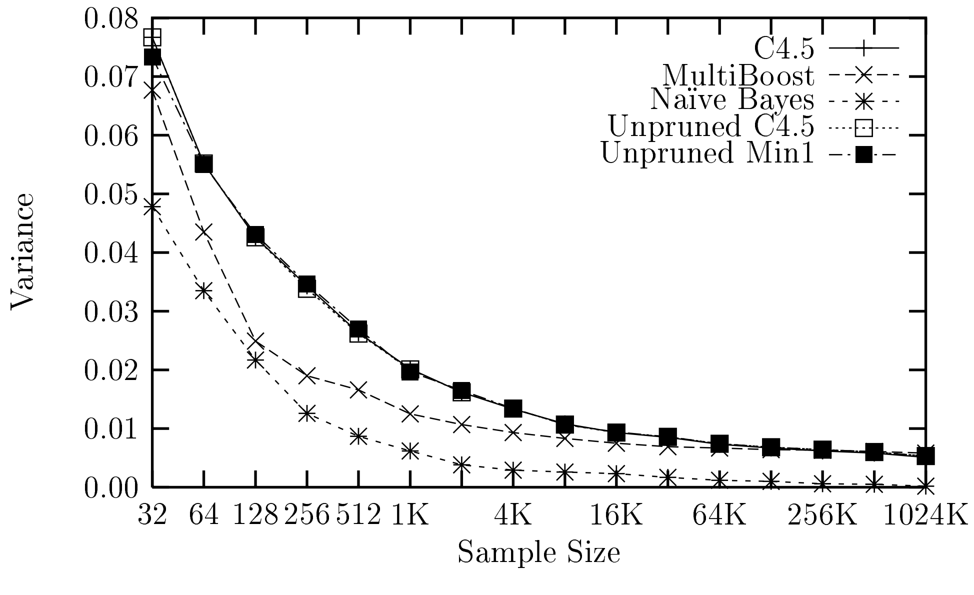

Figure 3.7: Variance of algorithms on the Waveform data set

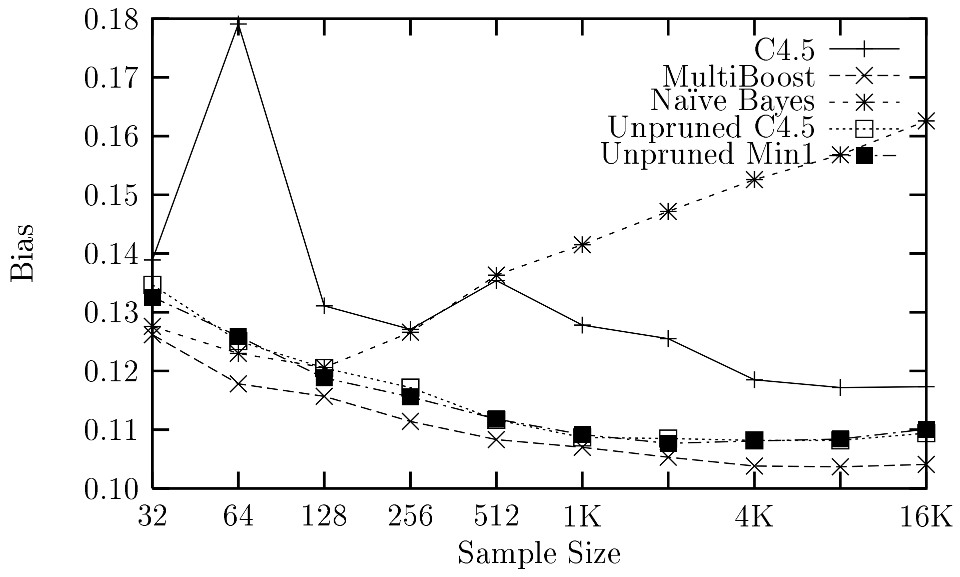

Figure 3.8: Bias of algorithms on the Adult data set

Bias

Figures 3.8–3.14 plot the bias component of error against training set size for the algorithms and data sets used. Across all data set used, all algorithms except Naïve-Bayes tend to decrease in bias error as training set size increases. Naïve-Bayes shows a trend in the opposite direction, increasing in bias for all data sets except Waveform. This may be attributable to the fact that Naïve-Bayes has little bias management, or to the very low variance of Naïve-Bayes at large sample sizes masking less of the underlying bias.

Ratio of Bias to Variance

Figures 3.15–3.21 plot the ratio of bias to variance against training set size. This may provide insight into whether bias or variance dominates error, and how the strength of this domination changes as training set size increases. Note that for the purpose of simplification of scales, results are presented as the proportion of error attributable to bias. Thus, a value of less than 0.5 represents a domination

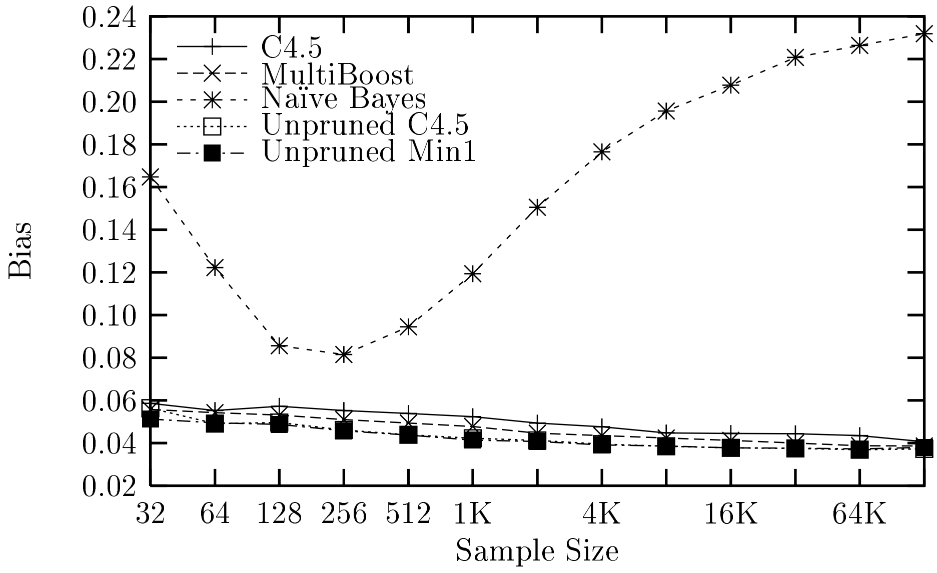

Figure 3.9: Bias of algorithms on the Census Income data set

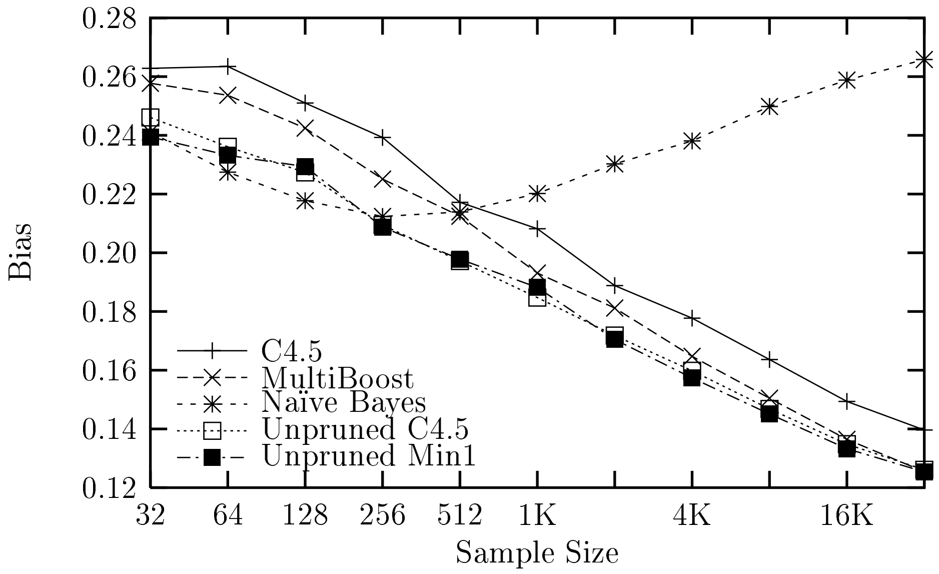

Figure 3.10: Bias of algorithms on the Connect-4 data set

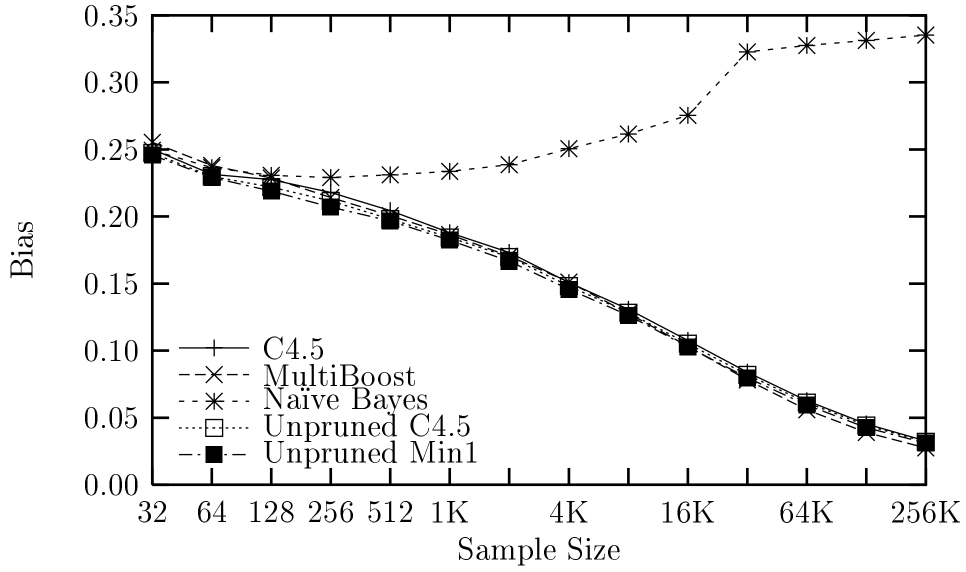

Figure 3.11: Bias of algorithms on the Cover Type data set

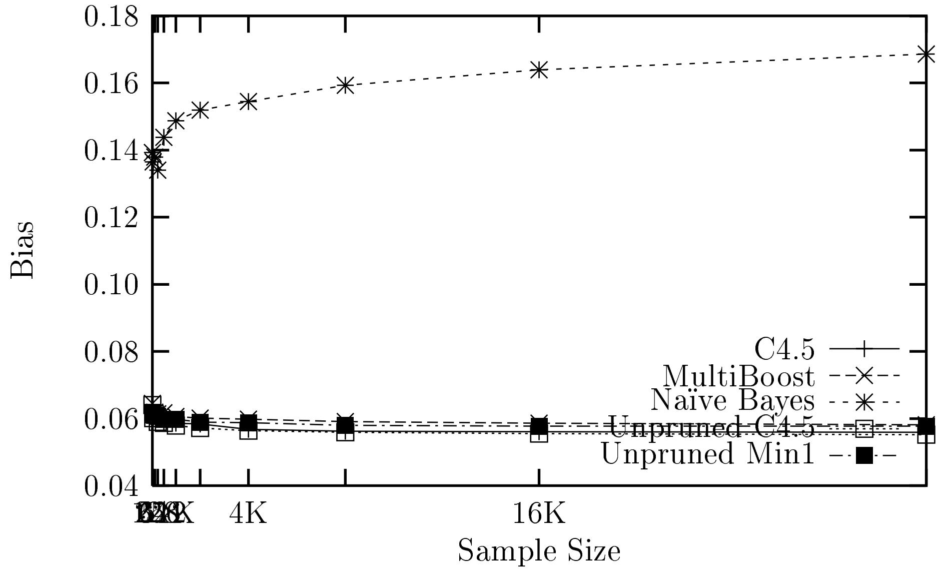

Figure 3.12: Bias of algorithms on the IPUMS data set

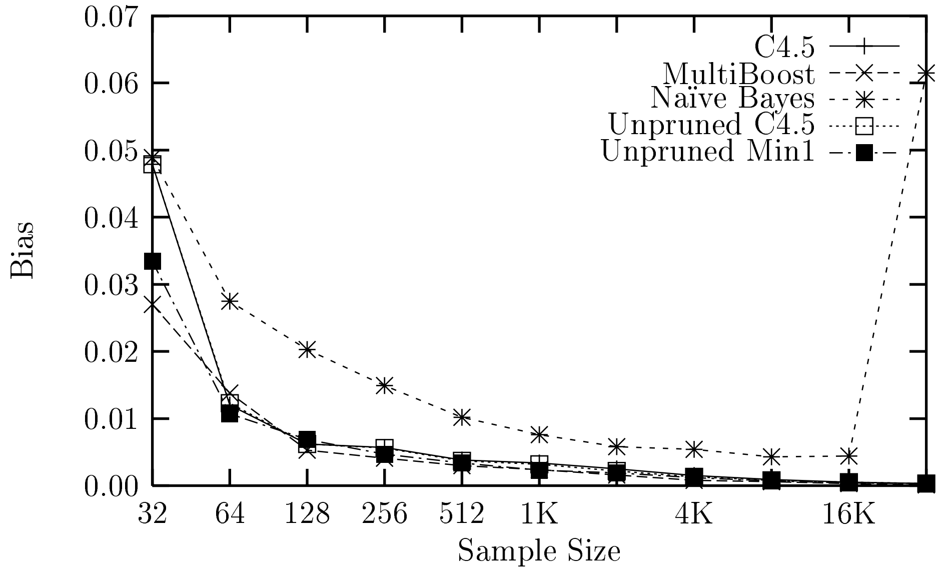

Figure 3.13: Bias of algorithms on the Shuttle data set

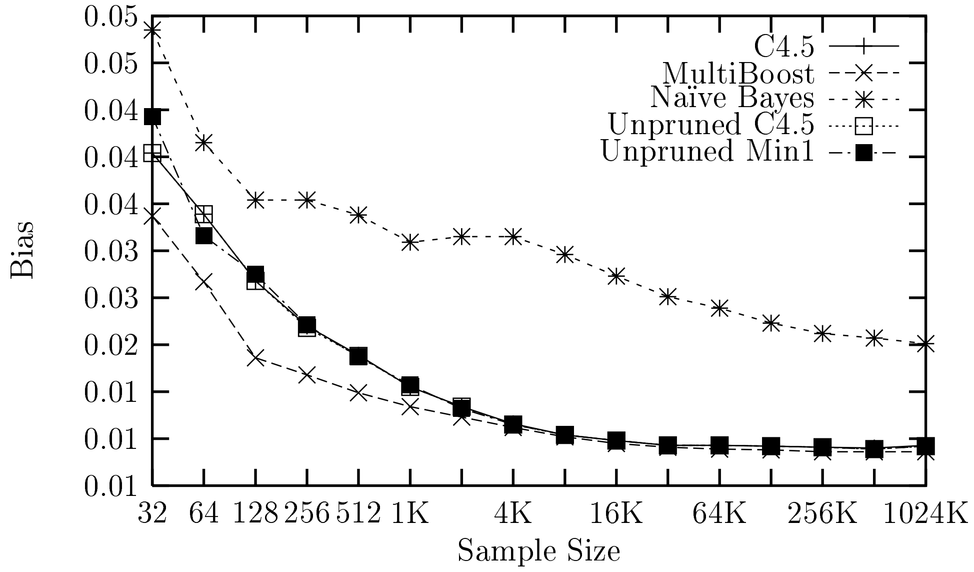

Figure 3.14: Bias of algorithms on the Waveform data set

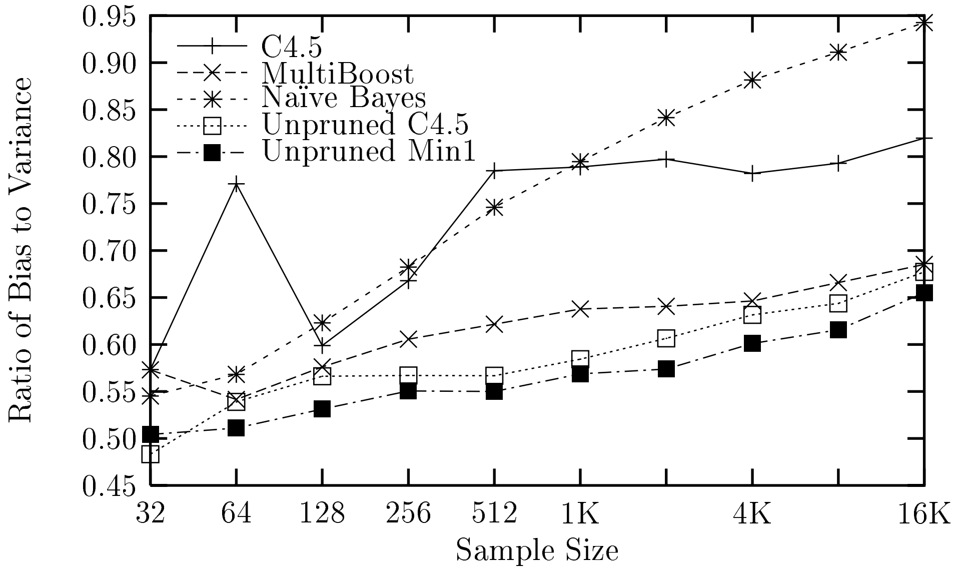

Figure 3.15: Ratio of Bias to Variance of algorithms on the Adult data set

of error by variance, and a value greater than 0.5 represents a domination of error by bias. Values further from 0.5 equate to stronger domination.

The second hypothesis is that, as training set size increases, bias will become a larger proportion of error. To evaluate this, it is necessary to compare the ratio of bias to variance between the smallest and largest training set size across the seven data sets for each of the five algorithms. If the ratio increases, then bias becomes more dominant in the final error term. An analysis finds that of the 35 comparisons, 28 support the hypothesis, with 7 against. This is significant at the 0.05 level using a one-tailed binomial sign test ($p = 0.0003$).

Further analysis comparing how often the ratio of bias to variance increases when compared to the ratio of the next smallest training set size also supports the hypothesis. Table 3.3 shows the number of times the ratio increased (considered a win) and the number of times it decreased for individual algorithms, and all algorithms, across all data sets. The result of a one-tailed binomial sign test is also given, with results considered significant at the 0.05 level. Note that since

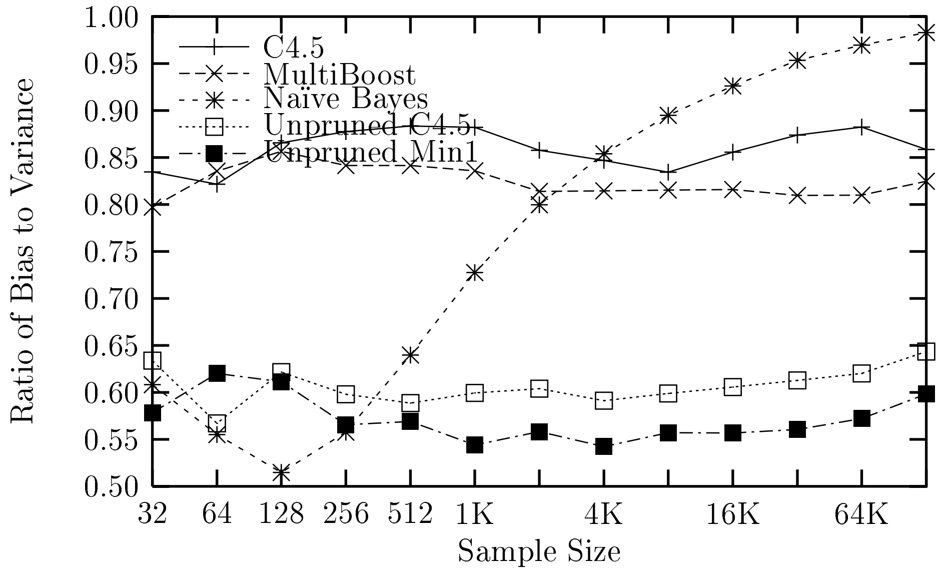

Figure 3.16: Ratio of Bias to Variance of algorithms on the Census Income data set

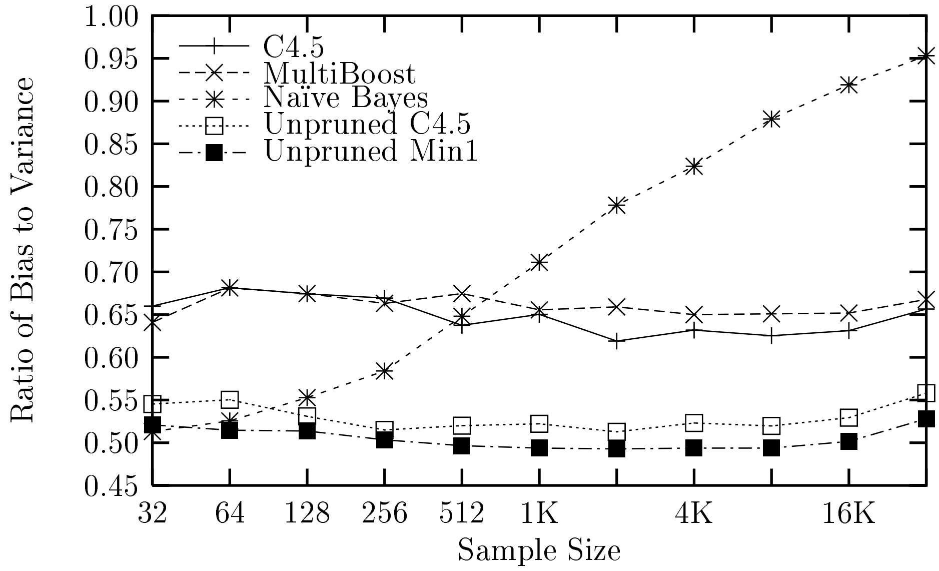

Figure 3.17: Ratio of Bias to Variance of algorithms on the Connect-4 data set

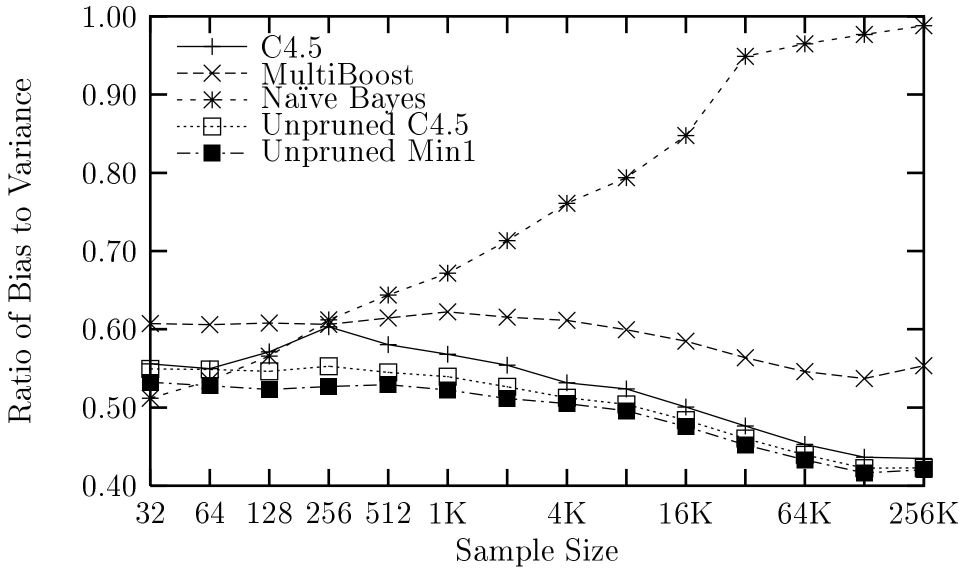

Figure 3.18: Ratio of Bias to Variance of algorithms on the Cover Type data set

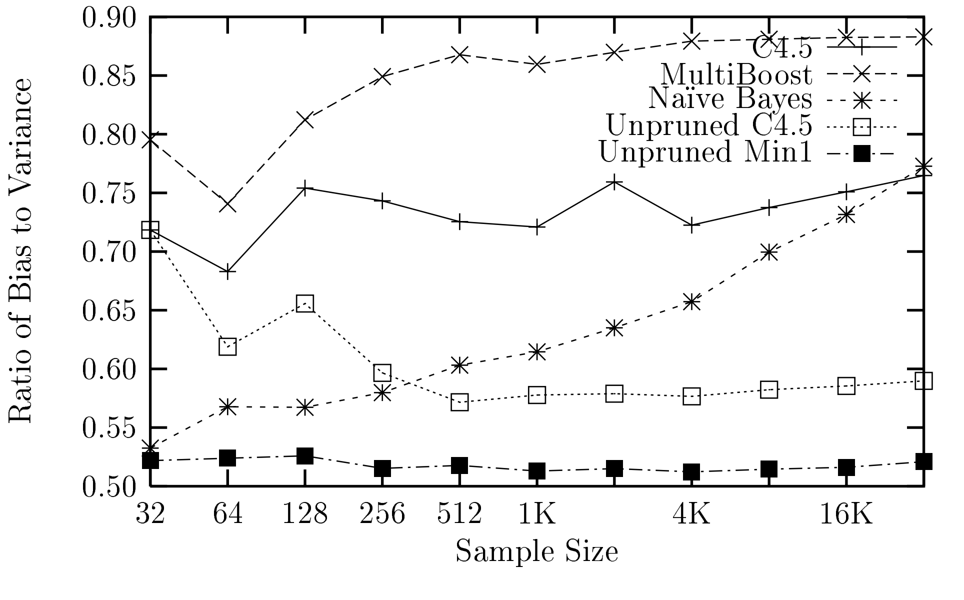

Figure 3.19: Ratio of Bias to Variance of algorithms on the IPUMS data set

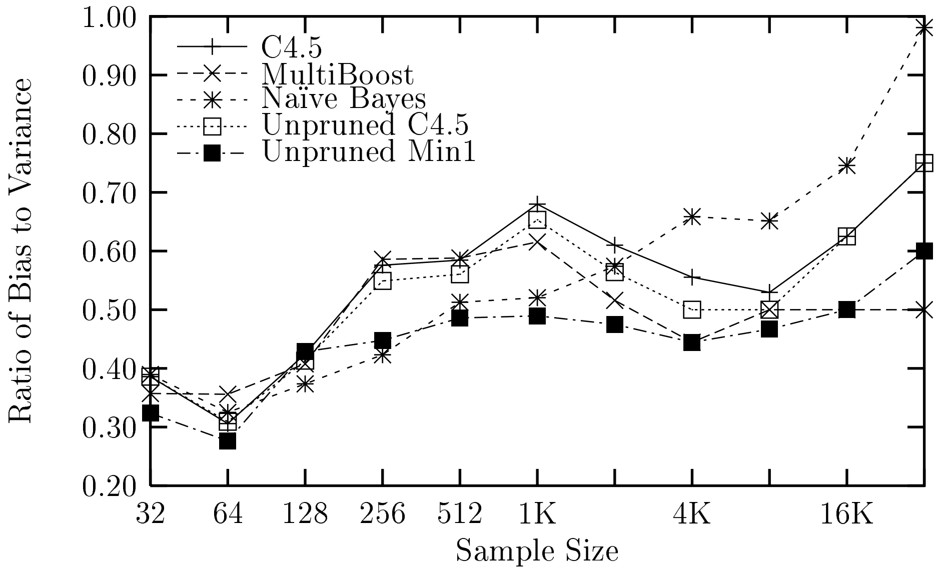

Figure 3.20: Ratio of Bias to Variance of algorithms on the Shuttle data set

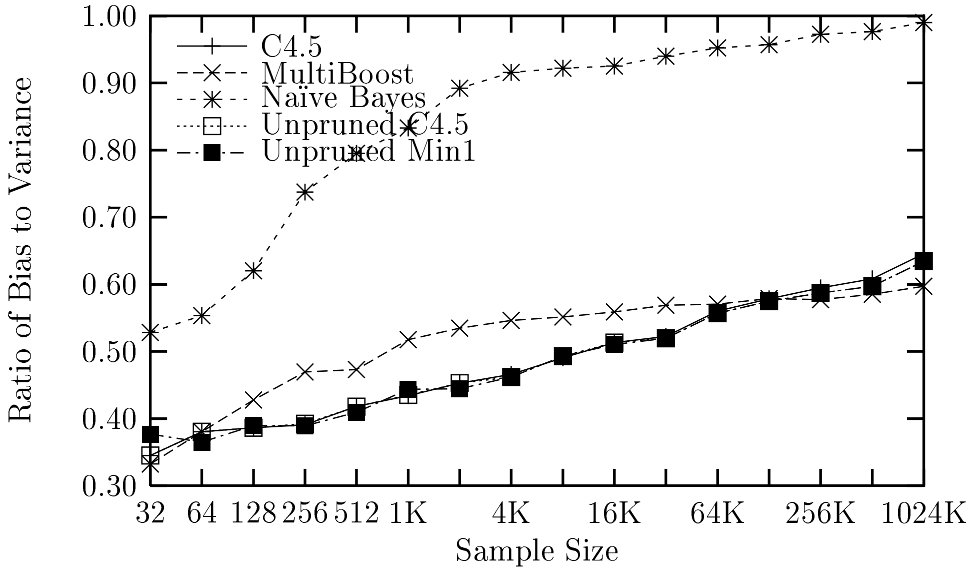

Figure 3.21: Ratio of Bias to Variance of algorithms on the Waveform data set

Table 3.3: Comparison of increases to decreases of the ratio of bias to variance

Algorithm	Win	Loss	p
Naïve-Bayes	73	5	< 0.0001
C4.5	49	29	0.0154
Unpruned C4.5	52	26	0.0022
Unpruned C4.5 Min 1	49	29	0.0154
MultiBoost C4.5	51	27	0.0044
All	274	116	< 0.0001

the hypothesis is that the ratio will *increase*, draws are considered a loss. Note also that this test violates one of the assumptions of the sign test in that not all measurements are independent. Thus, this test should be considered as broadly indicative only.

Analysis of the ratio of bias to variance at the largest training set size shows that bias dominates error in all but 4 of 35 cases. The exceptions are the three non-boosted versions of C4.5 on the Cover Type data set, where variance dominates error, and MultiBoost C4.5 on the Shuttle data set, where bias and variance are equal.

3.1.5 Summary

The above results show clear trends for both bias and variance to decrease as training set size increases. This supports the first hypothesis. There is also a general trend for bias to become a greater proportion of error with larger training sets, providing support for the second hypothesis.

It is interesting that unlike the trend for variance to decrease, bias does not decrease for every algorithm. The algorithm for which bias does not decrease is Naïve-Bayes, the algorithm with the least expected bias management. Contrarily, Unpruned C4.5 Min 1, an algorithm with low expected levels of variance management, still shows a decrease in variance. This implies that although variance may be expected to "naturally" decrease regardless of the level of variance management, the same is not true of bias.

This result, combined with evidence supporting the hypothesis that bias becomes a larger proportion of error, suggests that managing bias should be considered more important than managing variance as training set size increases.

Effect of Training Set Size on MultiBoost Error Reduction

Since MultiBoosting has been shown to reduce both bias and variance [101], it can be expected that MultiBoosting should have less total error than C4.5. The results show that this is in general true. It may therefore be interesting to compare how much of this error reduction is attributable to both bias and variance at small and large training set sizes.

At the smallest training set size, MultiBoost has less bias than C4.5 on five of the seven data sets, resulting in a mean reduction of 4.48 per cent of the C4.5 error. MultiBoost also has less variance on five data sets, resulting in a mean reduction of 6.44 per cent of the C4.5 error. However, bias and variance are reduced simultaneously for only three data sets. This results in the mean total error reduction of MultiBoost over all data sets being only 1.72 per cent. Since only three data sets show a decrease in both bias and variance, little is to be gained from calculating the proportion of error reduction attributable to each.

At the largest training set size, MultiBoost has less bias than C4.5 on six of the seven data sets, resulting in a mean reduction of 9.85 per cent of C4.5 error. MultiBoost has less variance on only three data sets, resulting in a mean reduction of 3.10 per cent of C4.5 error. Bias and variance are reduced simultaneously on only two data sets. This results in a mean total error reduction of MultiBoost over all data sets of only 0.71 per cent.

This analysis suggests that MultiBoost is more effective at reducing error with smaller training sets than it is with larger training sets. Error reduction is due primarily to a reduction in variance with small training sets, and a reduction in bias with large training sets.

3.1.6 The Relation Between Variance Management and Bias Management

At this point it may seem that management of bias and variance are related so that approaches to increase one will result in reduction of the other. This hypothesis was evaluated with respect to the experiments by analysing the effect altering C4.5 had on bias and variance. The bias and variance of each of the three modified versions of C4.5 (MultiBoost, Unpruned C4.5, and Unpruned C4.5 Min1) were compared to the bias and variance of default C4.5. The number of times the signs of the differences differed (i.e. an increase in bias resulted in a decrease in variance or vice versa) was 167, compared to 62 times where the signs were the same. The 29 occasions where there was no difference in at least one of bias or variance were ignored. A one-tailed binomial sign test ($p < 0.0001$) indicates that there is a significantly greater chance that a modification to a learning algorithm that results in a decrease in bias or variance will result in an increase in the other as opposed to both increasing or decreasing in unison.

This could be seen as justification for not focusing on bias management in preference to variance management with large data set sizes. However, while the effects on bias and variance tend to differ in direction they also differ in magnitude. The mean absolute difference between the variance of default C4.5 and the modified versions was 0.0178. The mean absolute difference between the bias of default C4.5 and the modified versions was 0.0065. The effect on variance being greater than bias is confirmed by a one-tailed matched-pair t-test ($p < 0.0001$). This result may reflect the relatively small sample sizes employed in this research. It also adds weight to the argument that current classification prediction algorithms reflect their small data set origins primarily through incorporation of methods for variance management.

This analysis could be argued to counter the proposal that with larger data sets bias management is more important than variance management, as reducing variance management leads to greater increases in variance than reductions in bias. However, three points must be considered. First, the fact that reducing variance management results in a greater increase in variance than decrease in bias, and thus an increase in total error, can be expected. Since it is proposed that many algorithms' effectiveness occurs mainly due to focusing on variance management, it is to be expected that limiting this ability should increase error. Second, the results empirically show that bias increasingly dominates error. Reducing variance further will only amplify this domination. Finally, it is not suggested that variance management is unimportant or unnecessary, but rather that bias management becomes increasingly important with increasing training set size. Therefore, algorithms that focus on variance management may not be able to take full advantage of large data sets.

3.2 Investigation of Bias Management

The previous experiments indicated that bias management may be an important factor in the success of algorithms when learning from large training sets. These experiments suggest that with larger training set sizes, an algorithm with more bias management could be expected to be more accurate than one with less bias management.

It would therefore be useful to investigate methods with which this might be achieved. It is important to investigate the ability of a potential bias reduction method to reduce bias when using large training sets. However, before this is performed, a further understanding of how bias and variance are created is required.

3.2.1 The Cause of Bias and Variance

Although the process of measuring bias and variance can be defined, the cause of each is not fully understood. It is reasonable to expect that variance is caused by the randomness of the sample, as it is with statistical inference. However, what causes an algorithm to generally have high bias? It is likely a deficiency of the model-building algorithm, but in what way? Is it due to lack of expressive power? Is it due to processing attributes one at a time instead of many at a time? Is it due to trying to fit too many model parameters to the data set? Is it all of these, or is it something else entirely?

This can be partially answered by again viewing bias and variance as regions of the attribute space where error occurs (see Section 3.1.3).

There are a number of ways in which labelling of an area of attribute space can lead to consistent incorrect classification of test instances. These are: noise; non-detection of a concept; false detection of a concept; and erroneous extrapolation of a concept. A discussion of each follows.

Noise

Bias can be caused by noise. Consider a heavily populated data set in which an area is densely populated with instances of one class. Now consider a single instance within this area that does not have the same description as any other instance, and is also labelled a different class. If this instance occurs in the training set it is likely to be ignored, as many algorithms would consider it unreasonable to make concessions for a single instance in the middle of overwhelming support for a different class. If, given a different selection of training and test data, the instance occurs in the test set, it will constantly be misclassified. Such an instance can be considered noise, and could be caused due to inaccurate data collection or an anomaly during data collection. Noise is extremely difficult to counteract, and often little can be done to remedy this problem.

Non-Detection of a Concept

An alternative explanation to the above is that the single instance is the sole representative of a rare concept. In this case, the problem is not noise, but non-detection of a concept. However, the support for such a concept is quite small, and there is likely insufficient evidence to justify introducing a new concept into the model. Alternatively, there may be a number of instances spread throughout a densely populated area in such a fashion that no reasonable way of grouping these instances exists with a given algorithm. These instances will be drowned out by surrounding instances, and in effect treated as noise.

Such problems can be mitigated by relaxing constraints on the minimum evidence required to introduce a new concept to the model. This can be expected to result in an increase in variance error, however, and overfitting is more likely to occur. Overfitting can be viewed as variance error since pruning (which reduces overfitting) has been shown to reduce variance [4].

False Detection of a Concept

A third way in which bias can be introduced is through false detection of a concept. In this case, there is sufficient evidence to support introduction of a new concept, but such a concept does not actually exist. Such occurrences can be expected to be rare, since to produce a bias error it would be necessary that training sets include evidence for such a concept, but test sets do not. If sufficiently large training and test sets are drawn from the same distribution this situation should rarely occur in practice.

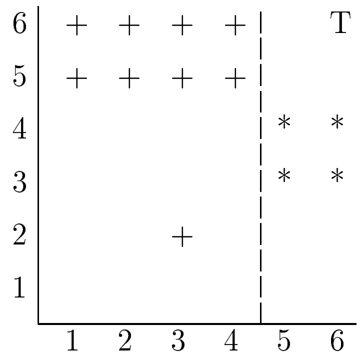

Figure 3.22: Extrapolation of a concept to cause bias

Erroneous Extrapolation of a Concept

The final method of introducing bias is erroneous extrapolation of a concept. With many algorithms, areas of the hypothesis space which contain no instances are still assigned a class label. Consider Figure 3.22: the T represents a hypothetical test instance, while the +'s and *'s represent opposing classes of training instances. Given this training set, it is reasonable to infer the decision boundary shown by the dashed line. Therefore, instances with $x \geq 4.5$ are classified *. However, evidence supporting this exists only between $3 \leq y \leq 4$. In other words, the representation of the concept that produced instances of class * has been extrapolated to include regions containing no evidence. When the test instance T is classified, it is labelled *, even though there is greater evidence for T to be labelled +. This will result in a bias error for instance T, as well as any similar instances, if such instances are of class +. Bias errors could also exist for instances of class + with $x \geq 4.5$ and $y < 3$, or instances of class * with $x < 4.5$ and $3 \leq y \leq 4$. Assuming a training set is a reasonable sample of the underlying concepts, such situations could occur quite frequently.

A remedy for erroneous extrapolation of concepts is decision tree grafting [100]. Grafting adds nodes to an inferred tree so that empty regions can be labelled independently from the regions from which they were extrapolated. The class assigned to the empty region is then based on global rather than local data. For example, in Figure 3.22, a potential region for grafting could be $x \geq 4.5$, presently assigned the label *. This region could be partitioned into $y < 4.5$ and $y \geq 4.5$, separating a populated area of this region from an empty area. If this empty area ($y \geq 4.5$) is then labelled considering global data, its label could be changed to +.

Grafting has been claimed to reduce both bias and variance error [100]. However, grafting has not been investigated within the context of large data sets.

Representational Power

It may also be reasonable to expect that bias errors could be introduced due to a lack of representational power. For example, consider two algorithms. The first represents concepts with decision surfaces orthogonal to the axes. The second represents concepts with decision surfaces that may be either orthogonal or oblique to the axes. Since the hypothesis space of the second algorithm is a super-set of the hypothesis space of the first algorithm, it may be reasonable to expect that the second algorithm has greater bias management as it can concisely represent more concepts than the first algorithm.

Similarly, an algorithm with greater representational power may also have greater variance, due to the necessity of selecting from a greater number of potential hypotheses. As shown in Section 3.1.6, an increase in variance can be expected to result in a decrease in bias.

However, increasing representational power may not necessarily decrease bias. For example, consider a data set that contains a single concept that is oblique to the axes. The second algorithm should be able to easily represent this concept with a single decision surface, while the first must approximate the concept with many orthogonal decision surfaces. In other words, the second algorithm is able to exploit its greater representational ability over the first algorithm. However, because the first algorithm must approximate the concept with many decision surfaces, the actual surfaces inferred will be highly dependent on the precise data used. Error of the first algorithm can thus be expected to be caused by variance rather than bias. Therefore, it is plausible that increasing representational power may actually be an exercise in increasing variance management, not bias management.

3.2.2 Experiments

Two sets of experiments were performed to investigate potential means of reducing bias when using large training sets.

The first set of experiments investigate how the bias reduction potential of decision tree grafting is affected by increasing training set size. Grafting has the advantage that the way in which it reduces bias is understood. It is believed that it will be beneficial to investigate a methodology that is understood, as it may be possible to analyse why the resulting behaviour occurs. The version of grafting used employs the all-tests-but-one partition [100].

The second set of experiments investigate whether increasing representational power does in fact result in a decrease in bias error, rather than variance error, as reasoned above. To do this, it is necessary to use an algorithm with which representational power can be easily restricted to a subset of the default set. OC1 [94] is one such algorithm, in that, by default, oblique decision boundaries are allowed, but can be restricted to axis-orthogonal boundaries only. At each node, default OC1 finds the best axis-orthogonal boundary and the best oblique boundary. The utility of these boundaries is then compared to determine the final boundary. Restricting OC1 to axis-orthogonal boundaries simply eliminates the search for the best oblique boundary. OC1 limited to inferring axis-orthogonal boundaries only will be hereafter referred to as orthogonal OC1.

Note that although adaptive resampling algorithms such as Arc-X4 [9], Boosting [41], and MultiBoosting [101] have been shown to reduce bias, the way in which they do so is not entirely clear. Such methods also dramatically increase execution time, and are therefore deemed unsuitable for the purposes of these experiments.

Methodology

The methodology used for these experiments is the same as that of the previous experiments (see Section 3.1.3), performing ten-times three-fold cross-validation.

Data Sets

The same data sets as previous experiments were used for the grafting experiments. However, since OC1 uses only continuous attributes, only four data sets (Connect-4, IPUMS, Shuttle, and Waveform) were suitable. To compensate, three artificial data sets and an additional real-world data set were introduced for OC1. These are described in Table 3.4.

Training Set Size

The emphasis of these experiments is on the effect of large training sets. It is therefore sufficient to compare bias and variance of algorithms at the largest possible training set size only, being two-thirds of the whole data set. This methodology was followed with OC1, as some experiments took up to 15 days for one data set. As such, performing numerous experiments with smaller data sets would require an unacceptable amount of time.

Grafting experiments did not require such large execution time. This allowed the opportunity to explore many training set sizes, so as to create a clearer picture of how the bias and variance of grafting changes with larger training sets. Training set sizes used for grafting experiments were determined in the

Table 3.4: Description of artificial data sets created

Name	Attributes	Classes	Concept
Distinct Boundary	2	2	Classes separated by $y = x$
Fuzzy Boundary	5	2	Each class centered around a point in 5-dimensional space, with attribute values selected randomly within a specified standard deviation
Random Binary	16	2	Each attribute is binary. Each possible combination of attribute values allocated a random class. Attribute values selected randomly, and assigned the class with the matching attribute pattern.
Sleep (real world)	13	6	Unknown

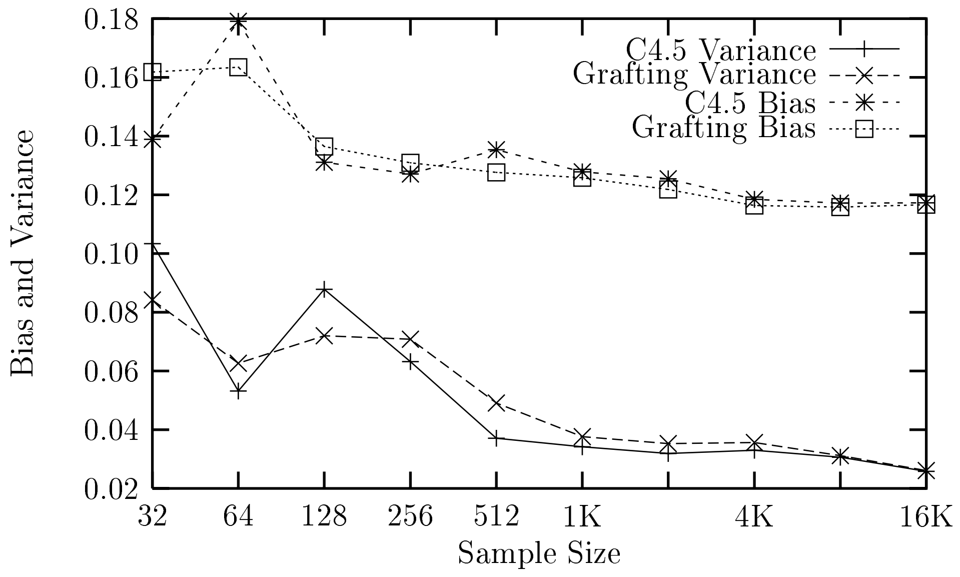

Figure 3.23: Bias and variance of Grafting and C4.5 on the Adult data set

same manner as previous experiments (see Section 3.1.3).

3.2.3 Results

Bias Reduction by Grafting

Figures 3.23–3.29 show results comparing bias and variance of C4.5 with grafting and default C4.5 for all training sets used.

At the smallest training set size, grafting has greater bias than default C4.5 for six of the seven data sets. As training set size increases the difference in bias between grafting and C4.5 decreases. At the largest training set size, default C4.5 and grafting have identical bias for the Census Income and Waveform data sets. Of the remaining data sets, grafting has less bias than default C4.5 on two data sets, and greater on three. The difference in bias at the largest training set size is never greater than 0.0007. Over all training sizes and data sets, grafting results in a decrease in bias on 31 occasions, an increase on 48, and no difference on seven.

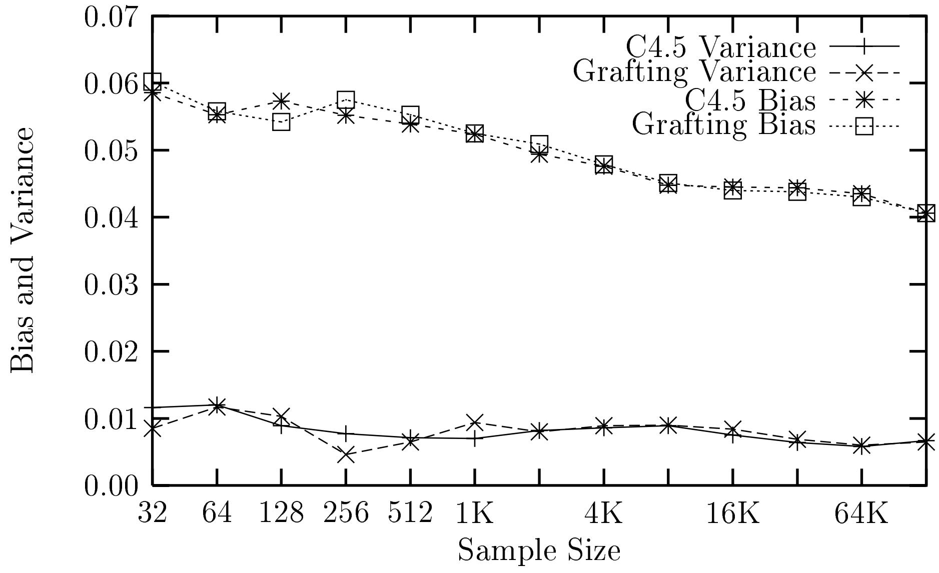

Figure 3.24: Bias and variance of Grafting and C4.5 on the Census Income data set

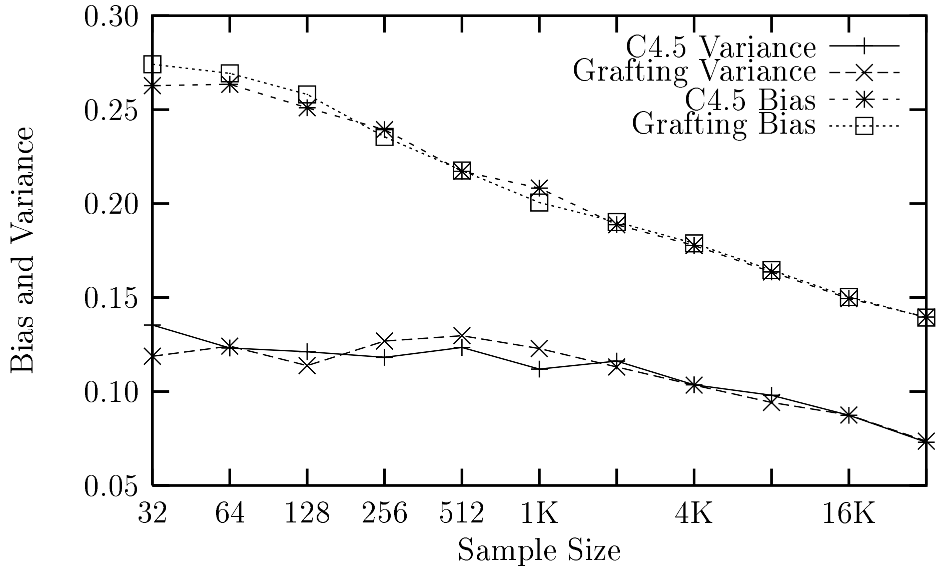

Figure 3.25: Bias and variance of Grafting and C4.5 on the Connect-4 data set

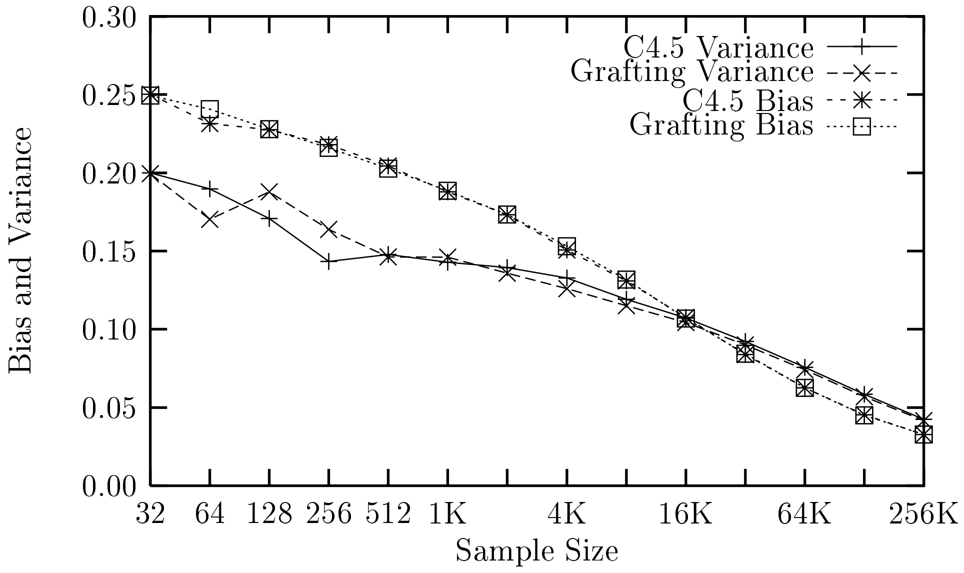

Figure 3.26: Bias and variance of Grafting and C4.5 on the Cover Type data set

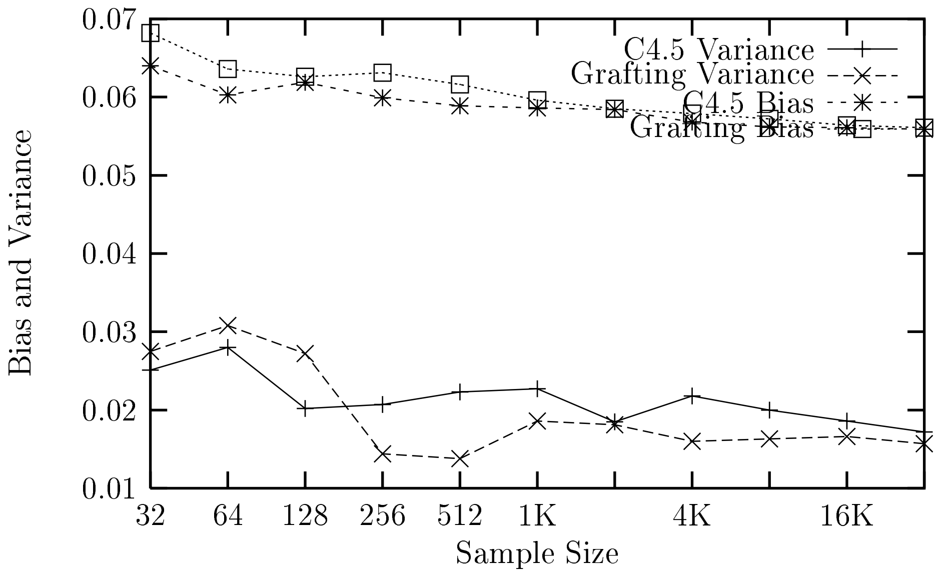

Figure 3.27: Bias and variance of Grafting and C4.5 on the IPUMS data set

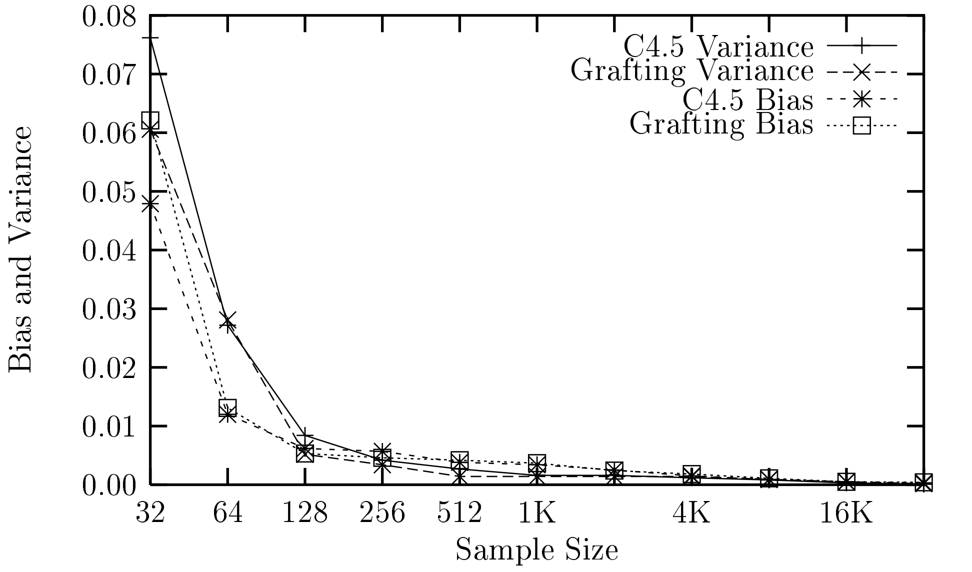

Figure 3.28: Bias and variance of Grafting and C4.5 on the Shuttle data set

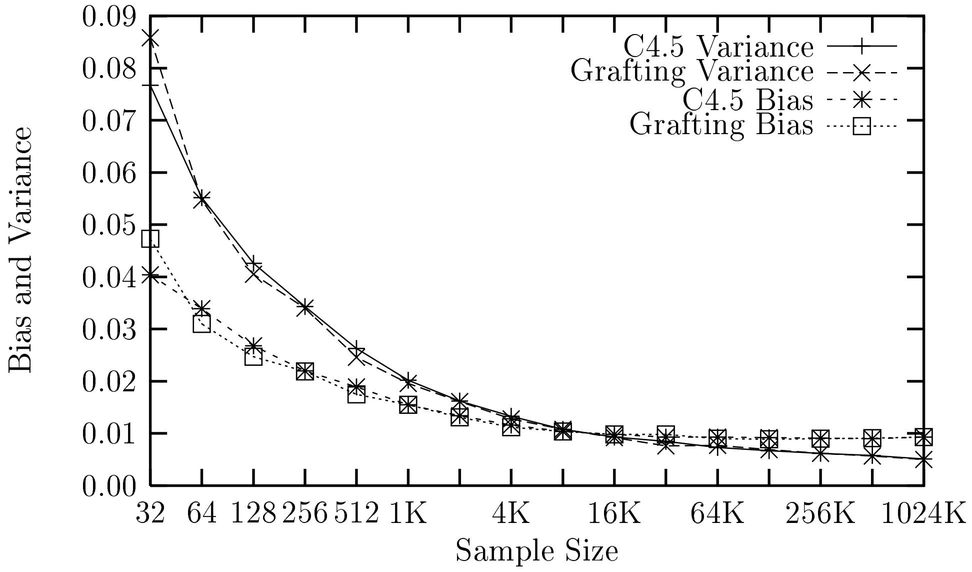

Figure 3.29: Bias and variance of Grafting and C4.5 on the Waveform data set

A possible explanation for this behaviour lies in the fact that adding instances to the training set can only reduce the amount of empty attribute space. If new training instances lie outside the boundaries of the space occupied by existing instances, the amount of empty space will be reduced. This reduces the size of regions where grafting can potentially alter the class, which in turn reduces the opportunity for grafting to have an effect. Similarly, this will also decrease the likelihood of test instances occurring in empty training regions, thus reducing the number of test instances that can be affected by grafting.

The results confirm that grafting reduces variance. C4.5 had greater variance than grafting on 49 occasions, less on 34, and equal on two. The way in which grafting reduces variance can be explained as follows. As stated previously, variance occurs in regions to which models built using different training sets have assigned different classes. Grafting affects regions in which no training instances lie. These two types of regions can overlap when different concepts are extrapolated into the same empty region over different training sets. This overlapping can occur with similar training data if, for example, the order in which decision boundaries are inferred differs across training sets. Figure 3.30 shows how this can occur. The figure shows three concepts: * centered around (2,2); + centered around (5,5); and X centered around (6,1). T represents a potential test instance. The test instance lies in an area into which one of the concepts must be extrapolated.

It is possible that grafting's use of global evidence acts as a stabilising influence on extrapolated areas, causing the class assigned to these regions to be consistent over training sets. In this example, the test instance T would then be consistently classified as class *, resulting in variance reduction either by reducing error or converting variance to bias, depending on the correct class of T. This behaviour may also explain why the bias of grafting is often higher than default C4.5.

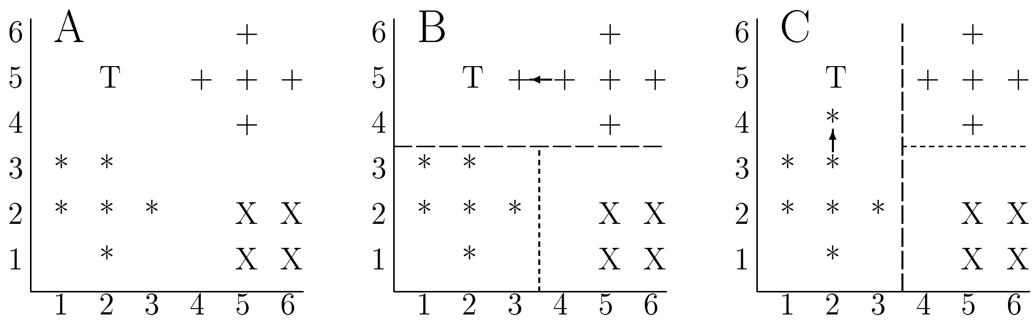

Figure 3.30: How small changes in data can affect extrapolation. Data sets B and C differ from data set A by movement of one instance only. Dashed lines represent the first decision boundary inferred, and dotted lines the second. Note that a small variation in data causes a change in assigned class in the area $x \leq 3.5, y \geq 3.5$.

The Effect of Increasing Representational Power

Table 3.5 shows results comparing the bias, variance, and total error of default OC1 and orthogonal OC1. Remember that due to the time taken to run OC1, these experiments were performed at the largest possible training set size only.

The results show orthogonal OC1 has higher bias than default OC1 on four data sets, and lower bias on only one. Variance of orthogonal OC1 is lower on five data sets, and higher on two. Although these results show no statistically significant trend, they suggest that increasing representational power will lower bias and raise variance.

The results for the Distinct Boundary data set are interesting. Distinct Boundary is the only data set for which both bias and variance are greater for orthogonal OC1 than default OC1. The underlying concept of this data set is $y = x$, and therefore should be approximated very well by default OC1. The increase in variance with orthogonal OC1 suggests that inferred decision boundaries change given different training sets. This can be expected from orthogonal OC1 as it must approximate the concept underlying this data set. However, the

Table 3.5: Comparison of bias and variance for default OC1 and orthogonal OC1

	Default			Orthogonal only		
Data Set	Bias	Variance	Error	Bias	Variance	Error
Cover Type	0.0269	0.0442	0.0711	0.0277	0.0395	0.0672
Distinct Boundary	0.0002	0.0003	0.0005	0.0004	0.0005	0.0009
Fuzzy Boundary	0.0011	0.0009	0.0020	0.0011	0.0009	0.0020
IPUMS	0.0568	0.0508	0.1076	0.0576	0.0500	0.1076
Random Binary	0.0000	0.0001	0.0001	0.0000	0.0004	0.0004
Shuttle	0.0002	0.0002	0.0004	0.0001	0.0001	0.0002
Sleep	0.1767	0.1556	0.3323	0.1853	0.1504	0.3357
Waveform	0.0087	0.0075	0.0162	0.0087	0.0073	0.0160

increase in bias with orthogonal OC1 also shows that even with decision boundaries changing across training sets, some instances are consistently misclassified.

Analysing the total error for each data set is also interesting. Orthogonal OC1 is more accurate than default OC1 on three data sets and less accurate on three, with two ties. It is possible that this is due to some data sets favouring orthogonal decision boundaries, although it must be remembered that default OC1 infers orthogonal decision boundaries as part of its induction process.

It may be that the occasional loss of accuracy of default OC1 is due to the number of parameters used to determine the split at a node. At each node in the decision tree, orthogonal OC1 must fit one parameter — the cut point of an attribute. Default OC1 requires fitting of $d + 1$ parameters, where d is the number of attributes. It is plausible that fitting these extra parameters may introduce error.

An alternate explanation is that allowing oblique cuts increases the chance of underfitting the data. Whereas overfitting occurs when too many decision boundaries are inferred, underfitting can occur when too few boundaries are

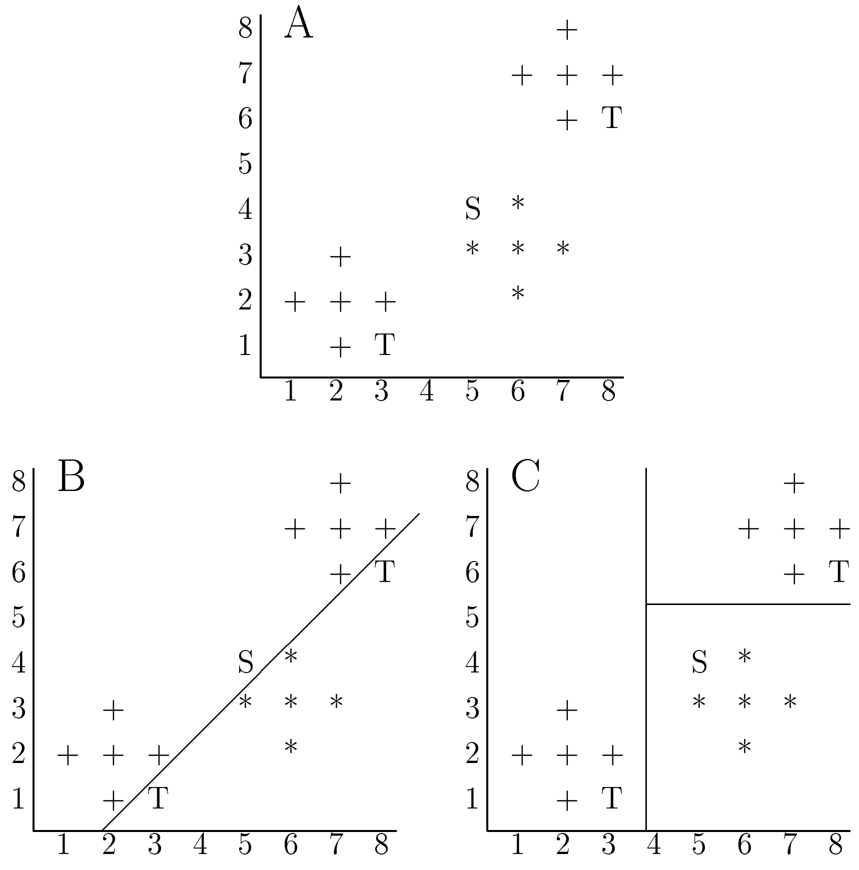

Figure 3.31: How oblique decision boundaries can cause underfitting.

inferred. Consider Figure 3.31.

Part A of the figure shows training instances of two classes from three distinct concepts. The two T's represent potential test instances of class +, and S represents a potential test instance of class *. Part B shows an oblique decision boundary inferred on the training instances. This boundary is optimal given the training data, and thus would be selected as the boundary to use. However, this boundary does not adequately represent the concepts, and results in misclassification of the test instances. Part C shows decision boundaries inferred using the same data, but with axis-orthogonal boundaries only. The test instances will be correctly classified.

This phenomenon could also explain why default OC1 had greater error for some data sets than orthogonal OC1.

3.2.4 Summary

The experiments investigated two potential methods of reducing bias when learning from large training sets. The results of experiments concerning decision tree grafting show that the difference in bias error between grafting and default C4.5 decreases as training set size increases. At the largest training set size, the bias of grafting is greater than that of default C4.5 more often than it is less. The results also call into question the ability of grafting to systematically decrease bias.

The results of experiments concerning representational power show that increasing representational power rarely relates to an increase in bias, and often relates to a decrease in bias. However, the results also show that increasing representational power leads to an increase in total error just as often as it leads to a decrease.

Due to the fact that the results do not show these methods to consistently reduce bias, the results do not offer enough evidence to draw conclusions as to whether an algorithm with lower expected bias produces more accurate models when learning from large training sets. However, the results do show that the behaviour of bias reduction mechanisms may alter with increased training set size.

It is reasonable to expect that an algorithm with extremely low variance management may produce models with extremely low bias. Such algorithms could be achieved by paying minimal attention to the data, creating a model almost totally randomly. However, the practicality of this method is questionable. An algorithm that minimises data analysis would require splits to be determined without the use of a measure of the utility of a potential split, as using such a measure would increase the chance of introducing bias. This then permits the possibility of creating trees where the number of leaf nodes equals the number of training instances. A stopping condition such as node purity cannot be used as this requires analysis of the data.

It would also be required that an algorithm have no limit on tree depth, as this could introduce bias toward smaller training sets. Therefore, building can only cease when a node contains a single instance, and further branching would be nonsensical. Each model would therefore classify every training instance correctly. Such an algorithm would produce unwieldy models at best. Pruning could not be employed, as this again requires analysis of training data. The lack of systematic generalisation may reduce bias, but is likely to result in each model performing poorly on unseen instances.

Previous research has shown that committees of semi-random models perform with accuracy similar to AdaBoost [12]. Additional research has been performed into randomising the model building process, taking different approaches toward introducing randomness [10, 27, 54, 2]. However, each of these methods use randomisation as only part of the inference process. Systematic methodologies are still employed, and thus these methods are not applicable for minimising bias.

Breiman [12] showed that the individual models forming committees are strong independently, where strength is a measurement of accuracy. Given the requirements of an algorithm that attempts to minimise bias through randomisation as discussed above, it is unlikely that the individual models discussed above will be strong. Boosting [41] has been shown to create a strong classifier from weak individual models by reweighting training instances according to correct or incorrect classification [73]. As noted above, bias minimisation models will have zero training set error, making Boosting unsuitable. It is therefore doubtful that committees of random trees will produce an acceptable classifier.

3.3 Conclusions

Experiments were performed to investigate the effect of training set size on variance and on the ratio of bias to variance. The results provide a clear indication that as training set size increases variance can be expected to decrease regardless of the expected bias plus variance profile of the classification learning algorithm. Bias can also be expected to decrease with larger training sets, although the reverse was true with one algorithm that has little bias management. The results show that as training set size increases it can be expected that bias will become a larger proportion of total error. Together, these results suggest that as training set size increases, bias management becomes increasingly important.

Experiments were also performed to investigate two methods of reducing bias. Results show that decision tree grafting reduces variance, but becomes less effective as a bias reduction mechanism as training set size increases. The results also indicate that grafting may reduce bias only for small training sets.

Results also show that increasing the representational power of an algorithm may be expected to reduce bias when using large training sets. However, this does not necessarily equate to a reduction in total error.

It is difficult to argue that these experiments are exhaustive. More and larger data sets would certainly provide more accurate estimations of bias and variance, but the benefit of this must be weighed against the extra execution time required. Experiments took up to 15 days to execute one algorithm on one data set at one training set size, and thus it was concluded that experimenting with more and larger data sets would currently be infeasible.

Even so, these experiments provide strong motivation for the development of algorithms with an emphasis on bias management rather than variance management, due to bias becoming a more dominant cause of error with larger training sets.

Chapter 4

A Comparison of Sampling Methodologies

The previous chapter investigated whether different types of algorithms are required when learning from large data sets. However, even with the strongest evidence that new algorithms may be necessary, such algorithms need to be developed and shown to have benefits beyond those of existing algorithms. Until then existing algorithms will continue to be employed. Thus, methods which make existing algorithms more practical must be researched.

Much research has been performed into "scaling up" existing algorithms so they can better handle large data sets. This includes parallelising algorithms, distributing algorithms, and investigating other ways in which algorithms can be made more efficient, thus reducing execution time. One simple way of reducing execution time is to reduce the amount of data input to the algorithm. Such data reduction can involve removing less relevant attributes or limiting the training set to include only a selection of instances. The most common way in which the latter is performed is with *sampling*, in which a number of instances are randomly or semi-randomly selected for inclusion into the training set. Sampling can also be performed within the inference process by using subsets of available data.

It is commonly believed that given large enough samples, sampling methodologies can produce models which represent the concepts underlying the data with an accuracy close to that obtainable without sampling, but with a significant decrease in execution time. Sampling can often achieve these purposes [84, 60, 99], but little is known about the trade-off a practitioner makes between the increase in error and decrease in execution time. Practitioners may benefit from having information regarding how different sampling methodologies affect this trade-off, and how sample size affects accuracy and execution time. Experiments were performed to clarify these issues for two popular sampling methodologies.

4.1 Definitions

Throughout this chapter a number of terms with specific meanings will be used. For clarity, they are defined here.

Whole data set - the full set of collected instances.

Full training set - the largest possible set of instances that can be used for training. Equivalent to the difference of the whole data set and the test set.

Training set - the actual set of instances input to an algorithm for inferring models. This may not be the full training set when using pre-sampling.

Available data - the set of instances available for inference at a particular time. Within a decision tree context, available data differs between nodes, and, at the root node, is equivalent to the training set.

4.2 Sampling Methodologies

It is a common occurrence that the amount of data available is more than enough to infer an acceptable model of the collected data [26, 42]. In situations where this is true, it is unnecessary to use all the data to infer a model. For example, if a data set contains two classes separated by a simple concept, it may be possible to infer adequate decision boundaries with only a small fraction of the data. However, reducing the amount of data increases the chance of error in the model as the precise decision boundaries required become less clear. This results in a trade-off between sample size and potential performance degradation. Regardless, sampling remains a popular means for reducing execution time of induction algorithms.

This section discusses two popular methods of performing sampling. For the purpose of this research, sampling is defined as representing a set of instances with a subset of those instances. Note that this excludes methods such as clustering and data squashing from consideration as sampling algorithms.

4.2.1 Pre-Sampling

The traditional view of sampling is of a subset of instances from the full training set being selected and input to the learning algorithm as the training set. A sample must be selected only once per inference process, resulting in little penalty in terms of increased execution time due to sampling overhead. The sample is selected before the chosen learning algorithm is initiated, and thus these methods can be viewed as sampling that is performed before the algorithm begins, or pre-sampling.

Instances may be selected with or without replacement from the full training set. Sampling may be performed totally randomly, so that every instance has equal chance of being selected, or with stratification, undersampling, or oversampling [74]. Stratification involves selecting a sample such that the class distribution of the data set is maintained in the sample. Undersampling evens the class distribution of the sample by removing instances of the majority class. Oversampling also evens class distributions, but does so by replicating instances of under-represented classes in the sample.

Benefits and Drawbacks

The justification for using pre-sampling is that if a sample is sufficiently representative of the full training set, then a model inferred from the sample should be similar to that which would be obtained from using the full training set. After taking the computational complexity of an algorithm into account, even a relatively small reduction in training set size can result in a significant saving of execution time. However, the pre-sampling methodology introduces a significant problem — determining an appropriate sample size such that the sample is sufficiently representative of the full training set while minimising sample size. This issue is discussed in Chapter 5.

The performance of algorithms that use a divide-and-conquer methodology, such as decision tree algorithms, can suffer greatly from pre-sampling. For example, consider a full training set containing 10,000 instances, and a sample of this set containing 500 instances. Given that many data sets used for experimental purposes in machine learning number only a few hundred instances, this might be considered a sufficiently large sample. Now consider a concept of this data that is represented in only one per cent of instances. In the full training set, this concept would be represented by 100 instances, but in the sample will be represented by only five. If the split chosen for the root node divides these five instances, it is unlikely that this concept will be detected when inferring from the sample. However, if the same split is chosen with the full training set, it may still be possible for this concept to be detected.

Meta-algorithms such as boosting [41] and bagging [10] select instances for an inference round using pre-sampling, although boosting complicates the process by taking instance weights into account.

4.2.2 Sub-Sampling

An alternative way of performing sampling is to select samples at specified points throughout the inference process. For example, within a decision tree context a sample could be taken from the available data at each node. The split at the node can then be determined using only the sample, resulting in a reduction in execution time. Once the split is selected, all instances of the node's available data may then be passed down the node's branches for the learning process to continue.

The effect of sub-sampling is to impose an upper limit on the number of instances used for inference at any time in an algorithm. If the number of instances in the available data set is less than the desired sample size, all of the available data may be used. As with pre-sampling, sub-sampling may be performed with or without replacement, and with or without stratification, oversampling, or undersampling.

Benefits and Drawbacks

Because the full training set is input, the number of instances available at the start of the inference process is greater with sub-sampling than pre-sampling given the same sample size. This continues to be true later in the process after a number of divisions, and delays the effect of fragmenting data due to the divide-and-conquer paradigm. For example, reconsider the example given for pre-sampling, where the full training set contains 10,000 instances with a sample size of 500 instances. Assuming all splits are binary, each node of depth four will have an average of 31.25 instances using pre-sampling. Using sub-sampling, each depth four node will have an average of 625 instances, and can still use 500 for inference. This may allow sub-sampling to select better splits lower in the tree.

Sub-sampling may also be able to build more descriptive models than pre-sampling. Continuing with the above example, pre-sampling can reach a maximum depth of 10 before each node will contain less than one instance on average. This does not occur with sub-sampling until depth 14.

However, the advantages of sub-sampling come at the cost of execution time. More samples must be taken, and using more instances for inference and creating larger models also increase execution time.

Sub-sampling is used in the seminal CART algorithm [13], where samples are selected such that the number of instances represented in the sample by each class are as even as possible.

4.3 The Problem With Sampling

Sampling has long been used as an effective method of reducing execution time. In some situations, sampling may be necessary before learning, as the amount of data available may be too great for a model to be built in a reasonable time with all the data. A different situation may require that a model be built as quickly as possible, with accuracy being of secondary importance. There may also be situations where accuracy of inferred models is of primary importance, but reduced execution time would be a substantial bonus. Alternatively, descriptiveness of models may be the main issue.

Given these scenarios, it is reasonable to expect that the requirements of sampling may be different in different circumstances. It is therefore illogical to expect that one type of sampling will be best in all situations.

There has been much research performed into both pre-sampling (e.g. [104, 99, 35]) and sub-sampling (e.g. [1, 77, 31]). However, although many of these papers compare the method introduced in the respective paper to no sampling, to date a comparison between pre-sampling and sub-sampling has not been performed. Additionally, it is contended that in spite of recent research focusing more on sub-sampling than pre-sampling, pre-sampling is likely to be employed more often. This is due to the fact that pre-sampling is undoubtedly simpler, and does not require modification of existing algorithms.

It is reasonable to expect that a practitioner faced with a situation in which sampling should be considered would gain from having knowledge about the advantages and disadvantages of different types of sampling. There is little information about what sample size should be used to achieve certain accuracy, how pre-sampling and sub-sampling compare against each other, and what risks are taken when performing sampling. The purpose of this chapter is to provide a comparison between pre-sampling and sub-sampling, in terms of accuracy, model descriptiveness, and execution time. It is believed that this could be of use to practitioners using sampling, rather than relying on assumptions and guesses.

4.4 Experiments

Experiments were performed to investigate how pre-sampling and sub-sampling compare to each other and to learning from the full training set. Inferring models from the full training set without use of a sampling methodology will henceforth be referred to as *no sampling*. Comparisons were made regarding accuracy, model complexity, and execution time. Such information may be useful to practitioners by providing insight into the trade-off encountered when sampling. It is reasonable to expect that pre-sampling will be less accurate than sub-sampling, which will in turn be less accurate than no sampling. It is also reasonable to expect that pre-sampling will be quicker than sub-sampling, which will be quicker than no sampling.

4.4.1 Methodology

Following from the experiments of Chapter 3, experiments were performed using ten-times three-fold cross-validation, resulting in 30 models being built for each methodology for each data set. C4.5 was used as the induction algorithm as its structure lends itself to implementing the sampling methodologies investigated. As test data was required for evaluation, the full training set is taken as two-thirds of the whole data set.

Sampling Implementations

The implementation of sub-sampling used in these experiments follows that of CART [13], selecting fixed size disproportionate samples at every node, such that the number of instances represented in the sample for each class are as equal as possible. Pre-sampling selects a random fixed-size sample as the training set, without regard of class distribution, and without replacement.

As both sub-sampling and no sampling have recourse to the full training set during induction, both used the full training set for pruning. However, since the input to pre-sampling is reduced, pre-sampling used only the input data for pruning.

4.4.2 Sample Size

Sample sizes started at 1,000 instances and doubled to the largest size less than the full training set. Sample sizes differ from the previous chapter as training sets of 32 instances are of little use for these experiments. Although it is possible that sub-sampling could produce reasonable models with such a small sample size, it is unlikely that this will be true for pre-sampling. Therefore, a more natural number for the minimum sample size was selected.

4.4.3 Data Sets

All data sets used in Chapter 3 were used in these experiments.

4.5 Results

Graphs show accuracy, time to infer one model, and model complexity in relation to sample size. Results are averaged across the thirty runs. Each graph shows results for pre-sampling, sub-sampling, and no sampling. Since no sampling provides only one measurement per data set, it is presented as a horizontal line in the graphs.

The number of instances in the full training set is shown in the key. For the Adult data set this is 32,561 instances. Using the above described methodology for determination of sample sizes, the largest sample should then be 32,000 instances. However, no results were recorded for this sample size as such a small difference in training set size offers little toward the purpose of these experiments.

Note that as opposed to Chapter 3, graphs show accuracy rather than error. This is because one of the uses of pre-sampling is in determining learning curves, where standard practice plots accuracy.

4.5.1 Accuracy

Figures 4.1–4.11 plot, for each data set, accuracy of the sampling methods used, at different sample sizes, compared to no sampling.

The results show pre-sampling is often substantially less accurate than the full training set at the smallest sample size, but has similar accuracy with the largest sample size. Pre-sampling reaches the accuracy of no sampling for only the Distinct Boundary and Random Binary data sets, and only at the largest sample size.

Sub-sampling also reaches accuracy comparable to no sampling, but often does so with smaller samples than pre-sampling. Of interest is that sub-sampling occasionally has greater accuracy than no sampling. As stated in the experiment description, it was expected that no sampling would have greater accuracy than sub-sampling. These results are therefore somewhat surprising.

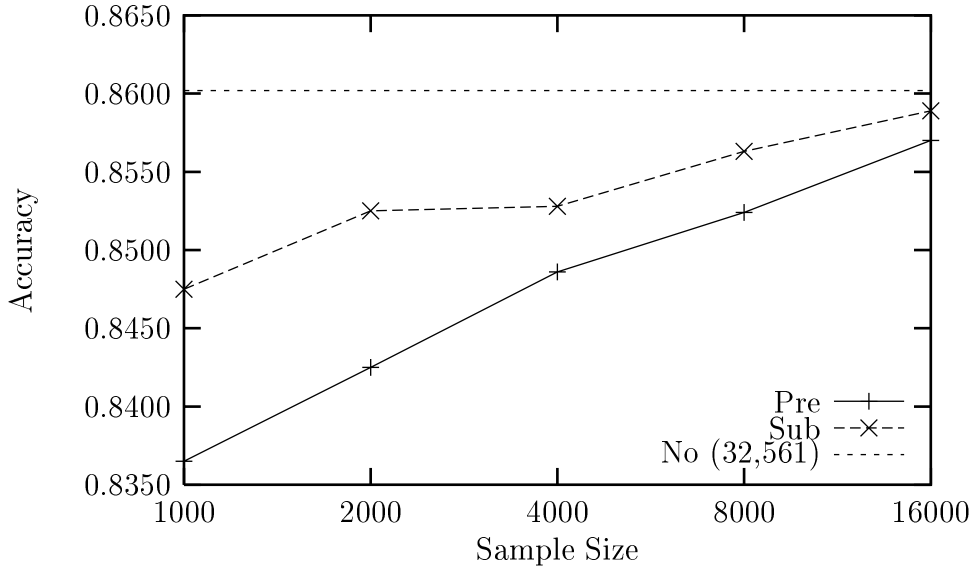

Figure 4.1: Accuracy on the Adult data set

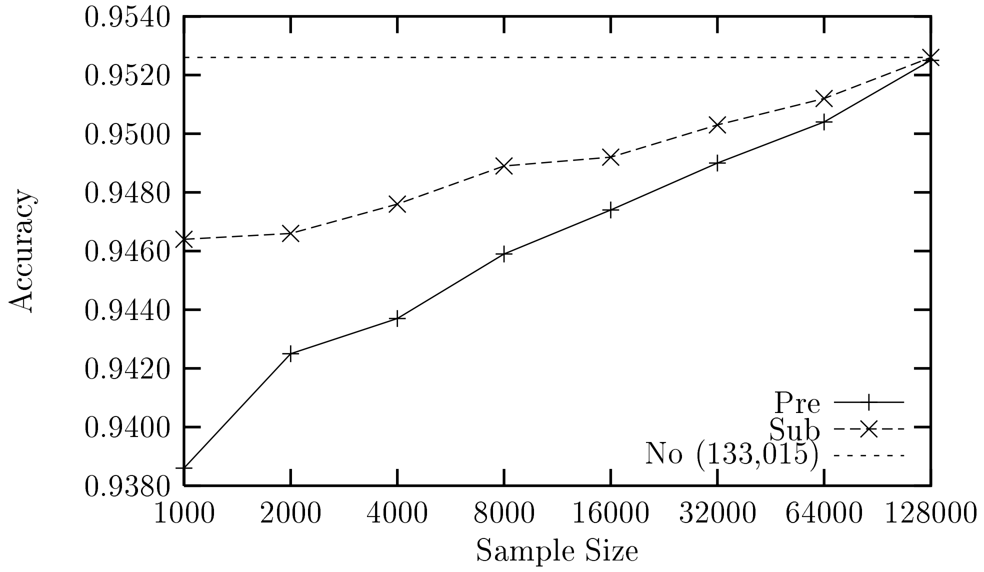

Figure 4.2: Accuracy on the Census Income data set

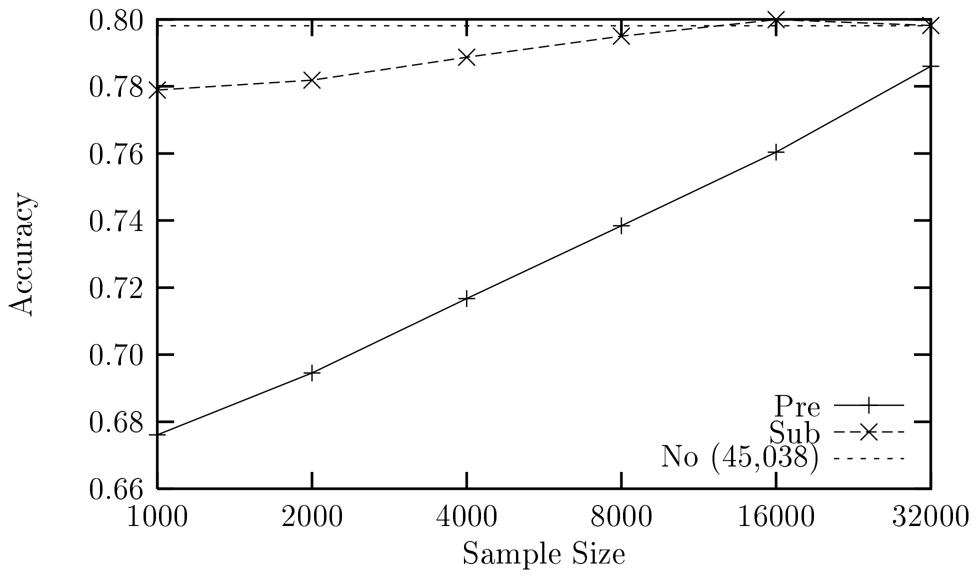

Figure 4.3: Accuracy on the Connect-4 data set

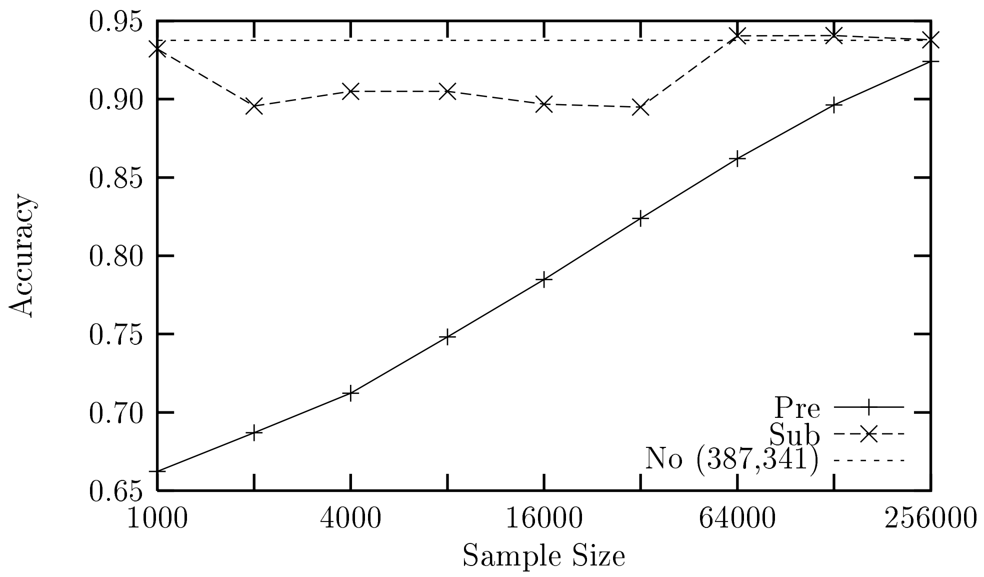

Figure 4.4: Accuracy on the Cover Type data set

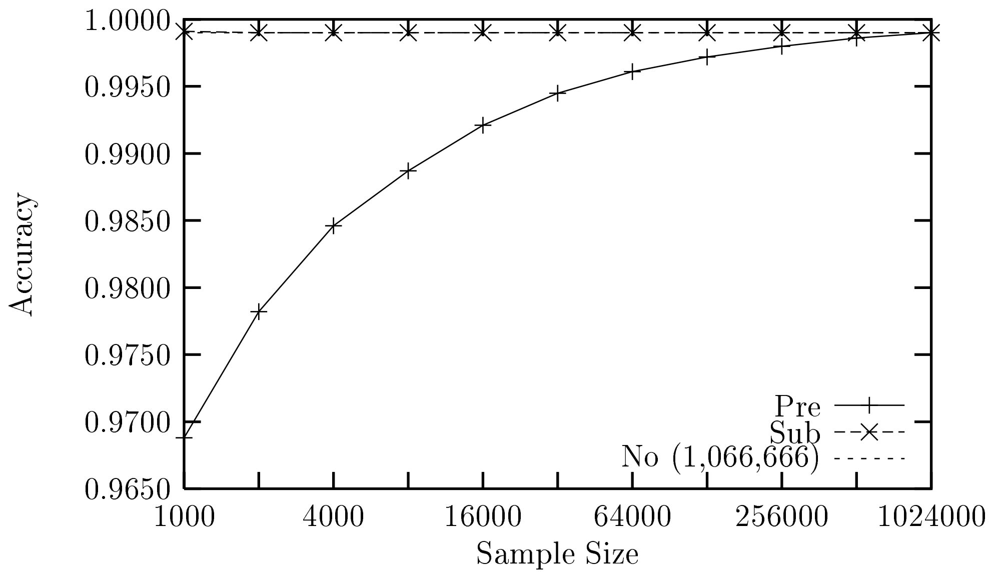

Figure 4.5: Accuracy on the Distinct Boundary data set

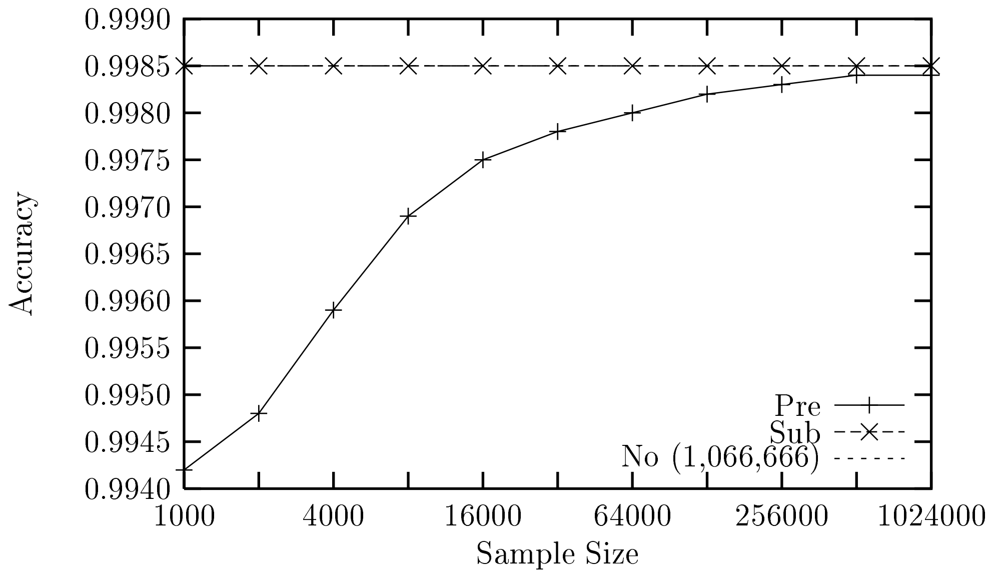

Figure 4.6: Accuracy on the Fuzzy Boundary data set

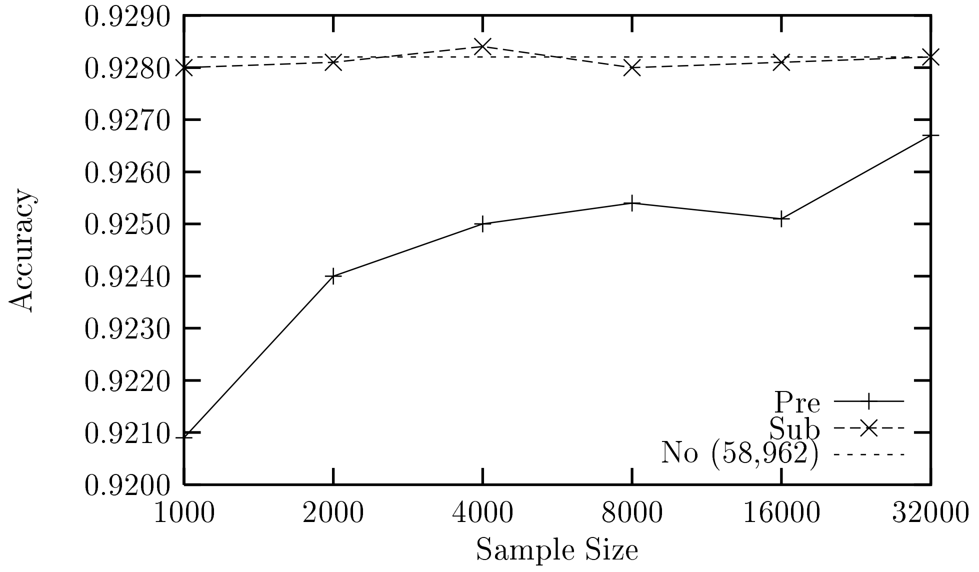

Figure 4.7: Accuracy on the IPUMS data set

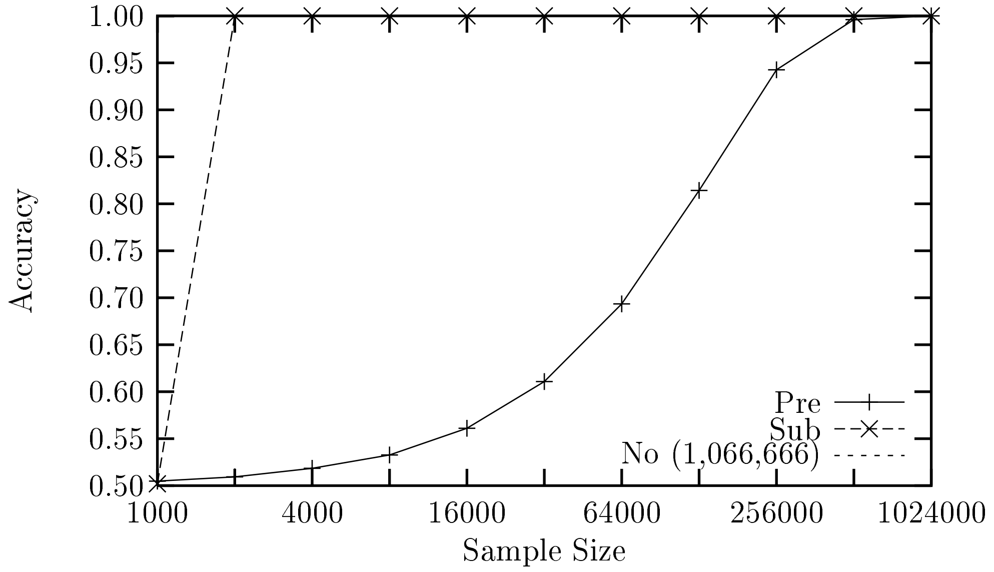

Figure 4.8: Accuracy on the Random Binary data set

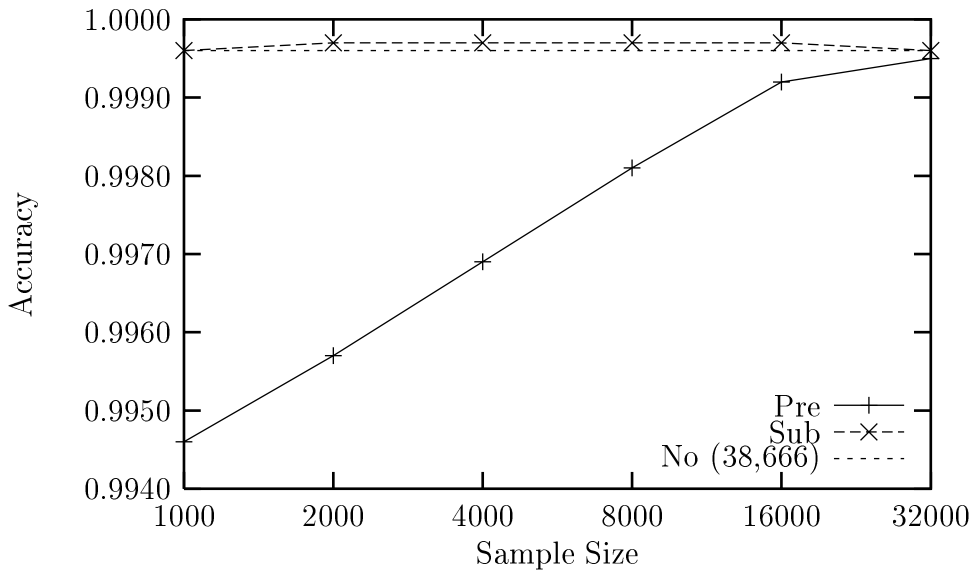

Figure 4.9: Accuracy on the Shuttle data set

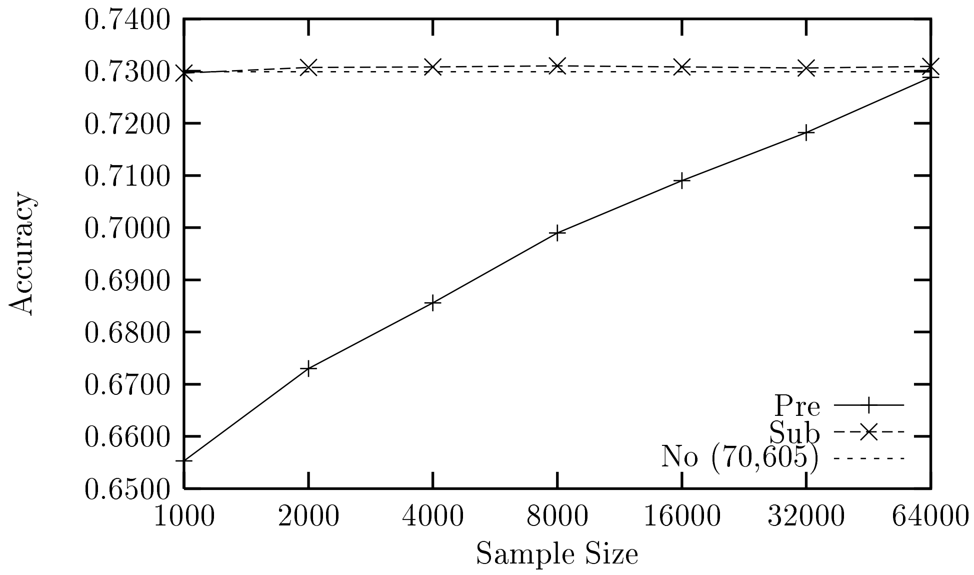

Figure 4.10: Accuracy on the Sleep data set

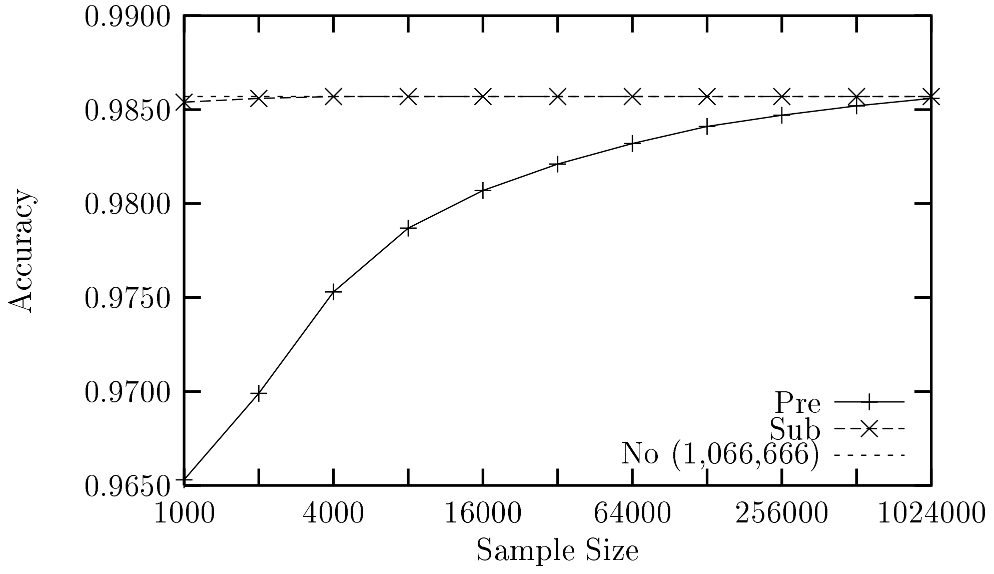

Figure 4.11: Accuracy on the Waveform data set

Table 4.1 shows the number of occurrences for each data set where sub-sampling had equal accuracy or greater accuracy than the full training set, and the smallest sample size after which this is consistently true. Of the 91 measurements taken, sub-sampling is at least as accurate as no sampling on 61 occasions.

It is interesting that this is often achieved with small sample sizes, and continues to be true for all larger sample sizes used. The strongest instance of this is the Sleep data set, where the accuracy of sub-sampling is greater than that of no sampling for all but the smallest sample size used. Also, there are only four data sets where sub-sampling is never more accurate than no sampling. The reason for this could be that sub-sampling may be a localised form of boosting. Boosting selects a set of instances from which a model will be inferred, with the set of instances selected being dependent on the performance of previously inferred models. This process is performed numerous times, forming a committee of models. Similarly, sub-sampling selects a set of instances from

Table 4.1: Number of times Sub-Sampling is at least as accurate as the Full Training Set

Data Set	Equal	Better	Size
Adult	0	0	—
Census Income	0	1	128,000
Connect-4	0	2	16,000
Cover Type	0	3	64,000
Distinct Boundary	10	1	1,000
Fuzzy Boundary	11	0	1,000
IPUMS	1	1	32,000
Random Binary	10	0	2,000
Shuttle	2	4	1,000
Sleep	0	6	2,000
Waveform	9	0	4,000
Total	43	18	

which a node will be inferred, again with the set of instances selected being dependent on previously inferred nodes. A number of nodes are combined into a tree. Although there are differences between sub-sampling and boosting, such as the calculation of instance selection probabilities with boosting, the similarity of sub-sampling to boosting may explain its high accuracy.

The aim of these experiments is not to provide insight into the optimal size with which to sample, but rather to compare the effects of pre-sampling to those of sub-sampling. Given this, and the expectation that sampling will impact accuracy, it may be more fruitful to provide an analysis of the sample size required to meet a given cost, as opposed to the cost of using a certain sample size. Therefore, a comparison of performance of algorithms within bounded accuracy losses was performed, where loss of accuracy due to sampling is considered within the bound if the accuracy of the sampling methodology is within a given percentage of the accuracy of no sampling. For example, if the accuracy of no sampling is 0.8, then given a five per cent margin a sampling methodology is considered within the bounded loss if its accuracy is no less than $0.8(1 - 0.05) = 0.76$. Such an analysis may provide insight into the relative costs of each sampling method. For example, rather than calculating the mean loss of accuracy for each sample size, it may be more useful to know at what sample size no more than a given accuracy loss can be expected.

Tables 4.2–4.4 show the results of this comparison for one, two, and five per cent bounded accuracy loss respectively. Wins represent the number of times the accuracy of a sampling method was within a bounded loss, and losses the number of times it was not. The smallest sample size after which the sampling method was consistently within these bounds is also given, along with the percentage of the full training set to which this size relates.

Sub-sampling provides accuracy within one per cent of accuracy of no sampling with a sample size of 1,000 instances on seven of the eleven data sets.

Table 4.2: Comparison of sampling methods to full training set with 1 per cent bounded accuracy loss

	Pre			Sub		
Data Set	Win:Loss	Size	% Full	Win:Loss	Size	% Full
Adult	2:3	8,000	24.6	4:1	2,000	6.1
Census Income	6:2	4,000	3.0	8:0	1,000	0.8
Connect-4	0:6	–	–	3:3	8,000	17.8
Cover Type	0:9	–	–	4:5	64,000	16.5
Distinct Boundary	7:4	16,000	1.5	11:0	1,000	0.1
Fuzzy Boundary	11:0	1,000	0.1	11:0	1,000	0.1
IPUMS	6:0	1,000	1.7	6:0	1,000	1.7
Random Binary	2:9	512,000	48.0	10:1	2,000	0.2
Shuttle	6:0	1,000	2.6	6:0	1,000	2.6
Sleep	1:6	64,000	90.6	7:0	1,000	1.4
Waveform	8:3	8,000	0.8	11:0	1,000	0.1
Total	49:42			81:10		

Table 4.3: Comparison of sampling methods to full training set with 2 per cent bounded accuracy loss

	Pre			Sub		
Data Set	Win:Loss	Size	% Full	Win:Loss	Size	% Full
Adult	3:2	4,000	12.3	5:0	1,000	3.1
Census Income	8:0	1,000	0.8	8:0	1,000	0.8
Connect-4	1:5	32,000	71.1	4:2	4,000	8.9
Cover Type	1:8	256,000	66.1	4:5	64,000	16.5
Distinct Boundary	9:2	4,000	0.4	11:0	1,000	0.1
Fuzzy Boundary	11:0	1,000	0.1	11:0	1,000	0.1
IPUMS	6:0	1,000	1.7	6:0	1,000	1.7
Random Binary	2:9	512,000	48.0	10:1	2,000	0.2
Shuttle	6:0	1,000	2.6	6:0	1,000	2.6
Sleep	2:5	32,000	45.3	7:0	1,000	1.4
Waveform	10:1	2,000	0.2	11:0	1,000	0.1
Total	59:32			83:8		

Table 4.4: Comparison of sampling methods to full training set with 5 per cent bounded accuracy loss

	Pre			Sub		
Data Set	Win:Loss	Size	% Full	Win:Loss	Size	% Full
Adult	5:0	1,000	3.1	5:0	1,000	3.1
Census Income	8:0	1,000	0.8	8:0	1,000	0.8
Connect-4	2:4	16,000	35.5	6:0	1,000	2.2
Cover Type	2:7	128,000	33.0	9:0	1,000	0.3
Distinct Boundary	11:0	1,000	0.1	11:0	1,000	0.1
Fuzzy Boundary	11:0	1,000	0.1	11:0	1,000	0.1
IPUMS	6:0	1,000	1.7	6:0	1,000	1.7
Random Binary	2:9	512,000	48.0	10:1	2,000	0.2
Shuttle	6:0	1,000	2.6	6:0	1,000	2.6
Sleep	4:3	8,000	11.3	7:0	1,000	1.4
Waveform	11:0	1,000	0.1	11:0	1,000	0.1
Total	68:23			90:1		

Pre-sampling can do this for only three data sets. Sub-sampling builds a model within bounded loss of one per cent for all data sets, whereas pre-sampling does not at any sample size for the Connect-4 and Cover Type data sets.

With bounded accuracy loss of two per cent, sub-sampling requires a sample of only 4,000 instances for all but one data set. Pre-sampling can build acceptable models for all data sets, but some tests require samples of substantial size.

With five per cent bounded accuracy loss, sub-sampling requires samples of 1,000 instances for all data sets but one. Pre-sampling still requires substantial samples for some data sets.

Random Binary is the only data set for which sub-sampling is not within 5 per cent bounded accuracy loss for sample size 1,000. At this sample size, both pre-sampling and sub-sampling perform only slightly better than chance on this data set. Both forms of sampling perform much better than chance with all other sample sizes and data sets.

The results for the Random Binary data set are also interesting from another perspective. As shown in Table 3.4, the Random Binary data set consists of 16 binary attributes, producing $2^{16} = 65{,}536$ distinct combinations. Sub-sampling reaches accuracy loss bounds of one per cent with a sample size of 2,000 instances. However, pre-sampling does not do so until 512,000 instances — substantially more than the number of unique instances in the data set. This highlights how sub-sampling can produce more accurate models with a smaller sample size. It must be remembered, though, that because a sample is taken at each node, sub-sampling considers more instances in total than pre-sampling with the same sample size.

The percentage of the full training set after which accuracy is consistently within a given bounded loss is interesting. Tables 4.2–4.4 show that only a small percentage of the full training set is required to build models within the given bounds. With pre-sampling, six of the eleven data sets require no more than three per cent of the full training set to build models with one per cent bounded accuracy loss, and two require less than one per cent. With two per cent bounded accuracy loss, four data sets require less than one per cent of the full training set, and nine require less than 50 per cent. With five per cent bounded accuracy loss, four data sets require less than one per cent of the full training set, and all require less than half. It is worth noting that of the five data sets that require greater than one per cent of the full training set to reach this bound, three of the data sets do so at the minimum sample size used in these experiments. If smaller samples were used, it is plausible that these data sets may also fall below the one per cent mark.

The results are in general similar for sub-sampling, although the sample size required for a given accuracy loss bound is often smaller than that for pre-sampling. With one per cent bounded accuracy loss, only two data sets require more than ten per cent of the full training set, and four require less than one per cent. With two per cent bounded loss, only one data set requires more than ten per cent of the full training set, and five require less than one per cent. With five per cent bounded loss, the maximum size required is 3.1 per cent, with six data sets requiring less than one per cent. As with pre-sampling, a number of data sets have reached the smallest sample size, and it is plausible that they may be able to reach one per cent of the full training set.

Summary

This section has compared the accuracy of models built using pre-sampling, sub-sampling, and no sampling. The comparison has shown that sub-sampling produces more accurate models than pre-sampling. The models built using sub-sampling also fall within bounded losses from the accuracy of no sampling more often and with smaller sample size than pre-sampling. That pre-sampling does

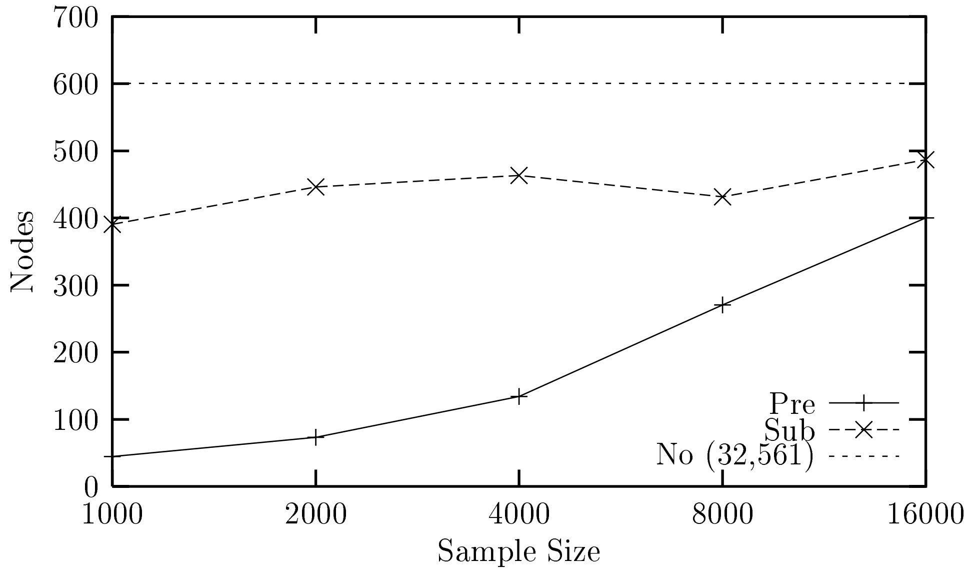

Figure 4.12: Mean number of nodes inferred for the Adult data set

not create models within one per cent bounded accuracy loss from no sampling for all data sets suggests it may be a risky methodology to use if accuracy is of utmost importance. Also, it is interesting that sub-sampling often produces models with accuracy at least as great as that of no sampling with small sample sizes.

4.5.2 Model Descriptiveness

Figures 4.12–4.22 plot, for each data set, the mean number of nodes inferred for the sampling methods used, at different sample sizes, compared to no sampling.

Figure 4.19 shows a large jump in model size for sub-sampling between 1,000 and 2,000 instances on the Random Binary data set. This is because at 1,000 instances both pre-sampling and sub-sampling inferred only one node. This is likely due to the nature of this data set, resulting in samples of this size containing insufficient information from which an appropriate split can be decided. This also causes the poor accuracy of both sampling methods for this data set

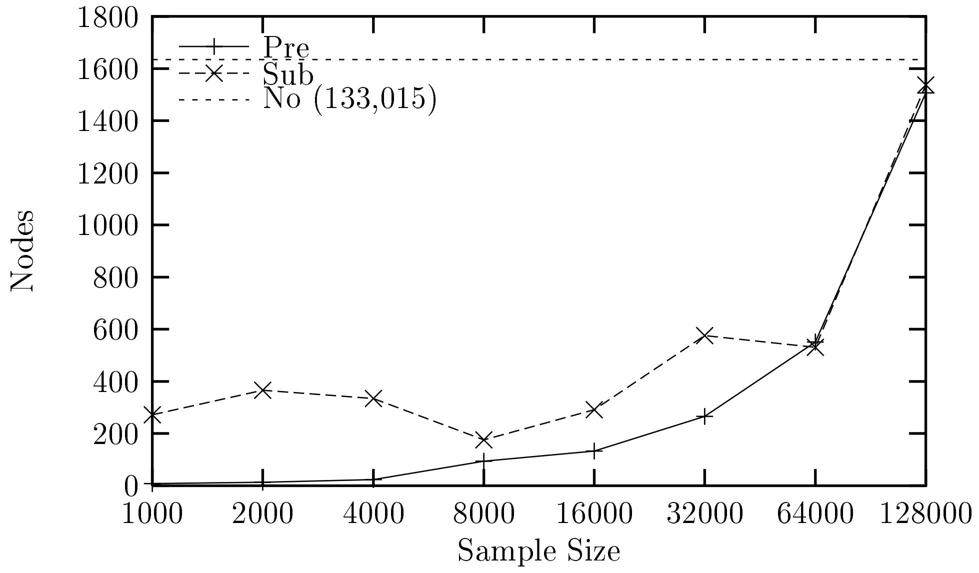

Figure 4.13: Mean number of nodes inferred for the Census Income data set

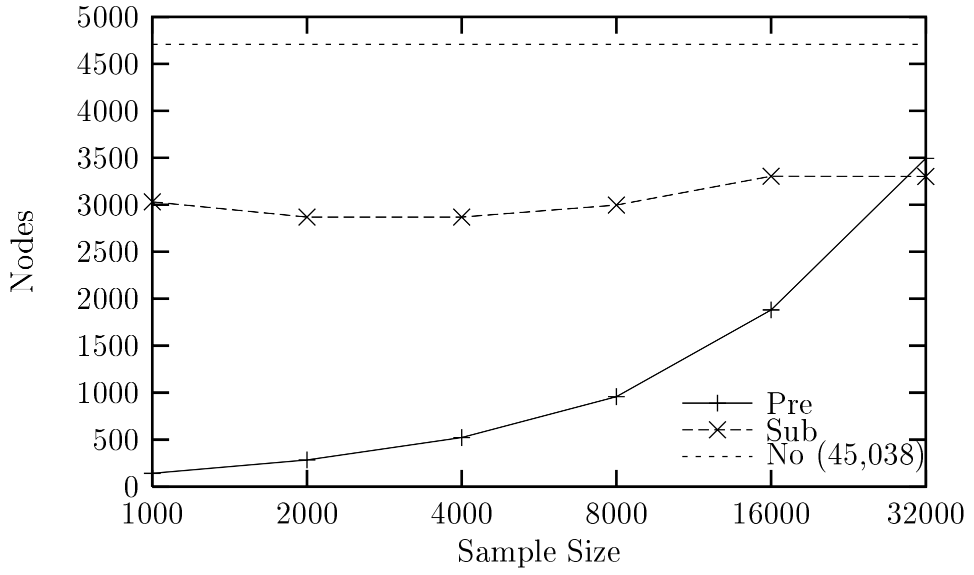

Figure 4.14: Mean number of nodes inferred for the Connect-4 data set

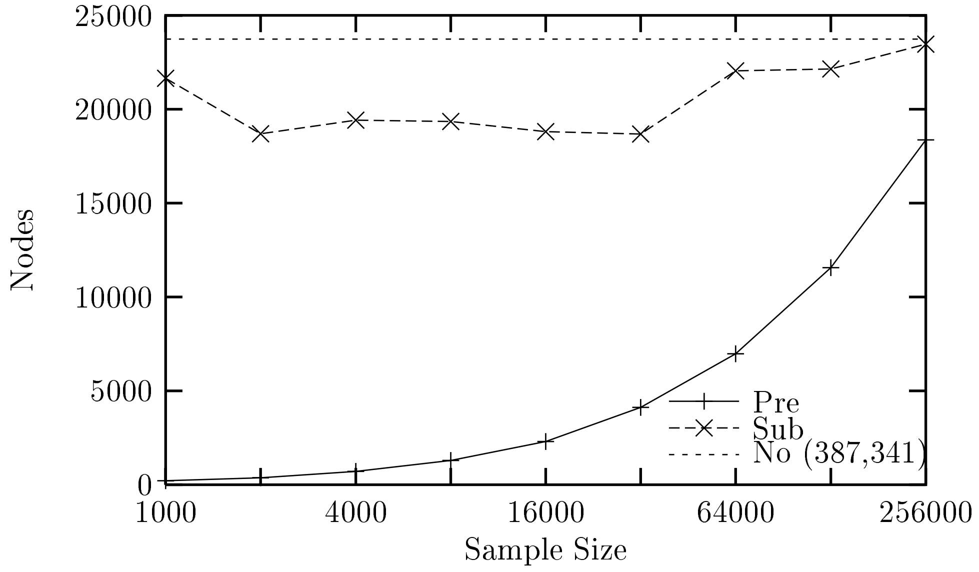

Figure 4.15: Mean number of nodes inferred for the Cover Type data set

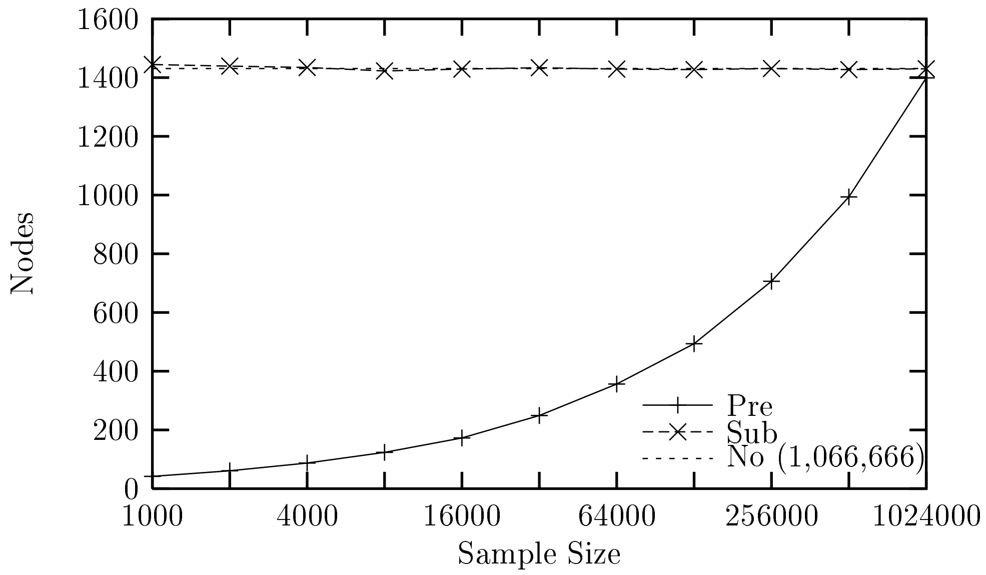

Figure 4.16: Mean number of nodes inferred for the Distinct Boundary data set

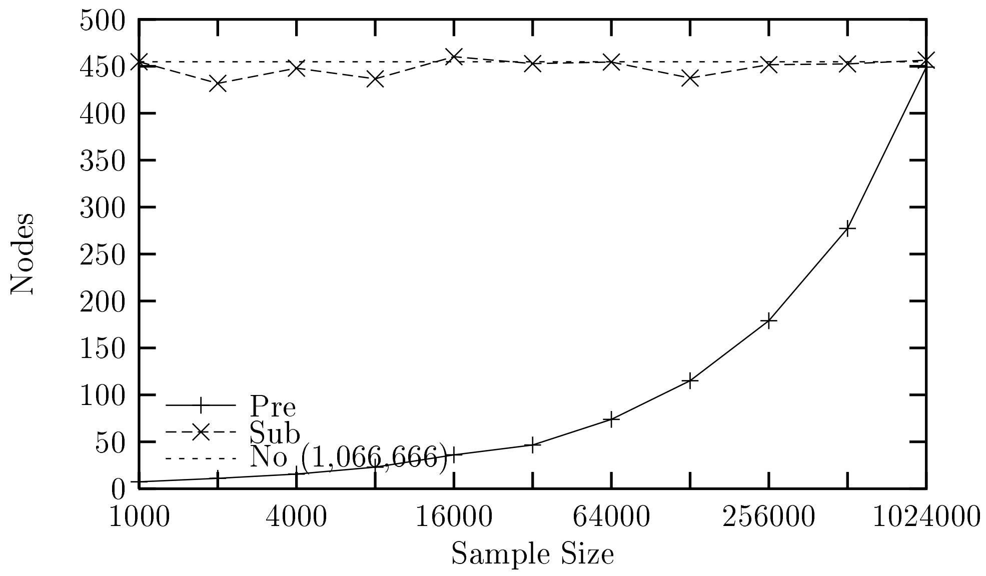

Figure 4.17: Mean number of nodes inferred for the Fuzzy Boundary data set

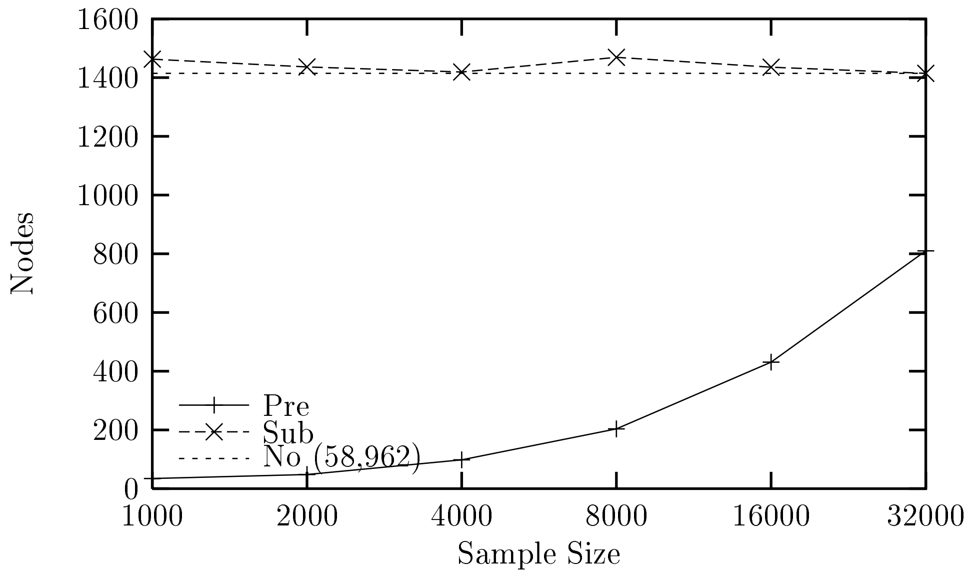

Figure 4.18: Mean number of nodes inferred for the IPUMS data set

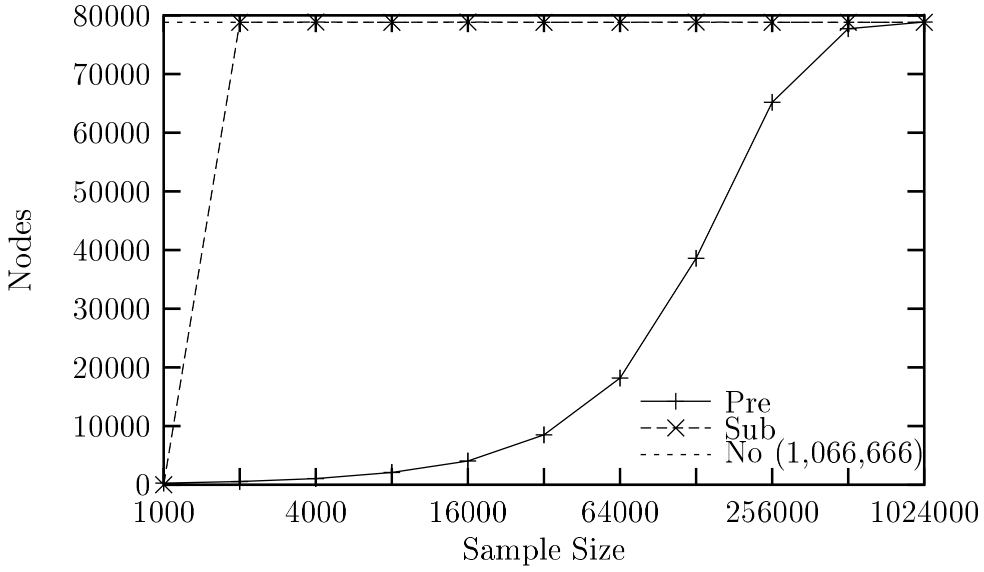

Figure 4.19: Mean number of nodes inferred for the Random Binary data set

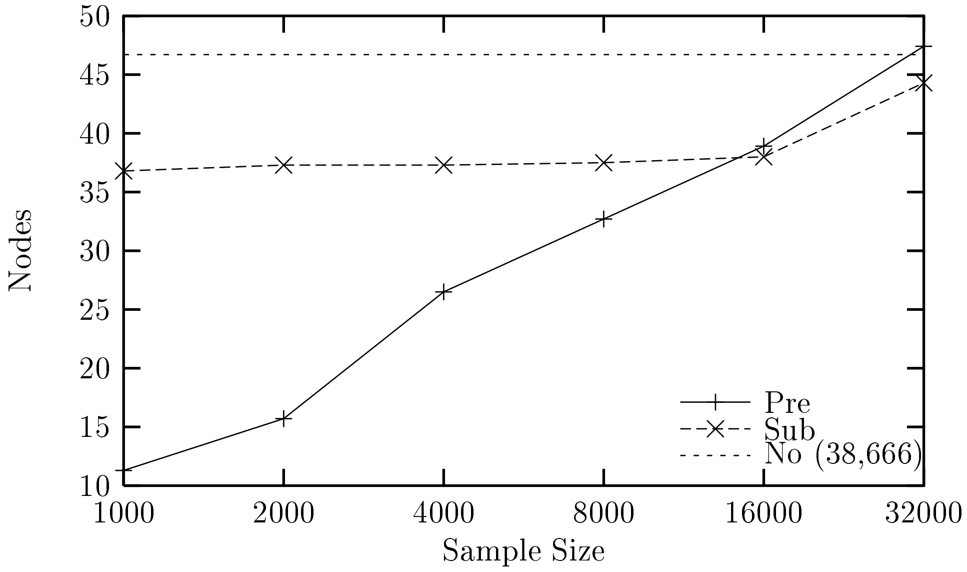

Figure 4.20: Mean number of nodes inferred for the Shuttle data set

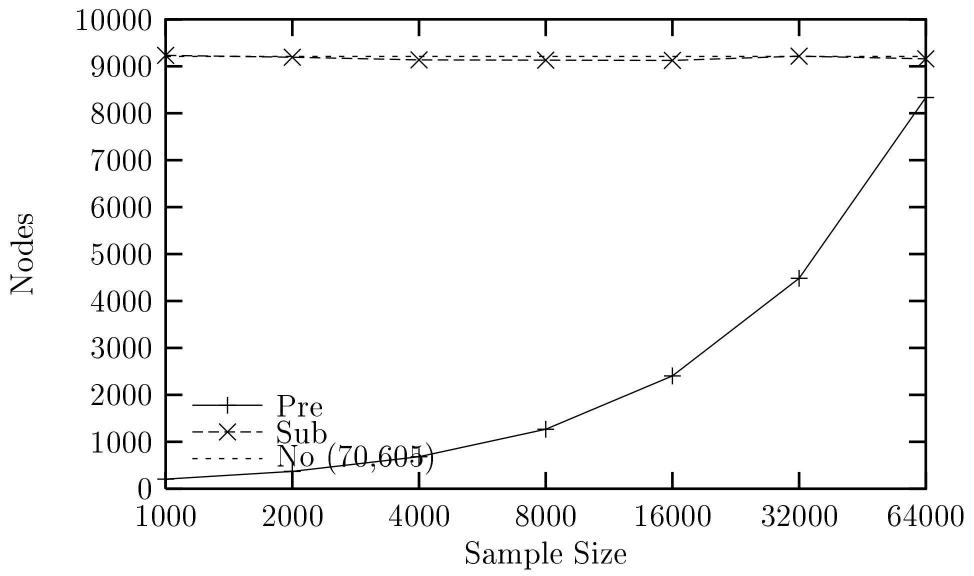

Figure 4.21: Mean number of nodes inferred for the Sleep data set

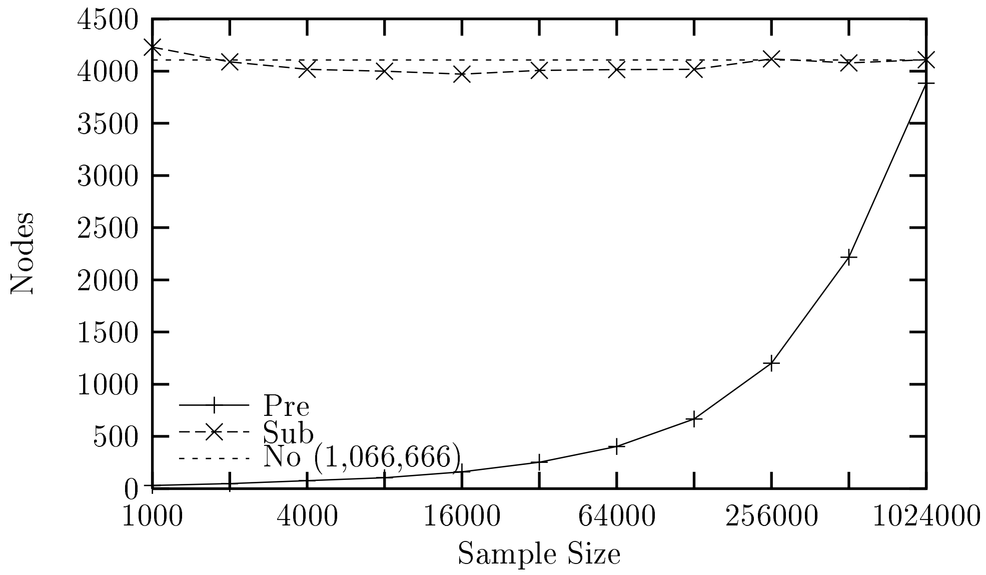

Figure 4.22: Mean number of nodes inferred for the Waveform data set

at this size. Note, however, that with a sample of 2,000 instances or more, sub-sampling results in a similar number of nodes to no sampling.

The results show that pre-sampling generally infers fewer nodes than no sampling. When this is not true it is always at the largest sample size. Pre-sampling generally infers fewer nodes than sub-sampling with the same sample size, with the few occasions where pre-sampling infers more nodes than sub-sampling occurring only with the two largest samples of a data set.

Sub-sampling often creates models with a similar number of nodes to no sampling, even with small samples. Of the 91 measurements taken, sub-sampling has fewer nodes on 69 occasions, and more on 22. This is statistically significant at the 0.05 level using a one-tailed binomial sign test ($p < 0.0001$). However, it must be noted that these measurements are taken after pruning has been applied, and that sub-sampling and no sampling use the same data (i.e the full training set) for pruning. An analysis of unpruned tree size shows that of the eleven data sets, sub-sampling infers larger trees seven times at the largest sample size, and six times at the smallest sample size. This suggests that sub-sampling has no consistent effect on the number of nodes inferred, but may build trees that are more susceptible to pruning.

Correlation Between Accuracy and Model Descriptiveness

The description of the experiments in Section 4.4 stated expectations that the accuracy of no sampling should be greater than that of either sampling method. The results of Section 4.5.1 show this to be generally true. Given that sub-sampling is occasionally more accurate than no sampling, it is worth investigating whether there is a correlation between model size and accuracy.

There are 22 occasions where sub-sampling infers more nodes than no sampling. Of these, the accuracy of sub-sampling is greater than that of no sampling only twice. The result of a one-tailed sign test ($p < 0.0001$) shows that this can be considered statistically significant evidence at the 0.05 level that sub-sampling loses accuracy when it infers larger models than no sampling.

For those occasions where sub-sampling has greater accuracy than no sampling, comparing the number of nodes inferred by no sampling and sub-sampling shows that sub-sampling infers fewer nodes on 14 occasions, and more on only three — statistically significant at the 0.05 level ($p = 0.0064$). Extending the analysis to include those occasions where accuracy is equal shows the trend becomes stronger (45:16, $p = 0.0001$). This suggests that when sub-sampling generates models at least as accurate as no sampling, those models will be less complex than those built without sampling.

There are 69 occasions where sub-sampling infers fewer nodes than no sampling. Comparing the accuracy of these occasions shows sub-sampling to produce models at least as accurate as no sampling 45 times, and models with less accuracy 24 times — statistically significant at the 0.05 level ($p = 0.0077$). This implies that when sub-sampling produces smaller models than the full training set, such models will generally not suffer from loss of accuracy.

Summary

The descriptiveness of models inferred by pre-sampling, sub-sampling, and no sampling has been compared. Sub-sampling generally produces models with similar descriptiveness to that of no sampling. Pre-sampling usually does this with only the largest sample size. Sub-sampling was also shown to generally benefit from increased accuracy when it infers fewer nodes than no sampling.

4.5.3 Execution Time

Figures 4.23–4.33 plot, for each data set, the mean time taken to build a single model for the sampling methods used, at different sample sizes, compared to no sampling. Measured execution time includes the time required to build and

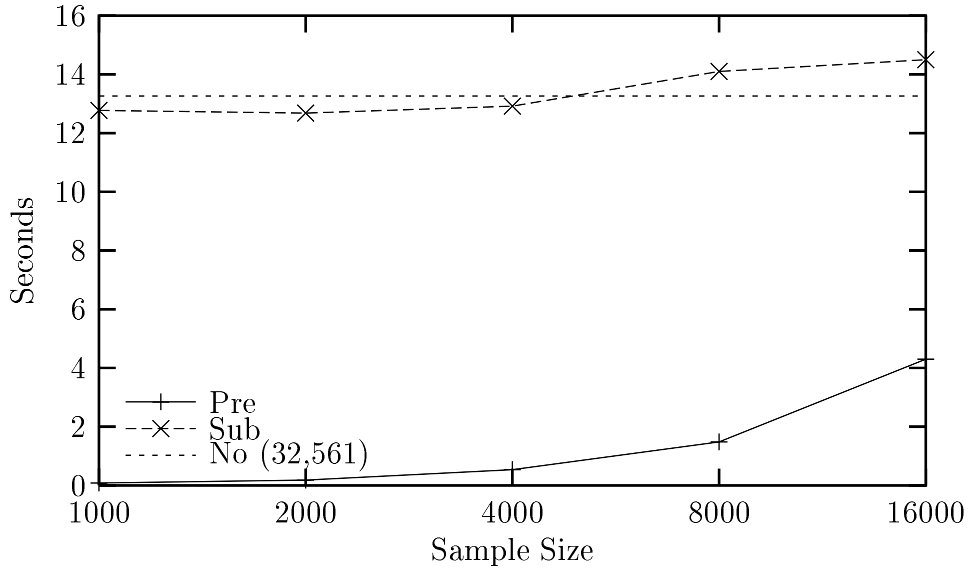

Figure 4.23: Mean model building time for the Adult data set

prune a model, and small associated overheads. The time to initialise data structures, read data from disk, and evaluate model accuracy is not included in this measurement.

Pre-sampling is always quicker than sub-sampling for the same sample size, and always quicker than no sampling. Sub-sampling is occasionally slower than no sampling, usually only with the two largest sample sizes.

An interesting phenomenon occurs with sub-sampling on the Connect-4 data set. Execution time for small samples is well above that of no sampling, but drops drastically with larger training sets. As can be seen in Figure 4.14, the number of nodes inferred does not significantly change. However, the structure of the trees does. An investigation of the unpruned trees sub-sampling inferred for this data set revealed that trees built with samples of size 1,000 to 4,000 have maximum depth ranging between 37 and 40, with the highest concentration of nodes around depth 31 to 33. Trees built with a sample size of 8,000 have maximum depth 31, with the highest concentration of nodes around depth 20.

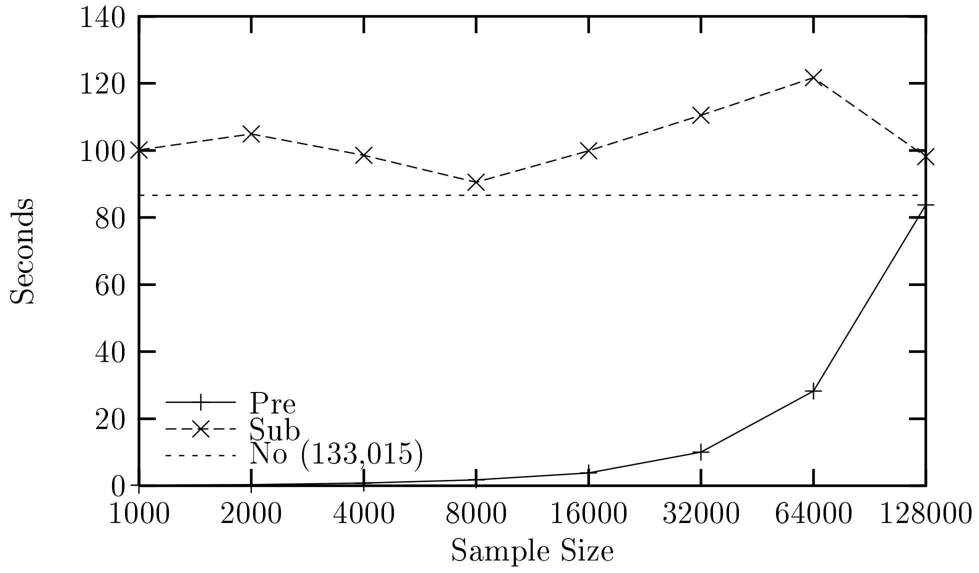

Figure 4.24: Mean model building time for the Census Income data set

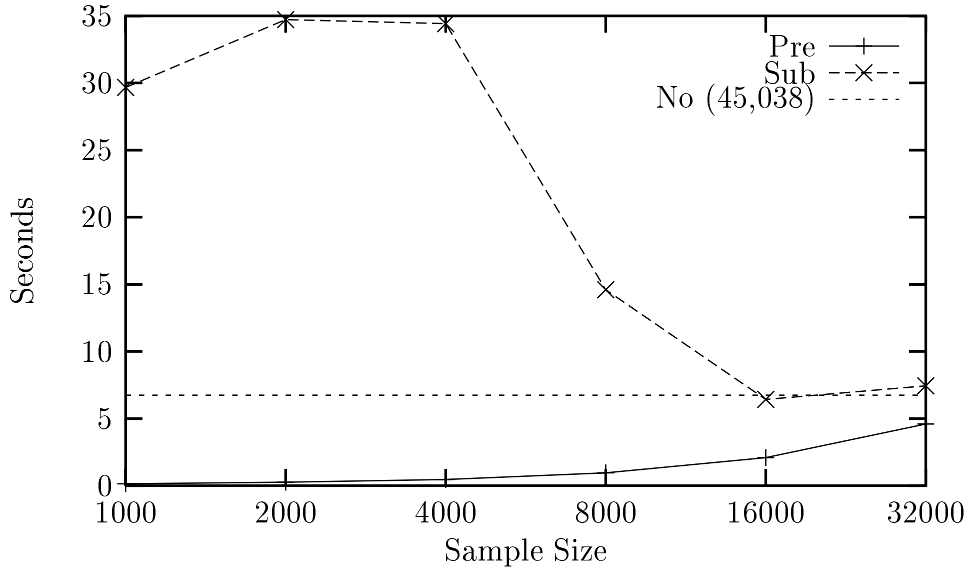

Figure 4.25: Mean model building time for the Connect-4 data set

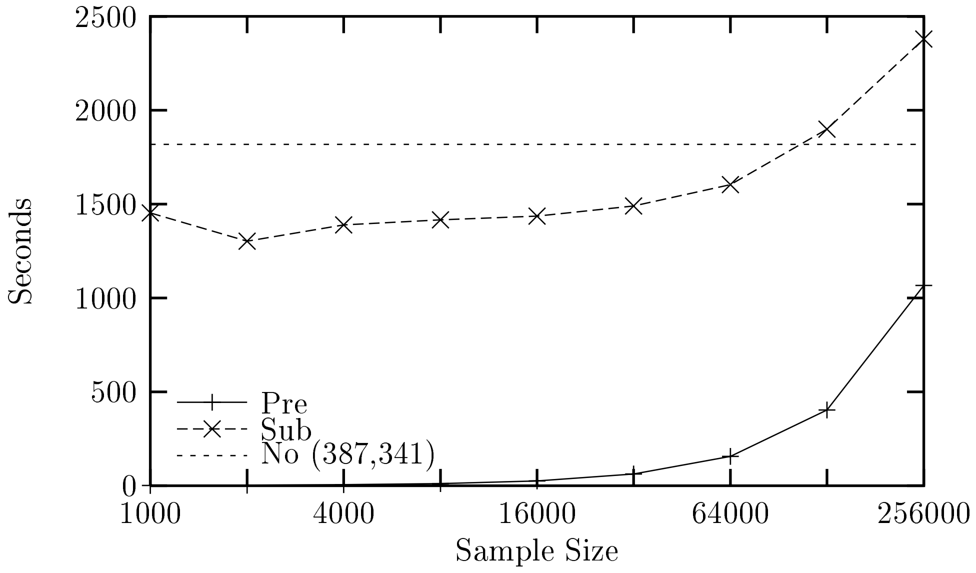

Figure 4.26: Mean model building time for the Cover Type data set

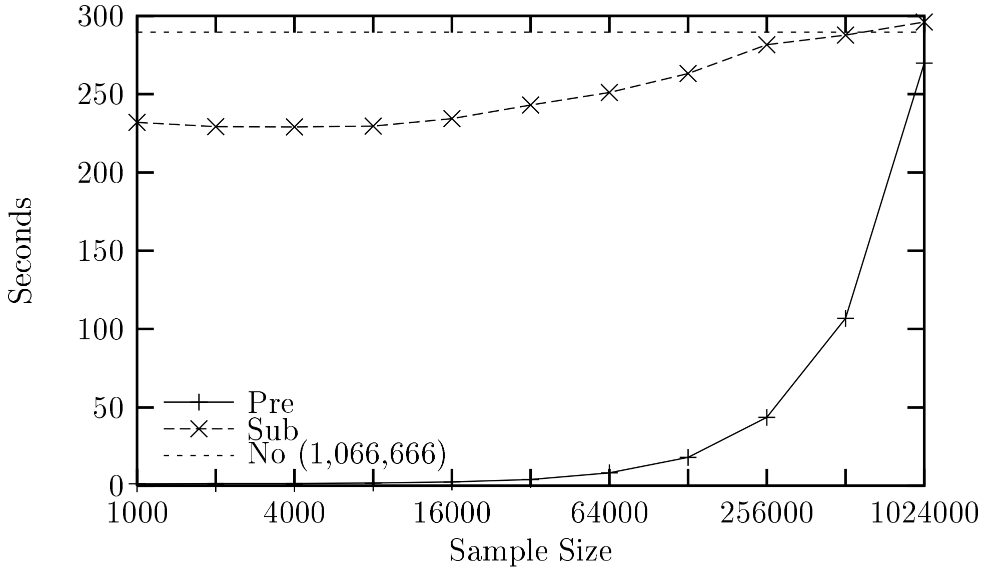

Figure 4.27: Mean model building time for the Distinct Boundary data set

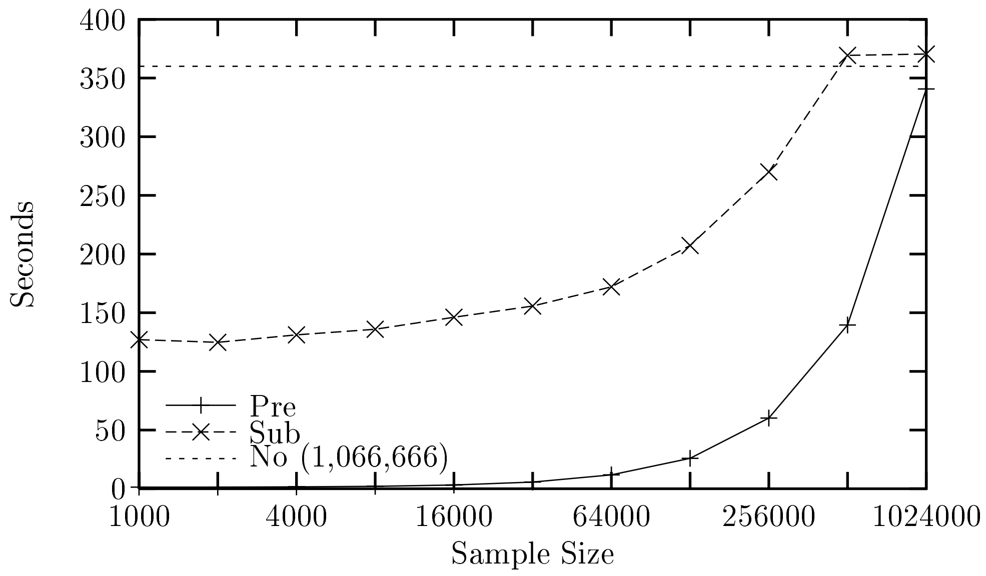

Figure 4.28: Mean model building time for the Fuzzy Boundary data set

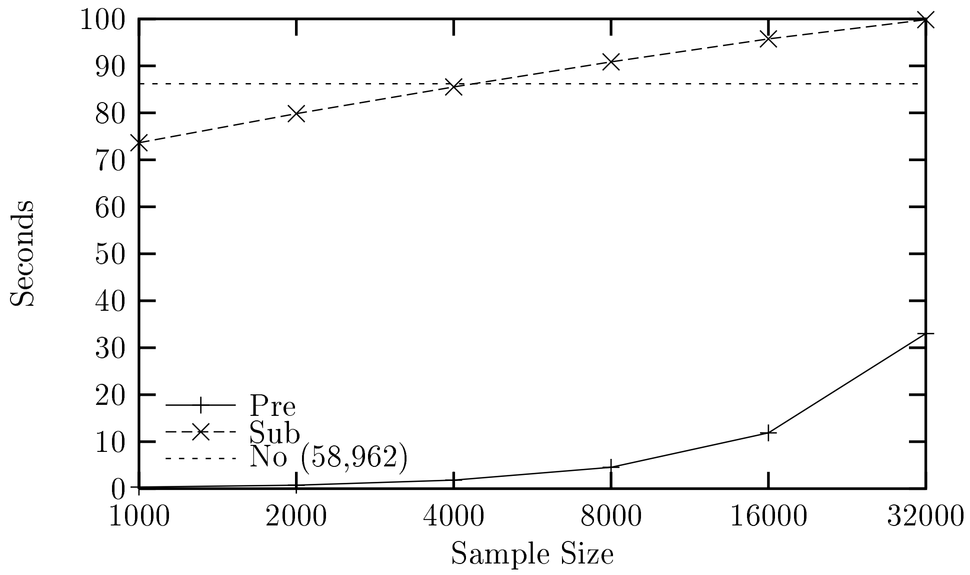

Figure 4.29: Mean model building time for the IPUMS data set

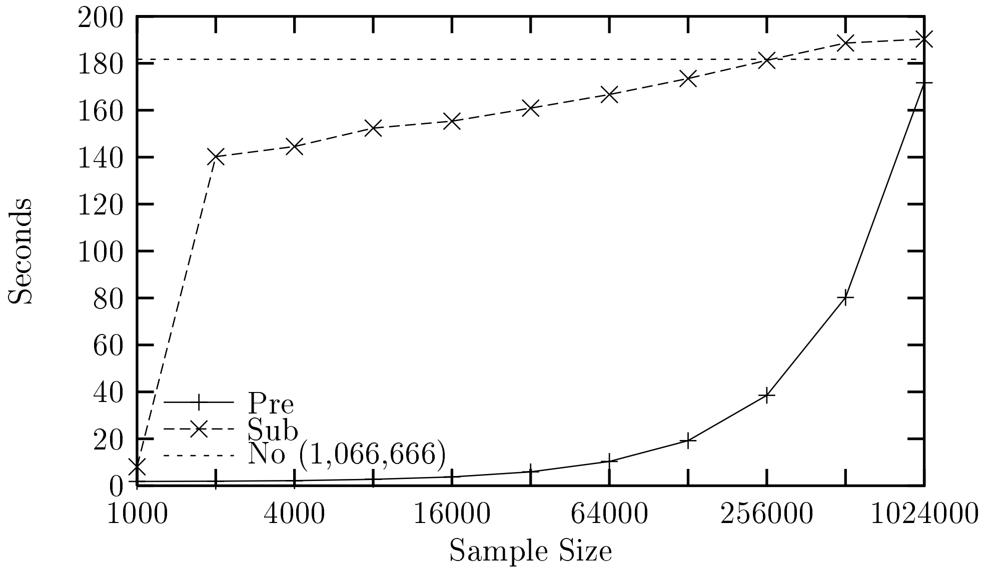

Figure 4.30: Mean model building time for the Random Binary data set

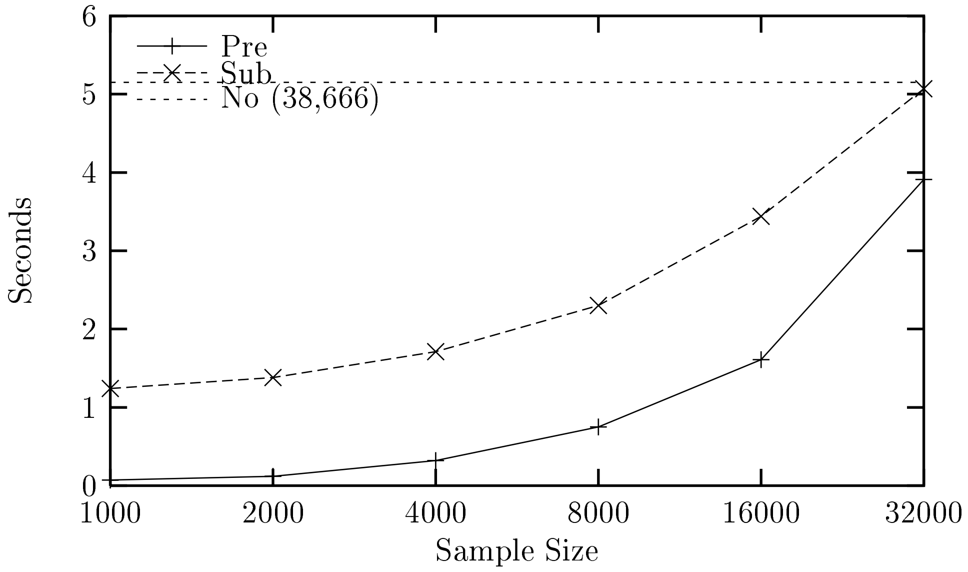

Figure 4.31: Mean model building time for the Shuttle data set

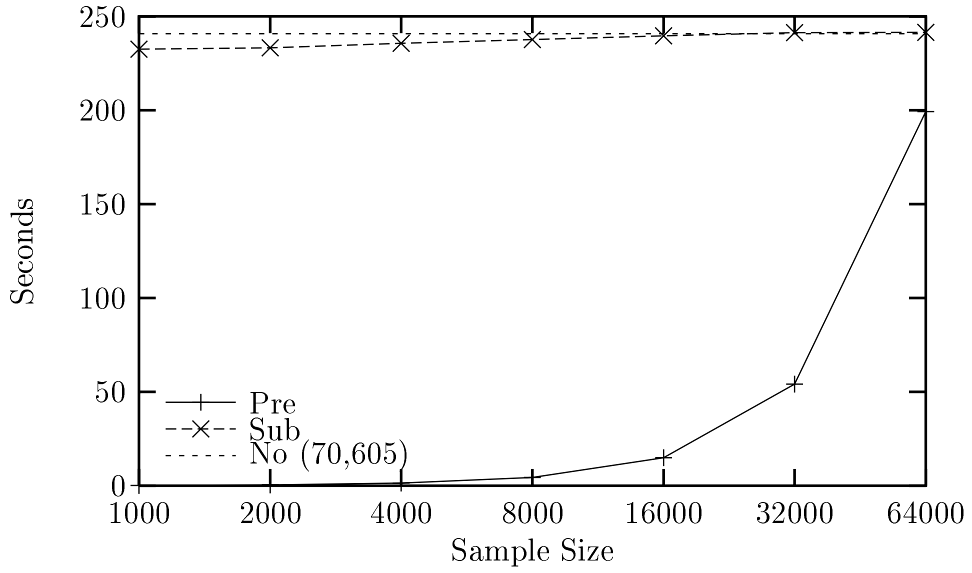

Figure 4.32: Mean model building time for the Sleep data set

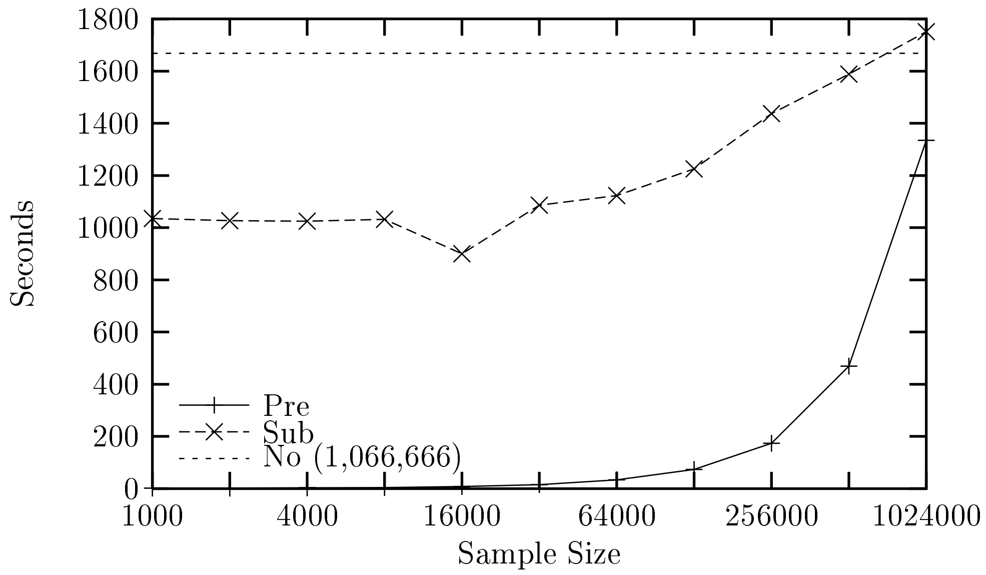

Figure 4.33: Mean model building time for the Waveform data set

Trees built with a sample size of 16,000 or 32,000 have maximum depth of 25, with the highest concentration of nodes around depth 15. This shows that trees built using larger samples tend to be broader than those with smaller samples. Although the total number of nodes inferred changes little, broader trees divide data earlier, resulting in nodes having fewer instances higher in the tree. After taking computational complexity into account, this can result in reduced execution time, as each node requires less computation.

Tables 4.5–4.7 show the percentage of execution time taken by pre-sampling and sub-sampling, relative to no sampling, for the smallest sample size after which the sampling methodology was consistently within a bounded accuracy loss. Note that many of the entries, especially for sub-sampling, are the same for differing levels of bounded accuracy loss. This is due to the fact that, for a number of data sets, accuracy reached differing bounded loss levels with the same sample size.

The time taken to build a model within a given bounded accuracy loss using pre-sampling is always less than the time taken to build a model with the same accuracy loss using sub-sampling. In fact, the time taken to build a model within one per cent bounded accuracy loss using pre-sampling is always less than that to build a model within five per cent bounded accuracy loss using sub-sampling. However, as shown previously in Table 4.2, it was not always possible to build a model within one per cent bounded accuracy loss using pre-sampling.

The time taken to build models within given accuracy loss bounds using pre-sampling is often an exceptionally small percentage of that taken for no sampling. With two per cent bounded accuracy loss, six data sets take less than 0.5 per cent of the time of no sampling. Only four take more than five per cent. Substantial amounts of time can be saved using pre-sampling, but the results do not impart overwhelming confidence that an acceptable model will be built.

The results for sub-sampling show it often requires a substantial amount of

Table 4.5: Percentage of the full training set required for one per cent bounded accuracy loss

	Time For	Pre		Sub	
Data Set	Full TS	Time	% Full	Time	% Full
Adult	13.26	1.48	11.2	12.68	95.6
Census Income	86.61	0.76	0.9	100.18	115.7
Connect-4	6.74	–	–	14.59	216.5
Cover Type	1818.31	–	–	1603.14	88.2
Distinct Boundary	289.65	2.47	0.9	232.02	80.1
Fuzzy Boundary	359.93	1.25	0.3	127.07	35.3
IPUMS	86.19	0.34	0.4	73.65	85.5
Random Binary	181.70	80.25	44.2	140.20	77.2
Shuttle	5.15	0.07	1.4	1.24	24.1
Sleep	240.8	199.35	82.8	232.56	96.6
Waveform	1668.54	4.39	0.3	1035.46	62.1

Table 4.6: Percentage of the full training set required for two per cent bounded accuracy loss

	Time for	Pre		Sub	
Data Set	Full TS	Time	% Full	Time	% Full
Adult	13.26	0.54	4.1	12.77	96.3
Census Income	86.61	0.19	0.2	100.18	115.7
Connect-4	6.74	4.59	68.1	34.43	510.8
Cover Type	1818.31	1066.29	58.6	1603.14	88.2
Distinct Boundary	289.65	1.41	0.5	232.02	80.1
Fuzzy Boundary	359.93	1.25	0.3	127.07	35.3
IPUMS	86.19	0.34	0.4	73.65	85.5
Random Binary	181.70	80.25	44.2	140.20	77.2
Shuttle	5.15	0.07	1.4	1.24	24.1
Sleep	240.8	54.13	22.5	232.56	96.6
Waveform	1668.54	1.85	0.1	1035.46	62.1

Table 4.7: Percentage of the training set required for five per cent bounded accuracy loss

	Time for	Pre		Sub	
Data Set	Full TS	Time	% Full	Time	% Full
Adult	13.26	0.08	0.6	12.77	96.3
Census Income	86.61	0.19	0.2	100.18	115.7
Connect-4	6.74	2.09	31.0	29.68	440.4
Cover Type	1818.31	402.87	22.2	1452.58	79.9
Distinct Boundary	289.65	1.26	0.4	232.02	80.1
Fuzzy Boundary	359.93	1.25	0.3	127.07	35.3
IPUMS	86.19	0.34	0.4	73.65	85.5
Random Binary	181.70	80.25	44.2	140.20	77.2
Shuttle	5.15	0.07	1.4	1.24	24.1
Sleep	240.8	4.43	1.9	232.56	96.6
Waveform	1668.54	1.57	0.1	1035.46	62.1

time to build models with a given bounded accuracy loss, although the amount of time taken is more consistent than pre-sampling. There is often little difference in execution time between the smallest and largest sample sizes for a data set with sub-sampling, whereas pre-sampling shows large increases in execution time with larger samples.

With one per cent bounded accuracy loss, sub-sampling takes an average of 88.8 per cent of the time of no sampling. With two per cent bounded accuracy loss, sub-sampling takes an average of 115.6 per cent. With five per cent bounded accuracy loss, it takes 108.5 per cent. However, the unusual nature of these results is largely due to the anomalous results of the Connect-4 data set. Ignoring this data set changes these results dramatically. With one per cent, the average drops to 76.0 per cent; with two per cent the average is 76.1 per cent; with five it is 75.3 per cent. The increase between the average for bounded accuracy losses of one and two per cent is due to a small decrease in accuracy for the Adult data set between samples of 1,000 and 2,000 instances. These results suggest some confidence can be had in the ability of sub-sampling to build models within a given accuracy loss of no sampling while reducing execution time.

4.5.4 Implications for Learning Curves

The results concerning pre-sampling for the Cover Type data set have an interesting implication for methods that employ learning curves to find an acceptably accurate model. Learning curves build models using an increasing number of instances until it is found that continuing the process will result in no or negligible increase in accuracy. At this point, the curve is said to have "converged." At the second largest sample size for the Cover Type data set, accuracy is 0.8963, and 0.9241 at the largest sample size. Given that this is close to a three per cent improvement in overall accuracy, it is unreasonable for learning curve methods to detect convergence before the largest sample size. However, the total time required to build models with the sample sizes used is substantial. The time taken to build all models using pre-sampling for the Cover Type data set using pre-sampling takes 95.4 per cent of the time to build a model using no sampling, with pre-sampling not having built a model within one per cent on the accuracy of no sampling. The results for the Connect-4 data set are direr. Building models for this data set takes 125.8 per cent of the time of no sampling, again without reaching one per cent accuracy loss. The Sleep data set is similar, taking 113.3 per cent of the time of no sampling, but one per cent accuracy loss is reached.

The results for these three data sets cast doubt over the general applicability of learning curve methods. Although the results show eight data sets where learning curves are likely to work very well, it cannot be known *a priori* that this will be the case. It is also interesting that Figure 4.7 shows the IPUMS data set has a decrease in accuracy when moving from a sample size of 8,000 instances to 16,000 instances. It is reasonable to expect that learning curve methodologies would detect this as convergence, as accuracy has not continued to improve. The large gain at 32,000 instances, which results in the most accurate models, would therefore remain unrealised.

Summary

The execution time of the sampling methods investigated was compared. Pre-sampling was shown to be faster than sub-sampling and no sampling. Sub-sampling is generally faster than no sampling — when it is not, it is usually with only the two largest sample sizes. When comparing the time required to build models within a given bounded accuracy loss from no sampling, pre-sampling results in substantial savings of execution time, and is always faster than building models with the same bounded loss with sub-sampling.

4.5.5 The Effect of Pruning Set Size

One potential reason for the difference between pre-sampling and sub-sampling is pruning. Pruning with pre-sampling was performed with only the pre-sample, while pruning with sub-sampling had access to the full training set. To investigate the effect of this disparity, results of models built with and without pruning were compared.

Accuracy

Tables 4.8 and 4.9 show the effect of pruning on the accuracy of pre-sampling and sub-sampling for the smallest and largest sample sizes employed respectively. The tables show that when pruned sub-sampling is more accurate than pruned pre-sampling, unpruned sub-sampling is also more accurate than unpruned pre-sampling. The sole exception is with the largest sample size for the Census Income data set. Note that, at the smallest sample size, pre-sampling models built without pruning are marginally more accurate than those built with pruning on the IPUMS, Sleep, and Waveform data sets. However, these unpruned accuracies are still less than those obtained with sub-sampling either with or without pruning.

The scale of the effect of pruning on the accuracies of pre-sampling and sub-sampling is generally similar. When there is little or no difference between the pruned and unpruned accuracies for a sampling methodology, there is generally a similar difference between the pruned and unpruned accuracies of the other sampling methodology. This suggests that accuracy results of previous sections are not dramatically affected by the use of different pruning sets between methodologies.

Table 4.8: Accuracy of pre-sampling and sub-sampling with and without pruning at the smallest sample size

	Pre-sampling		Sub-sampling	
Data Set	Pruned	Unpruned	Pruned	Unpruned
Adult	0.8365	0.8158	0.8475	0.8325
Census Income	0.9386	0.9306	0.9471	0.9352
Connect-4	0.6761	0.6414	0.7789	0.7664
Cover Type	0.6623	0.6502	0.9321	0.9312
Distinct Boundary	0.9688	0.9688	0.9991	0.9991
Fuzzy Boundary	0.9942	0.9942	0.9985	0.9984
IPUMS	0.9209	0.9026	0.9280	0.9073
Random Binary	0.5047	0.5052	0.5019	0.5019
Shuttle	0.9946	0.9947	0.9996	0.9996
Sleep	0.6553	0.6203	0.7296	0.6882
Waveform	0.9653	0.9654	0.9854	0.9853

Table 4.9: Accuracy of pre-sampling and sub-sampling with and without pruning at the largest sample size

	Pre-sampling		Sub-sampling	
Data Set	Pruned	Unpruned	Pruned	Unpruned
Adult	0.8570	0.8385	0.8589	0.8409
Census Income	0.9525	0.9421	0.9525	0.9409
Connect-4	0.7860	0.7720	0.7982	0.7902
Cover Type	0.9241	0.9236	0.9379	0.9374
Distinct Boundary	0.9990	0.9990	0.9990	0.9990
Fuzzy Boundary	0.9984	0.9984	0.9985	0.9984
IPUMS	0.9209	0.9060	0.9282	0.9070
Random Binary	1.0000	1.0000	1.0000	1.0000
Shuttle	0.9995	0.9995	0.9996	0.9996
Sleep	0.7288	0.6861	0.7309	0.6892
Waveform	0.9856	0.9655	0.9857	0.9856

Execution Time

The difference in the sets used for pruning between the two sampling methodologies could also affect reported execution times. Because sub-sampling uses the full training set for pruning, it is likely that any effect of this disparity will manifest itself as increased execution time for sub-sampling. If this is true, then it is possible that this is the cause of the substantially increased execution time of sub-sampling over that of pre-sampling. This can be investigated by comparing the execution time of sub-sampling without pruning to that of pre-sampling with pruning. If the execution time of unpruned sub-sampling is greater than that of pruned pre-sampling, then it is reasonable to conclude that the increase in execution time is not due to the size of the pruning set.

Table 4.10 shows the execution time of sub-sampling without pruning and pre-sampling with pruning. Measured time includes the time required to infer and, for pre-sampling, prune a model, along with small account keeping overheads. The time taken to read data from disk is not included.

Unfortunately, the system on which the earlier experiments were performed was not available at the time of these subsequent experiments. Therefore, these results are not directly comparable to those previously presented in the chapter. For example, on the Connect-4 data set, the time taken in this investigation to infer models using sub-sampling without pruning is greater than that taken to infer pruned sub-sample models in Section 4.5.3.

Comparisons of execution time are made at the smallest and largest sample size. These sizes are chosen as they can be expected to present the extremes of the effect of pruning set size. At the smallest sample size, the difference between the sample size and pruning set size of sub-sampling is the greatest, and, at the largest sample size, is least.

The results show that sub-sampling without pruning has greater execution time than pre-sampling with pruning. Although there is little difference in

Table 4.10: Execution times of pruned pre-sampling (PP) and unpruned sub-sampling (US) at the smallest and largest sample sizes

	Smallest		Largest	
Data Set	PP	US	PP	US
Adult	0.08	10.47	4.05	11.94
Census Income	0.27	185.65	86.16	187.14
Connect-4	0.12	2.71	4.44	5.94
Cover Type	1.41	1236.00	1047.40	1861.96
Distinct Boundary	1.29	219.33	269.72	275.23
Fuzzy Boundary	1.26	115.71	333.79	335.98
IPUMS	0.35	62.86	38.05	91.92
Random Binary	1.76	7.60	171.33	90.91
Shuttle	0.07	0.70	3.89	4.42
Sleep	0.20	228.47	199.99	237.84
Waveform	1.49	847.28	1336.40	1713.63

execution time at the largest sample size on the Fuzzy Boundary and Shuttle data sets, the execution time of sub-sampling is often substantially larger than that of pre-sampling. The only occasion where sub-sampling without pruning was faster than pre-sampling with pruning was the Random Binary data set at the largest sample size. These results suggest that the longer execution times of sub-sampling than pre-sampling reported in Section 4.5.3 are not due to the disparity in pruning set size, and are therefore due to the sampling methodology.

4.6 Conclusions

Experiments were performed to compare two sampling methodologies and no sampling, with respect to accuracy, model size, and execution time.

The results show sub-sampling produces models that are more accurate than pre-sampling. Sub-sampling also produces more descriptive models than pre-sampling. When models built with sub-sampling are at least as accurate as those built with no sampling, the sub-sampling models are generally more concise. This is may be due to sub-sampling building trees that are more susceptible to pruning.

When it is acceptable to build models with accuracy less than that obtainable without using sampling, the results suggest a practitioner could have high confidence in the sub-sampling methodology producing models within bounded accuracy loss of one, two, and five per cent. Pre-sampling also generally produces models within two and five per cent bounded accuracy loss, but the results do not suggest confidence can be had in the ability of pre-sampling to produce models with only one per cent bounded accuracy loss.

The cost of the extra confidence in sub-sampling is increased execution time. Sub-sampling often takes substantially longer to execute than pre-sampling, and occasionally longer than no sampling. However, at the minimum sizes required for two and five per cent bounded accuracy loss, sub-sampling takes less time to execute than no sampling, excepting the anomalous Connect-4 data set.

The results also show occasions where, if learning curves were used, pre-sampling would take longer to execute than no sampling. There is also an occasion where learning curve methodologies are likely to detect convergence, but then miss a substantial gain in accuracy with larger samples. This raises questions as to whether learning curves may contain many local maxima. Although local maxima do not often appear on a learning curve when viewed on a large scale, they may exist on a small scale. This could cause significant problems for many learning curve algorithms.

The results presented in Section 4.5.1 showed that it is possible to use sub-sampling to build trees that perform as well as no sampling using fewer instances at each node. However, the implementation used in these experiments employs a relatively expensive method of selecting a sample. It may be possible to decrease execution time by selecting samples using a less computationally complex method.

On the other hand, the fact that the implementation of sub-sampling uses a simple method of selecting instances, yet still occasionally has greater accuracy than no sampling raises the possibility that more complex methods of selecting a sample could potentially make sub-sampling even more accurate. The cost of this is likely to be increased execution time.

Neither sampling method provides overwhelming evidence of its ability to solve the problem of large data. Pre-sampling is likely to be more practical in terms of time, but comes at a significant risk of substantial accuracy loss. The fact that on two of the eleven data sets pre-sampling could not produce models with accuracy within one per cent of that obtainable with the full training set is concerning. Sub-sampling can produce highly accurate models with a small sample size, but comes at the cost of high execution time. Given that execution time is often a limiting factor in the application of learning algorithms, it is difficult to suggest that sub-sampling, regardless of its potential to boost accuracy, can be used with massive data sets. However, it must be remembered that sub-sampling has the distinct advantage that it does not require all instances to be stored in main memory. This can allow sub-sampling to learn from much larger data sets than methods that require all data be retained in core memory.

It must also be remembered that the methods used to select a sample differ between pre-sampling and sub-sampling. Pre-sampling selects instances totally at random without replacement, potentially skewing class distributions. Sub-sampling selects instances disproportionately, attempting to make the distribution in the sample as even as possible across classes. Although it is plausible that these differences in sample selection contribute to the differences between pre-sampling and sub-sampling shown in the results, it is unlikely that this is the main cause of the disparities. It is much more likely that the ability of sub-sampling to in effect use many more instances than pre-sampling during induction results in the differences between sampling methodologies.

Chapter 5

Improving Sub-Sampling

The experiments performed in Chapter 4 showed the accuracy of models built using sub-sampling to be generally greater and more reliable than that of models built using pre-sampling. Sub-sampling also had the significant benefit that, at times, it produced models with greater accuracy than those obtained when learning from the full training set without sampling. This fact alone makes sub-sampling a method worth exploring. However, sub-sampling does have some problems.

One of these problems is execution time. The results of Section 4.5.3 showed occasions where sub-sampling took longer to execute than learning from the full training set without sampling. This is obviously undesirable, and it is reasonable to expect there may be situations where the risk of greater execution time outweighs any benefits, such as increased accuracy, which could be obtained by using sub-sampling.

A second problem with sub-sampling is the use of the same sample size throughout the learning process. Although the results of Chapter 4 showed this is often effective, using a static sample size has a number of potentially detrimental implications. For example, it is possible that a much greater number of instances will be used for inference than necessary, resulting in needless expenditure of computational resources.

This chapter investigates three modifications to sub-sampling aimed at remedying these problems. The first modification investigates the effects of disproportionate sampling and non-replacement. Since both increase time required to sample, it is worth evaluating the effect each has on the performance of sub-sampling. This modification will also provide evidence regarding concerns raised in Chapter 4 that the results presented in that Chapter may have been confounded by the different sample selection strategies employed by pre-sampling and sub-sampling. The second and third modifications investigate ways of dynamically calculating sample size: one uses a varying proportion of available data, the other a statistical determination of a “sufficient” sample. Experiments are performed to empirically evaluate these modifications.

5.1 Definitions

Throughout this chapter a number of terms with specific meanings are used. Some of these have been previously defined in Section 4.1. The remainder are defined here.

Disproportionate sampling - selecting a sample such that instance counts for each class are kept as equal as possible.

Sub-sampling - any sampling methodology where a sample of instances is selected at various points throughout the learning process.

Standard sub-sampling - the implementation of sub-sampling where a fixed size, disproportionate sample are taken at every point in the algorithm. This is the implementation used both in CART [13] and Chapter 4.

5.2 Effects of Disproportionate Sampling and Replacement

One of the results of the Chapter 4 experiments involving standard sub-sampling was that it occasionally had greater execution time than learning from the full training set. Indeed, it could be argued that the purpose of sampling is to reduce execution time, and a methodology that cannot guarantee this is of limited utility. On the other hand, it could also be argued that it is highly desirable for a sampling methodology to build accurate models. Sub-sampling excelled at this. This then leaves the problem of how the execution time of sub-sampling can be reduced. However, before a remedy can be sought, the source of the problem must be identified.

5.2.1 Potential Factors Increasing Execution Time

Before examining potential problems with the learning algorithm, it is prudent to first eliminate any possibility that the results were influenced by factors unrelated to the learning algorithm. All experiments conducted in Chapter 4, except those noted in Section 4.5.5, were executed on the same hardware. Binary executables were created using the same compiler, with the same settings. Only the learning program and necessary system processes executed during experimentation. To confirm this, experiments were re-run, producing almost identical results. This then suggests that the occasional long execution time of sub-sampling is likely caused by problems with the learning algorithm, rather than external influences.

The implementation of sub-sampling used in this research was performed within the existing framework of C4.5. Thus, to convert standard C4.5 (no sampling) to sub-sampling requires only a small modification of the procedure used to build a node. The modification needs only to limit the data used to determine worth of a split to the selected sample. Such a small alteration is unlikely to significantly impact execution time of the induction process. However, it is possible that the sampling process itself is the cause of increased execution time.

There are two reasons why the sampling process might cause sub-sampling to have longer execution time than no sampling. The first is that models built using sub-sampling are not as broad as those built by no sampling. Broader trees result in data being divided earlier, and, since the computational complexity of finding a split is greater than O(n), greater division of data can be expected to reduce execution time. As discussed in Section 4.5.3, this appears to be the cause of the unusual behaviour of execution time on the Connect-4 data set.

However, making sub-sampling infer broader trees requires either introducing a learning bias toward broader trees, or enabling continuous attributes to be split into more than two intervals at each node. The first of these options defies the ethos followed many learning algorithms, including C4.5, of selecting the best possible split. The second requires a fundamental change in the base learning algorithm, and would result in no difference with domains consisting of only discrete attributes. Thus, forcing sub-sampling to build broader trees cannot be considered a viable option.

The second reason why sub-sampling could have longer execution time than no sampling is that the overhead associated with sampling may outweigh any reduction in execution time gained by using fewer instances to infer splits. This might be remedied by using a less computationally expensive method of selecting samples. In standard sub-sampling, disproportionate samples are selected, so that the class distribution of the sample is as uniform as possible. Sampling in this manner requires maintenance of lists of instances in each class, and calculation of the number of instances to be selected from each class. If the available data does not contain enough instances of a particular class to meet this requirement, then all instances of that class are selected, and the number of instances required for remaining classes recalculated. This process is repeated until all remaining classes can fill their quota.

Standard sub-sampling also performs sampling without replacement, requiring maintenance of a record of instances already selected for a sample. Although maintenance of such lists is an uncomplicated process, the need to continually update these lists can potentially cumulate to a substantial amount of time.

A simpler and likely quicker method than that used in standard sub-sampling is to select a sample without disproportionate class representation and with replacement. This eliminates the need to maintain records of selected instances and lists of instances in each class. It also eliminates the need for calculation and subsequent recalculation of the number of instances required for each class. This was studied by Guerts [51], where models built with samples taken with replacement were compared to models built with samples taken without replacement with smaller training sets. However, this disparity in training set size may bias results toward models built using replacement. Also, the experiments only investigated training sets up to 6,000 instances.

Such a modification can be expected to substantially reduce the overhead required for sample selection. However, it is plausible that part of the reason standard sub-sampling produces accurate models is due to the way in which samples are selected. Therefore, it is possible that making such a modification may have a negative impact on accuracy, and may even increase execution time if it causes thinner trees to be built.

5.2.2 Experiments

Experiments were performed to compare standard sub-sampling with three versions of sub-sampling with various combinations of the use of disproportionate sampling and non-replacement. These are:

- *Non-Replacement sub-sampling* — samples are selected randomly without replacement and without regard to class distribution.

- ***DisProportionate sub-sampling*** (DP)— disproportionate samples are selected with replacement.

- *Simple sub-sampling* — samples are selected with the least amount of overhead possible by selecting instances with replacement and without disproportionate sampling.

Note that standard sub-sampling selects disproportionate samples without replacement.

This is a 2x2 experimental design, comparing replacement against non-replacement and disproportionate against non-disproportionate sampling. Comparisons between the sub-sampling types are made in regard to execution time, model complexity, and accuracy.

The methodology and data sets used for these experiments are the same as those used in Chapter 4. To allow comparison between these experiments and the results of previous experiments, the sample sizes used in Chapter 4 are again employed.

5.2.3 Results

Graphs show accuracy and time to infer one model in relation to sample size. Results are averaged across the thirty runs. Each graph shows results for standard sub-sampling, simple sub-sampling, non-replacement sub-sampling, DP sub-sampling, and, as a baseline, no sampling.

Accuracy

Figures 5.1–5.11 plot, for each data set, accuracy of no sampling and standard, simple, non-replacement, and DP sub-sampling.

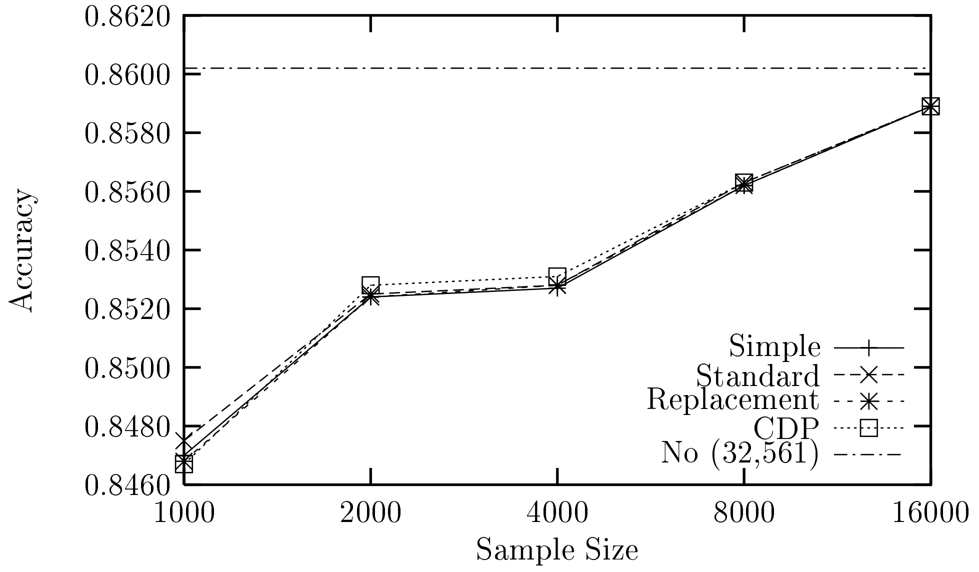

Figure 5.1: Accuracy of sub-sampling variants on the Adult data set

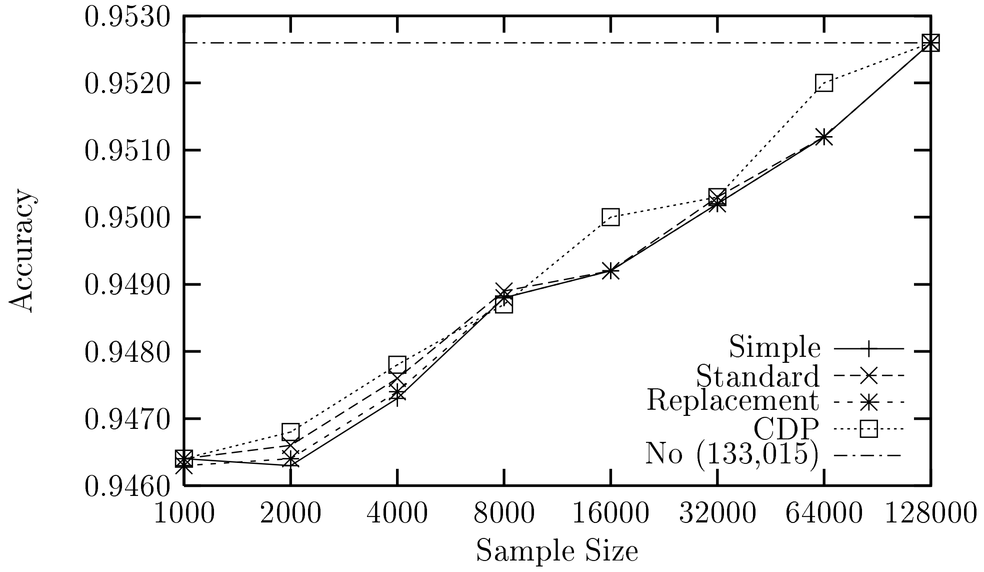

Figure 5.2: Accuracy of sub-sampling variants on the Census Income data set

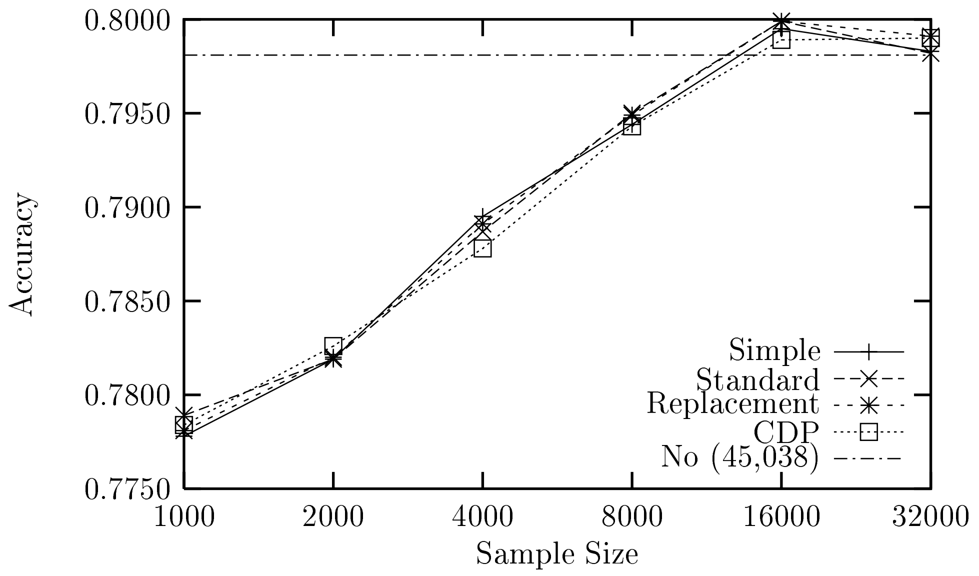

Figure 5.3: Accuracy of sub-sampling variants on the Connect-4 data set

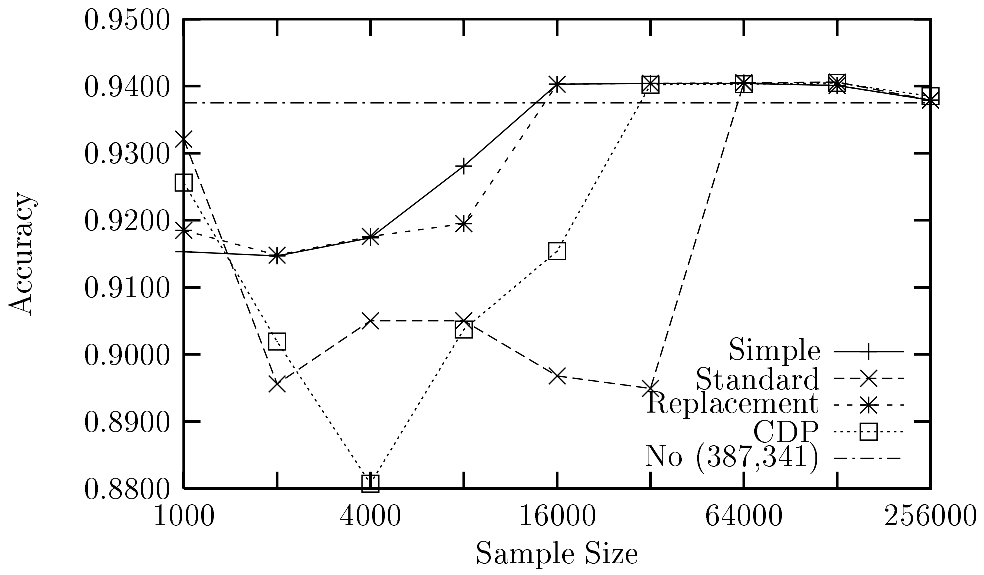

Figure 5.4: Accuracy of sub-sampling variants on the Cover Type data set

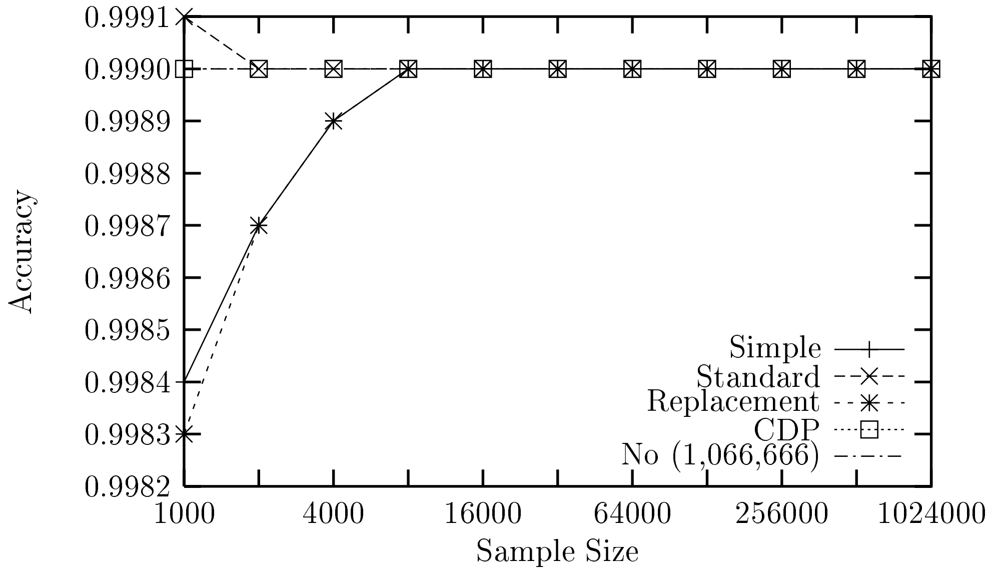

Figure 5.5: Accuracy of sub-sampling variants on the Distinct Boundary data set

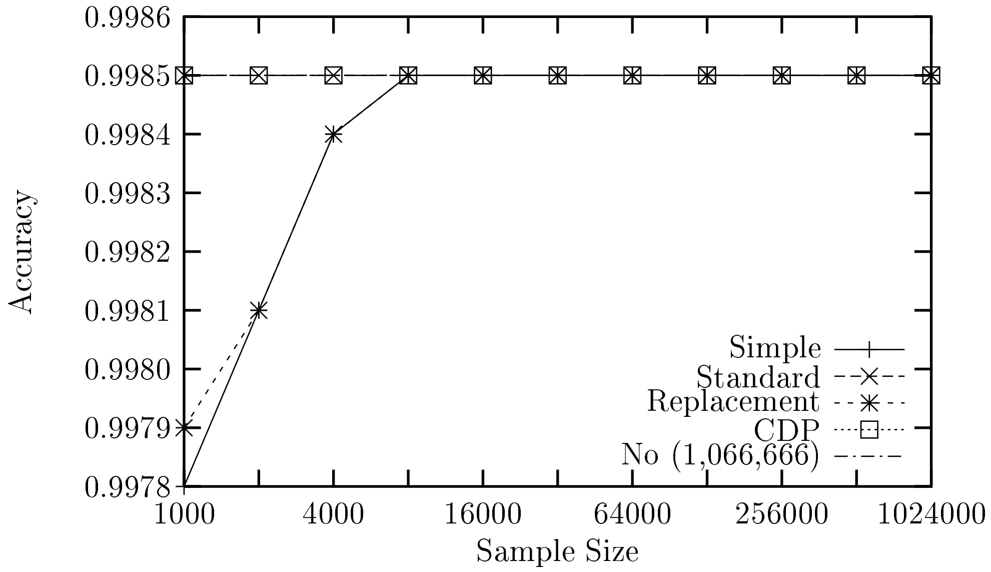

Figure 5.6: Accuracy of sub-sampling variants on the Fuzzy Boundary data set

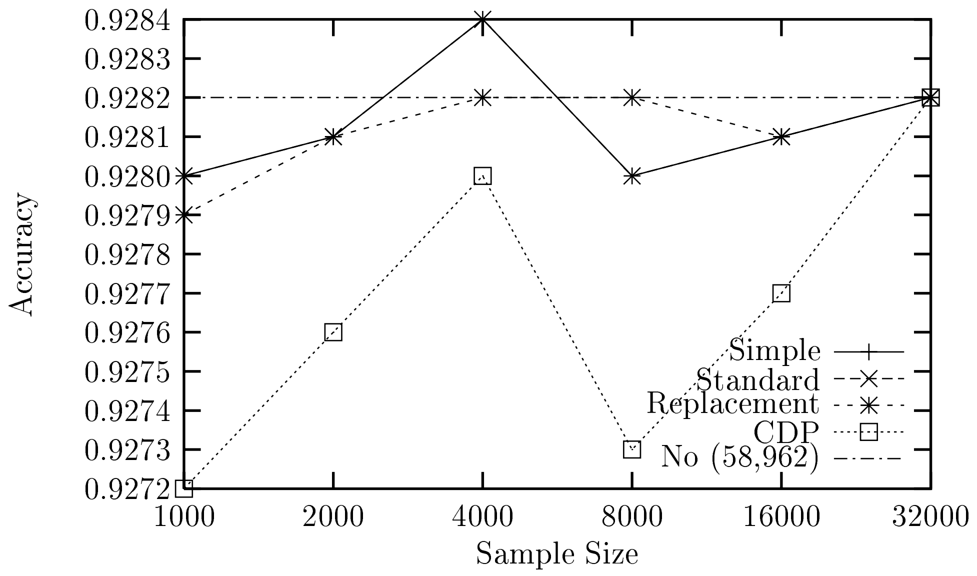

Figure 5.7: Accuracy of sub-sampling variants on the IPUMS data set

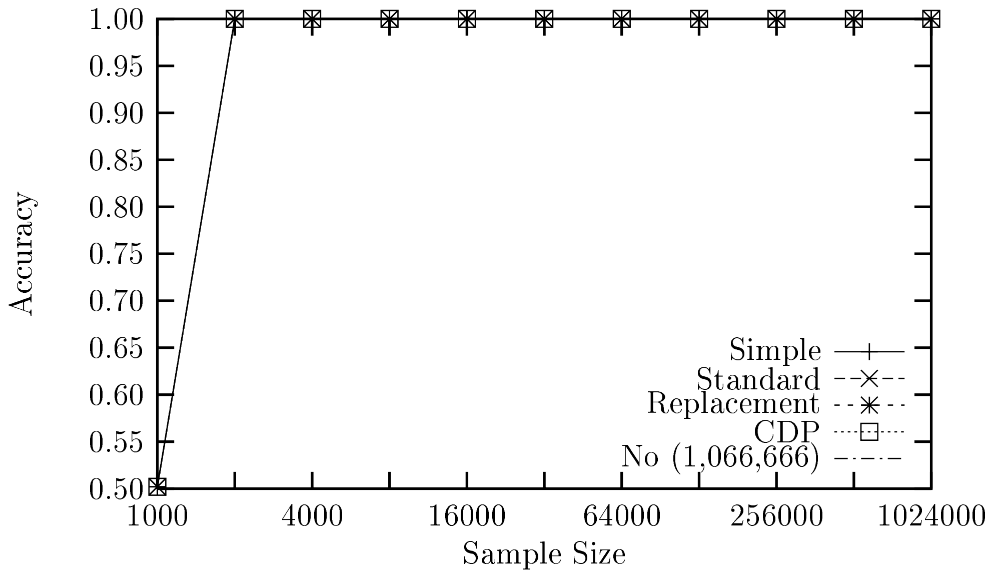

Figure 5.8: Accuracy of sub-sampling variants on the Random Binary data set

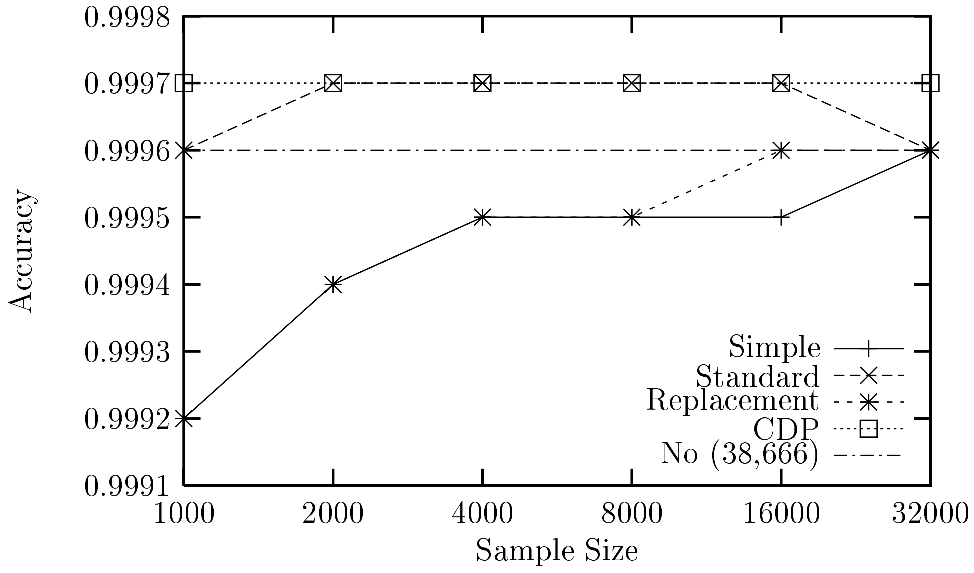

Figure 5.9: Accuracy of sub-sampling variants on the Shuttle data set

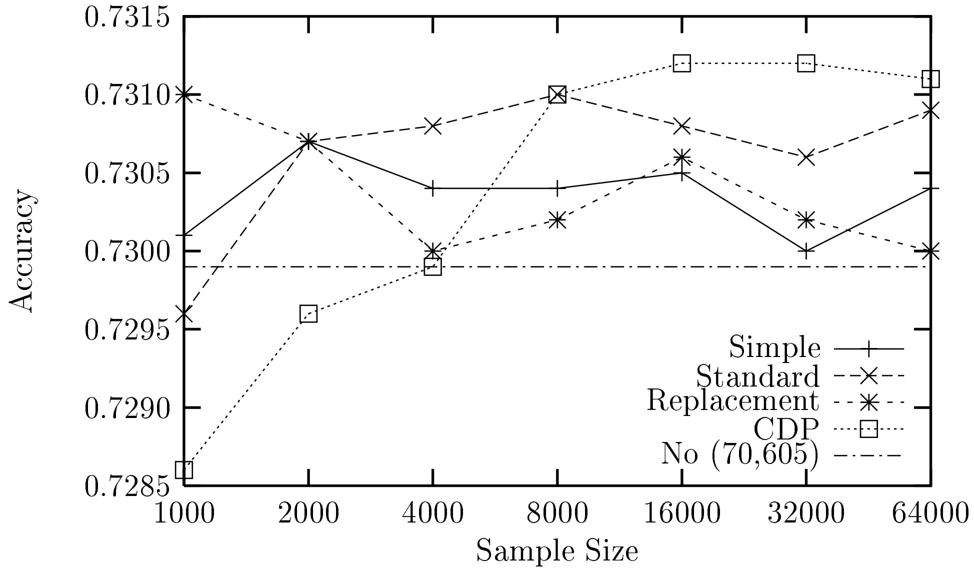

Figure 5.10: Accuracy of sub-sampling variants on the Sleep data set

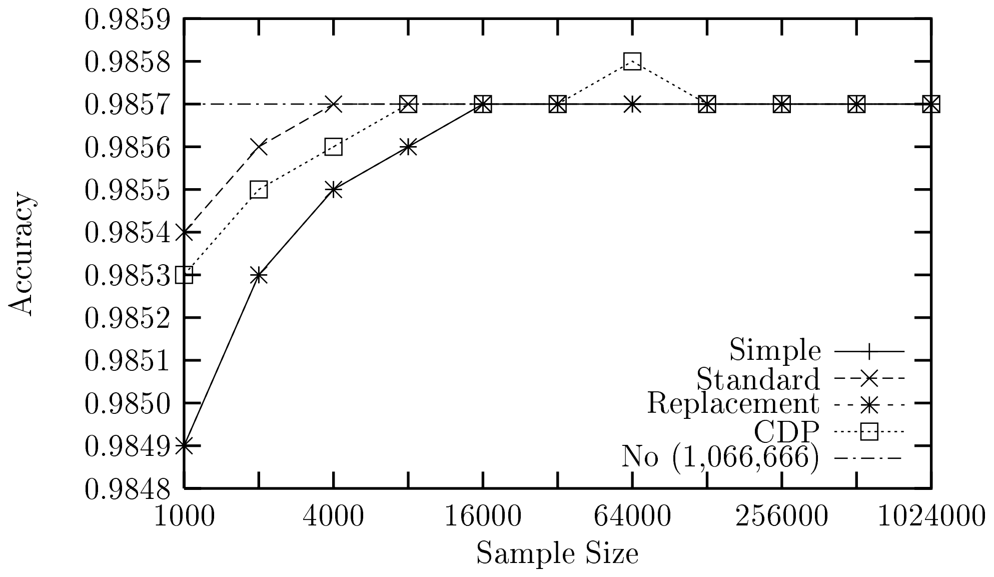

Figure 5.11: Accuracy of sub-sampling variants on the Waveform data set

The figures show that replacement and disproportionate sampling often have little effect on the accuracy of sub-sampling. Of the eleven data sets, the magnitude of the effects of the sub-sampling variants appears to be large on only four — Cover Type, IPUMS, Shuttle, and Sleep.

A summary of the effect sub-sampling variants have on accuracy is given in Table 5.1. Results are presented as win:loss records, with wins attributed to the variant specified in the row. The table shows that standard sub-sampling is generally more accurate than other variants. It is interesting that standard sub-sampling is less accurate than non-replacement sub-sampling only two more times than it is simple sub-sampling. This suggests that disproportionate sampling is more beneficial to accuracy than non-replacement, and is supported by the close win:loss record between standard and DP sub-sampling.

It is interesting that on data sets where the accuracy of standard sub-sampling consistently equals that of no sampling, simple sub-sampling will also equal the accuracy of no sampling, but with slightly larger sample size.

Table 5.1: Effect of sub-sampling variants on accuracy

	Standard	Non-Replacement	DP	Simple
Standard	—	34:10	23:18	34:8
Simple	8:34	10:18	19:35	—

The results for the Cover Type data set are surprising. Here, simple sub-sampling is substantially more accurate than standard sub-sampling for many sample sizes. Simple sub-sampling also reaches accuracy greater than no sampling with smaller sample size than standard sub-sampling.

Changing the way in which samples are selected made no difference to accuracy for the Random Binary set. This may be due to the apparent "all-or-nothing" behaviour of sub-sampling methodologies on this data set. The results suggest that with a sample of 1,000 instances, both sub-sampling versions cannot justify inference of any more than a decision stump. However, samples of 2,000 or more instances are sufficient to infer a model with exceptional predictive accuracy.

Table 5.2 shows the sample size required for standard sub-sampling and simple sub-sampling to reach bounded accuracy loss from no sampling of one, two, and five per cent. The results show that both versions of sub-sampling almost always reach one, two, or five per cent bounded loss with the same sample size. The only difference is on the Cover Type data set, where simple sub-sampling is within one and two per cent accuracy loss bounds with a smaller sample size than standard sub-sampling.

Model Complexity

Figures 5.12–5.22 plot, for each data set, the mean number of nodes inferred for no sampling and standard, simple, non-replacement, and DP sub-sampling.

The figures show that the type of sub-sampling often has a substantial effect

Table 5.2: Comparison of sample size required to achieved given bounded accuracy loss between standard and simple sub-sampling

	1 per cent loss		2 per cent loss		5 per cent loss	
Data Set	Std	Simple	Std	Simple	Std	Simple
Adult	2,000	2,000	1,000	1,000	1,000	1,000
Census Income	1,000	1,000	1,000	1,000	1,000	1,000
Connect-4	8,000	8,000	4,000	4,000	1,000	1,000
Cover Type	64,000	16,000	64,000	8,000	1,000	1,000
Distinct Boundary	1,000	1,000	1,000	1,000	1,000	1,000
Fuzzy Boundary	1,000	1,000	1,000	1,000	1,000	1,000
IPUMS	1,000	1,000	1,000	1,000	1,000	1,000
Random Binary	2,000	2,000	2,000	2,000	2,000	2,000
Shuttle	1,000	1,000	1,000	1,000	1,000	1,000
Sleep	1,000	1,000	1,000	1,000	1,000	1,000
Waveform	1,000	1,000	1,000	1,000	1,000	1,000

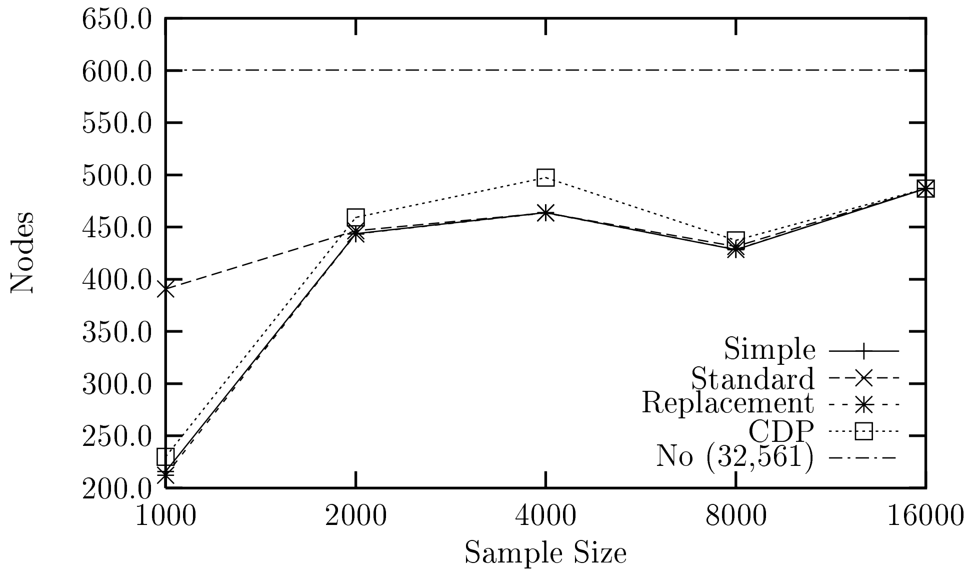

Figure 5.12: Mean nodes inferred by sub-sampling variants on the Adult data set

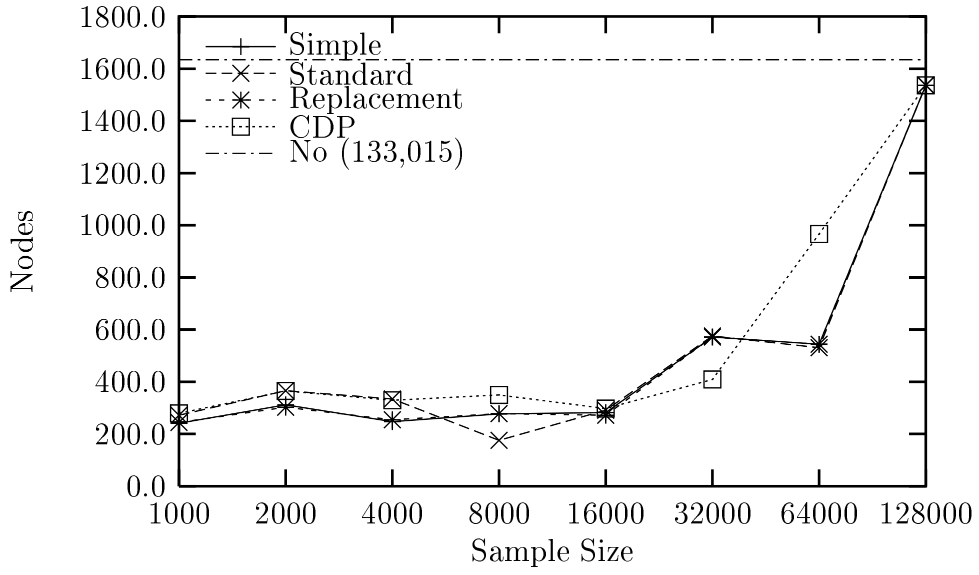

Figure 5.13: Mean nodes inferred by sub-sampling variants on the Census Income data set

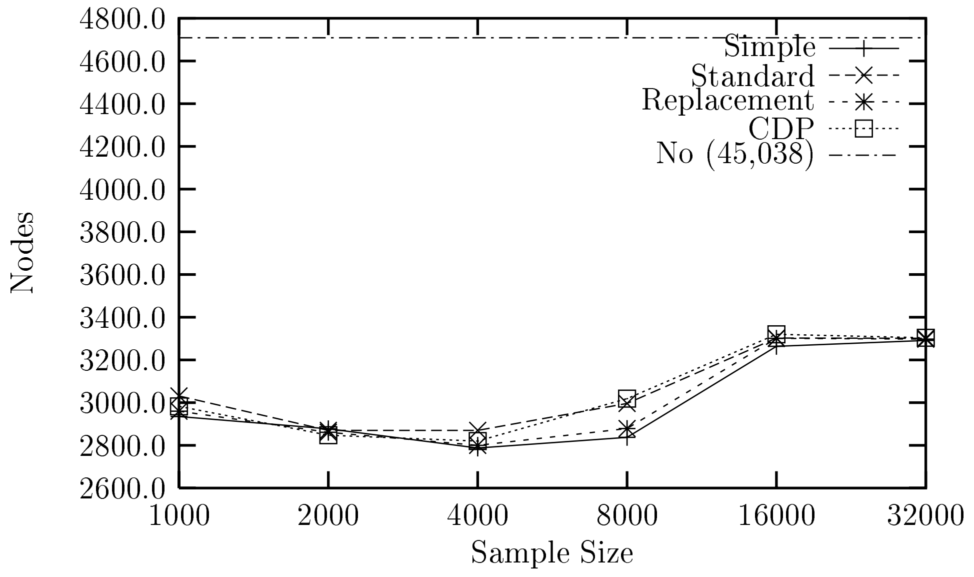

Figure 5.14: Mean nodes inferred by sub-sampling variants on the Connect-4 data set

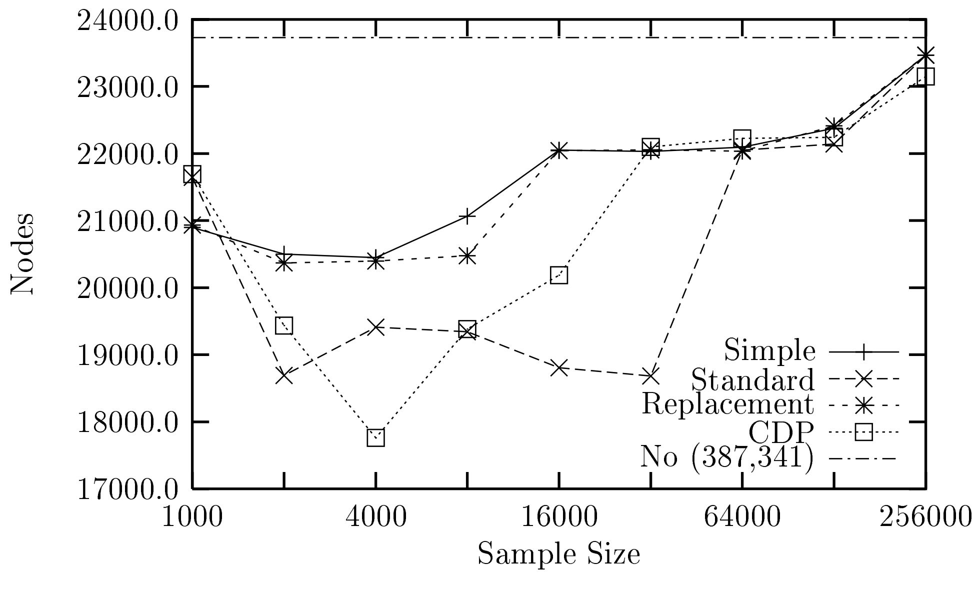

Figure 5.15: Mean nodes inferred by sub-sampling variants on the Cover Type data set

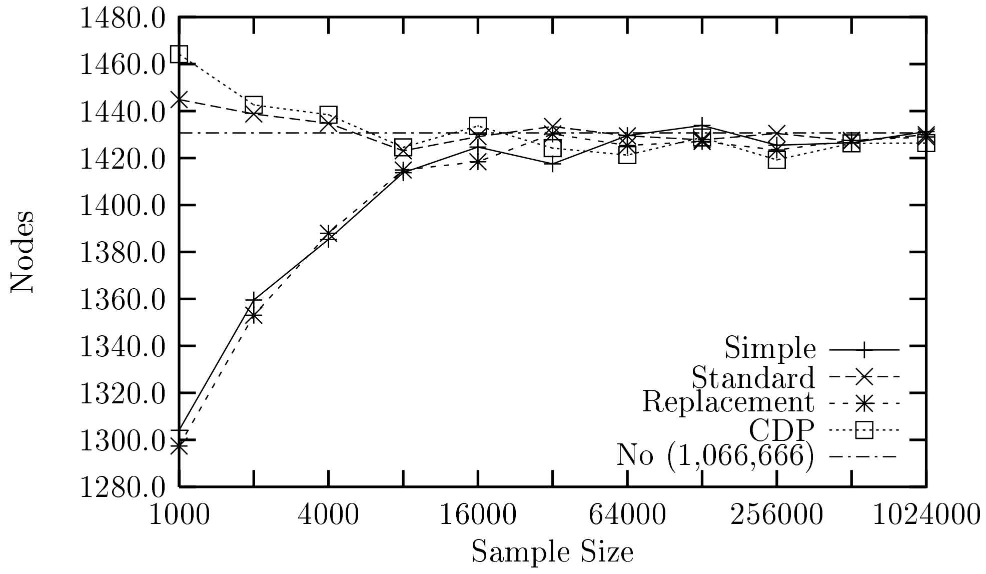

Figure 5.16: Mean nodes inferred by sub-sampling variants on the Distinct Boundary data set

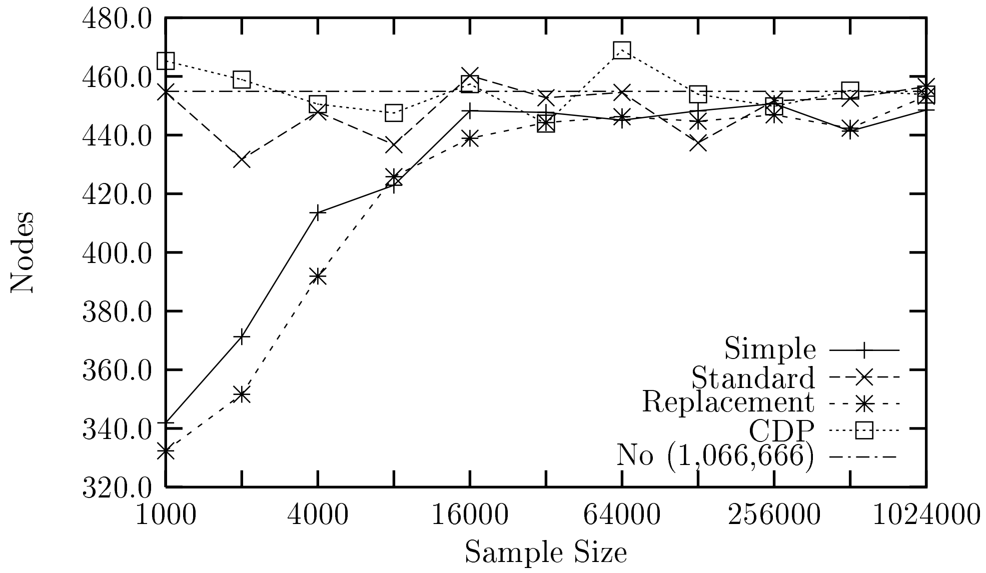

Figure 5.17: Mean nodes inferred by sub-sampling variants on the Fuzzy Boundary data set

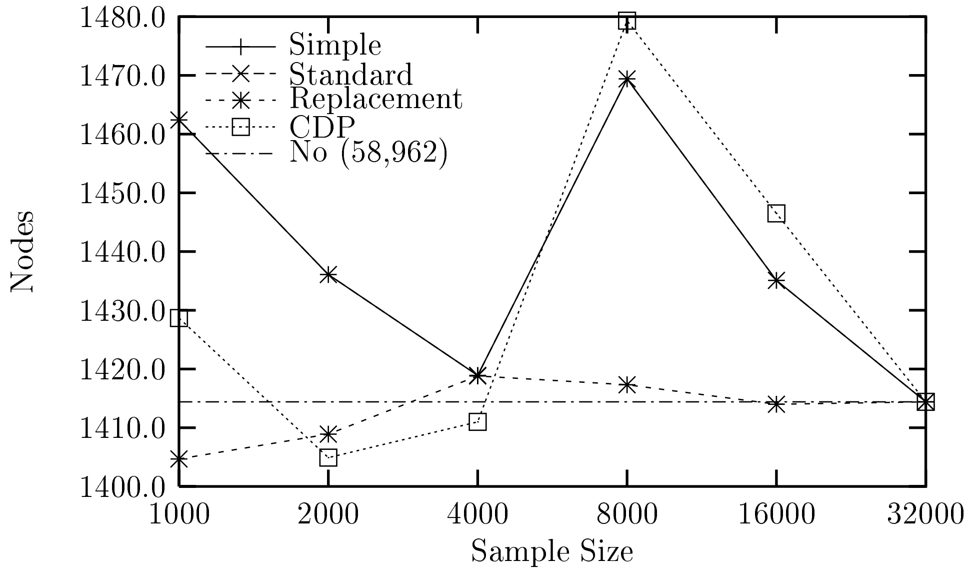

Figure 5.18: Mean nodes inferred by sub-sampling variants on the IPUMS data set

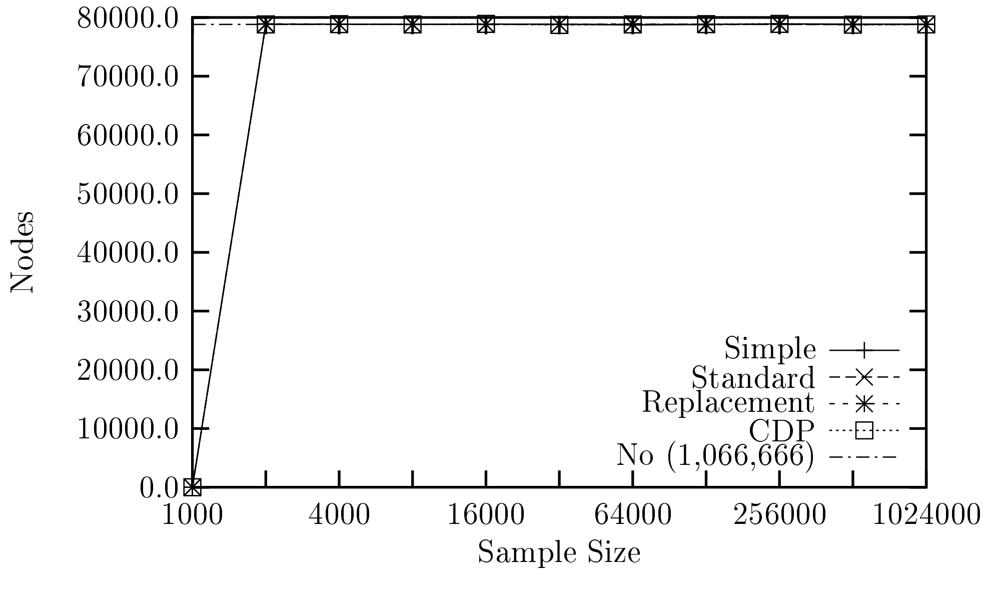

Figure 5.19: Mean nodes inferred by sub-sampling variants on the Random Binary data set

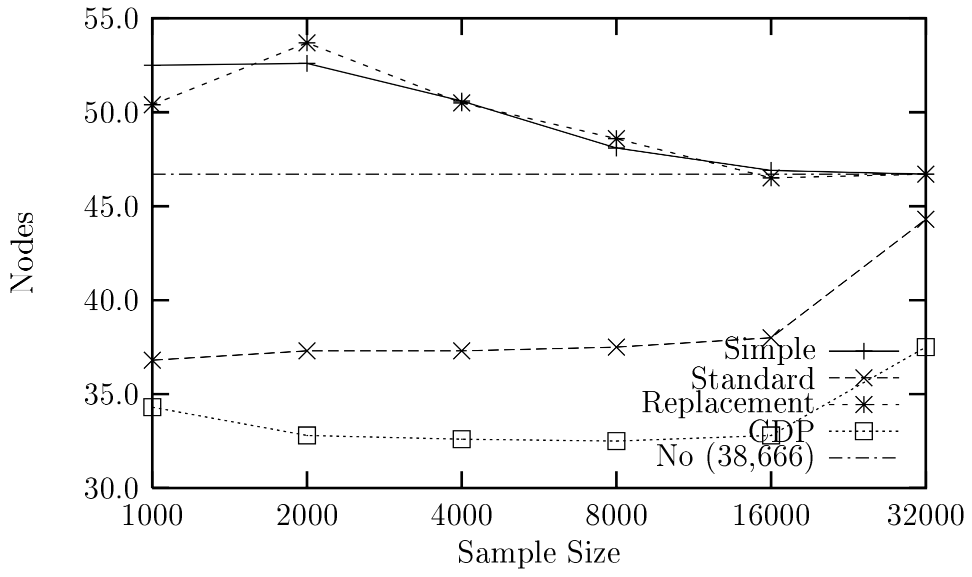

Figure 5.20: Mean nodes inferred by sub-sampling variants on the Shuttle data set

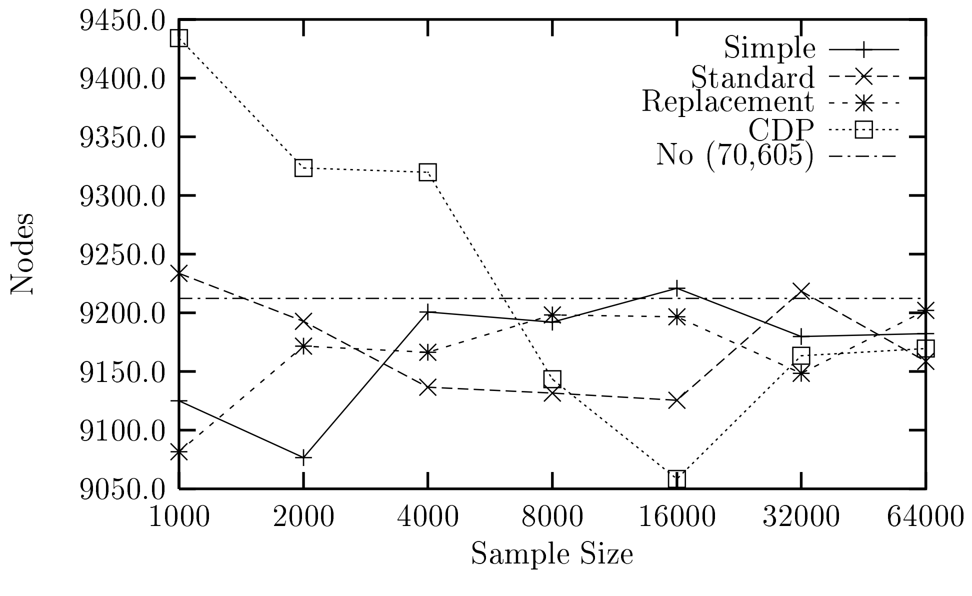

Figure 5.21: Mean nodes inferred by sub-sampling variants on the Sleep data set

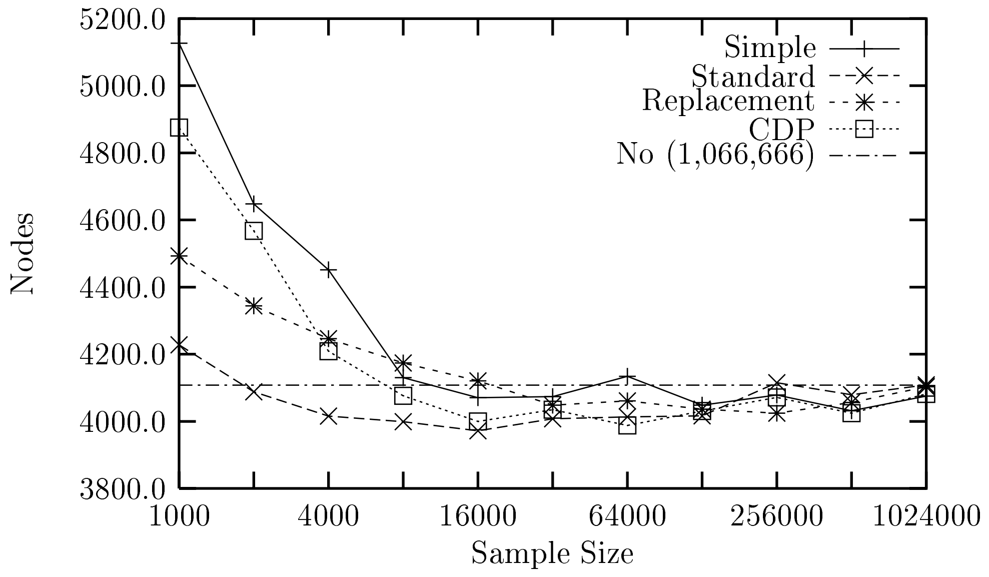

Figure 5.22: Mean nodes inferred by sub-sampling variants on the Waveform data set

on the number of nodes inferred. It is interesting that on the Shuttle data set, both simple and standard sub-sampling infer more nodes than no sampling, but non-replacement and DP sub-sampling infer fewer.

Execution Time

Figures 5.23–5.33 plot, for each data set, execution time for no sampling and standard, simple, non-replacement, and DP sub-sampling. Measured execution time includes the time required to build and prune a model, and small associated overheads. The time to initialise data structures, read data from disk, and evaluate model accuracy is not included in this measurement.

The figures show that the type of sub-sampling affects execution time. However, the results are not unexpected. A summary of the effect sub-sampling variants have on execution time is given in Table 5.3. Results are presented as win:loss records, with wins attributed to the variant specified in the row.

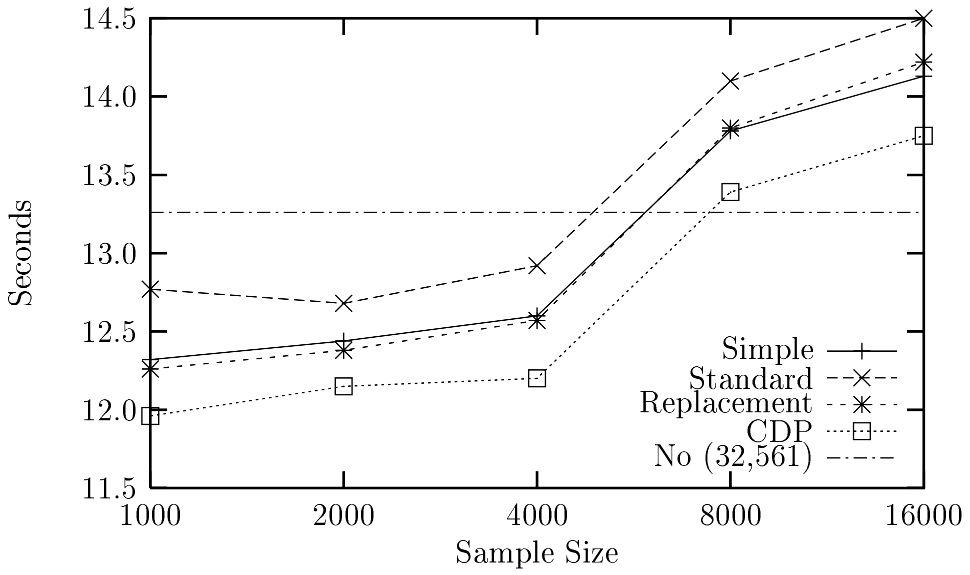

Figure 5.23: Mean model building time for sub-sampling variants on the Adult data set

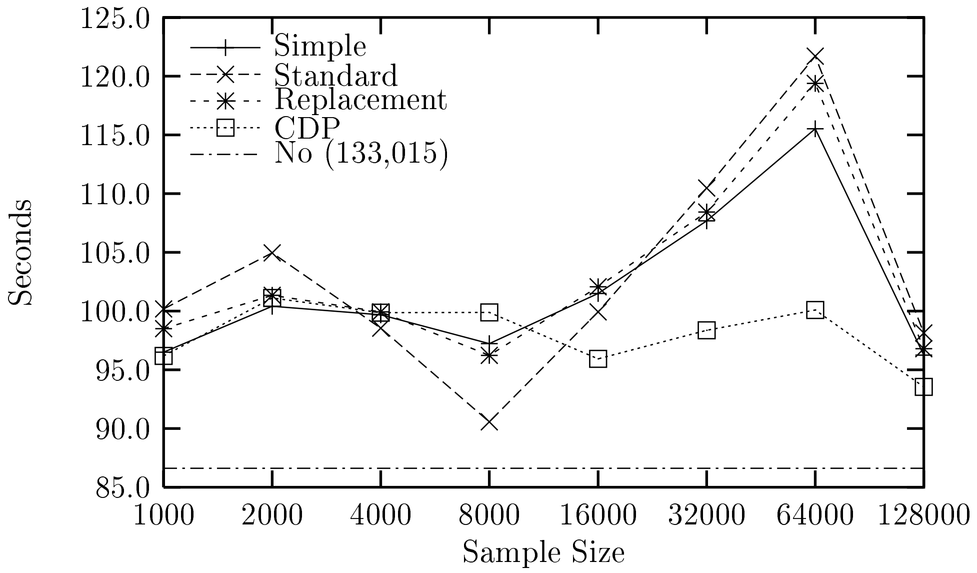

Figure 5.24: Mean model building time for sub-sampling variants on the Census Income data set

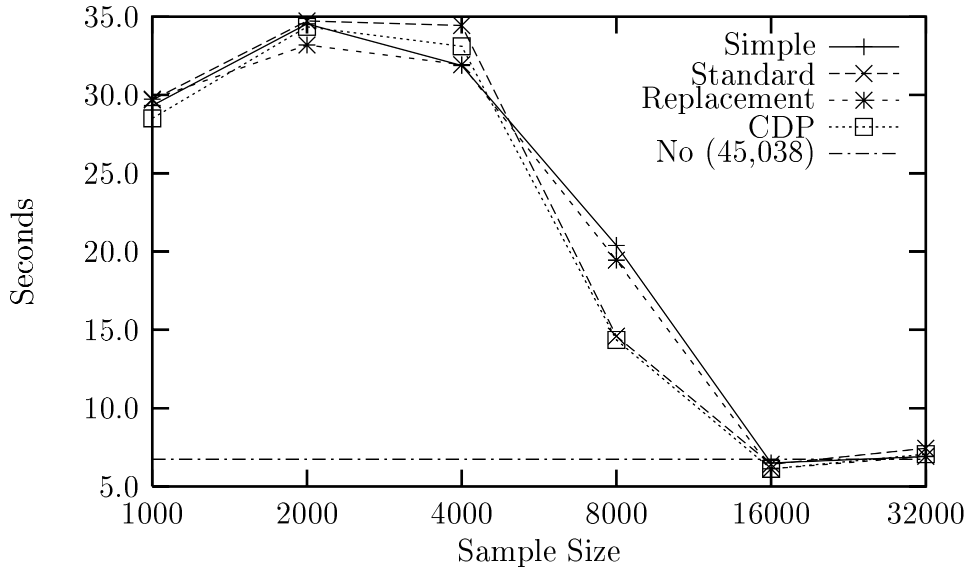

Figure 5.25: Mean model building time for sub-sampling variants on the Connect-4 data set

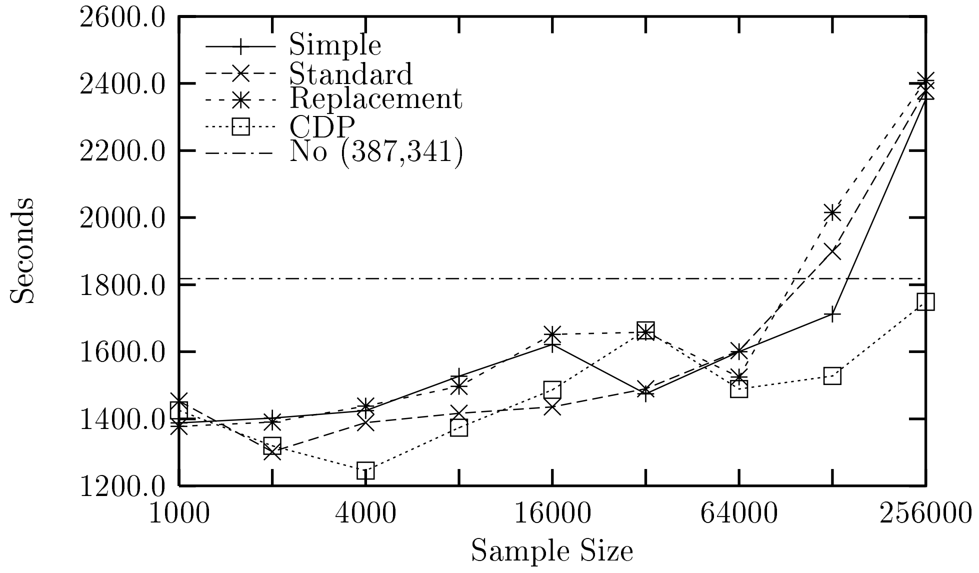

Figure 5.26: Mean model building time for sub-sampling variants on the Cover Type data set

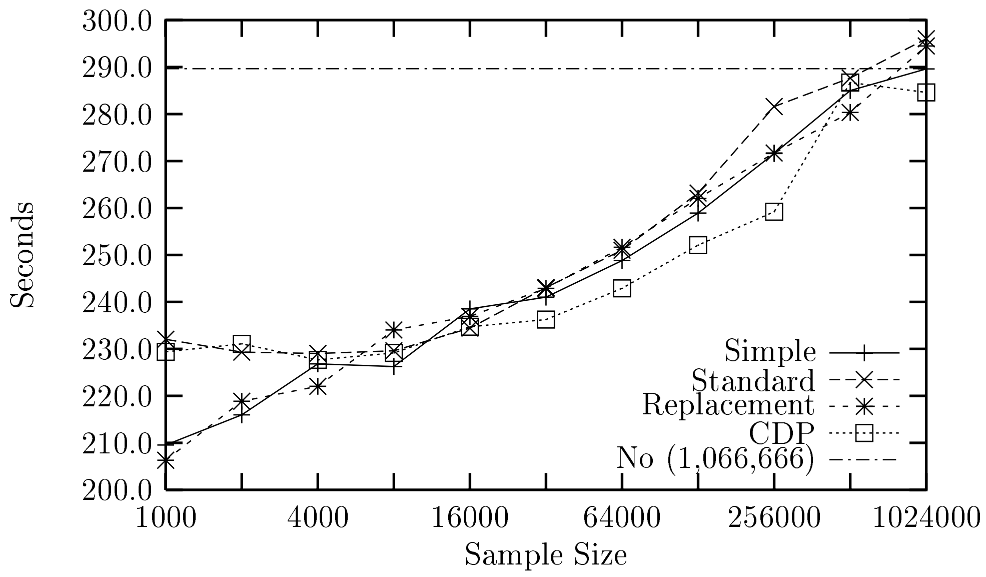

Figure 5.27: Mean model building time for sub-sampling variants on the Distinct Boundary data set

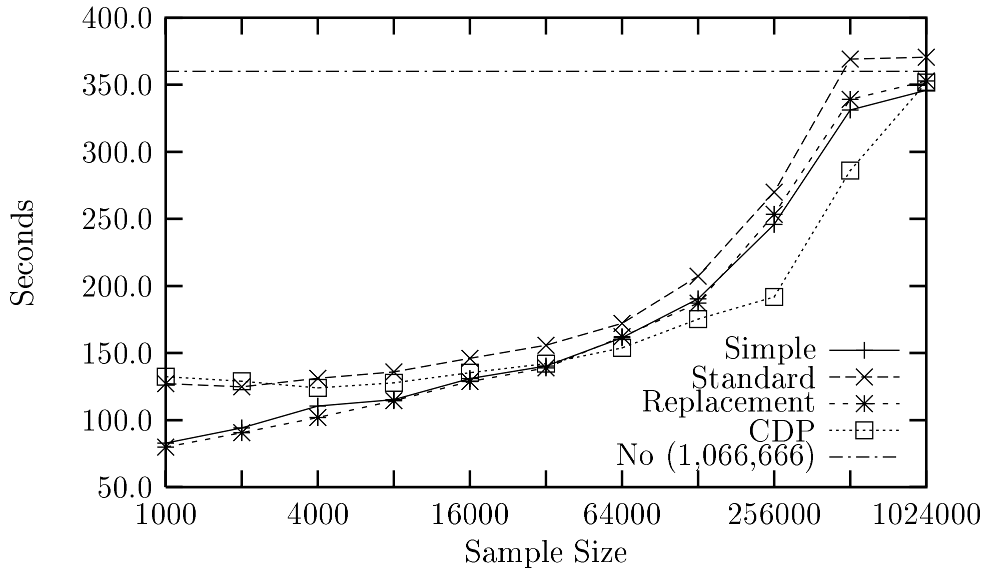

Figure 5.28: Mean model building time for sub-sampling variants on the Fuzzy Boundary data set

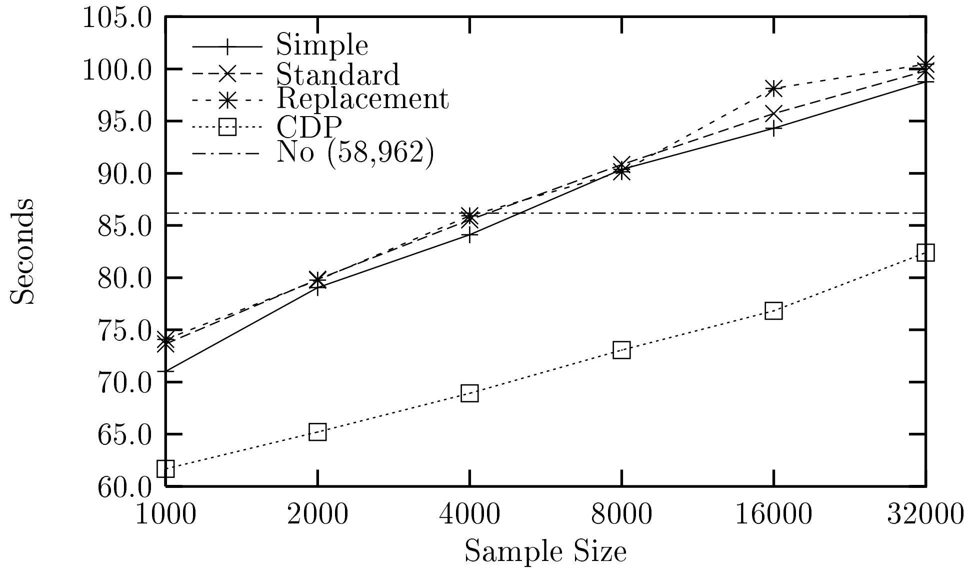

Figure 5.29: Mean model building time for sub-sampling variants on the IPUMS data set

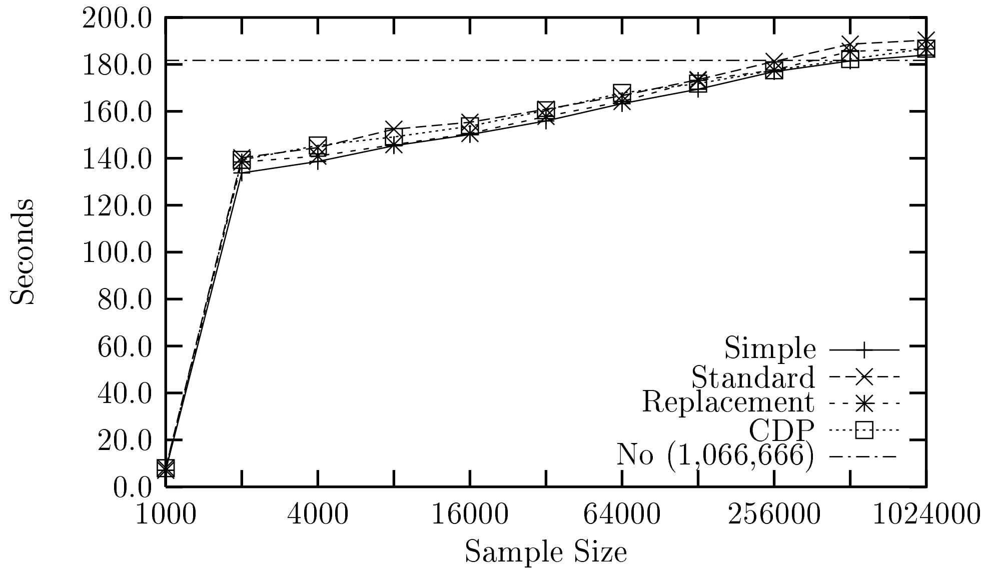

Figure 5.30: Mean model building time for sub-sampling variants on the Random Binary data set

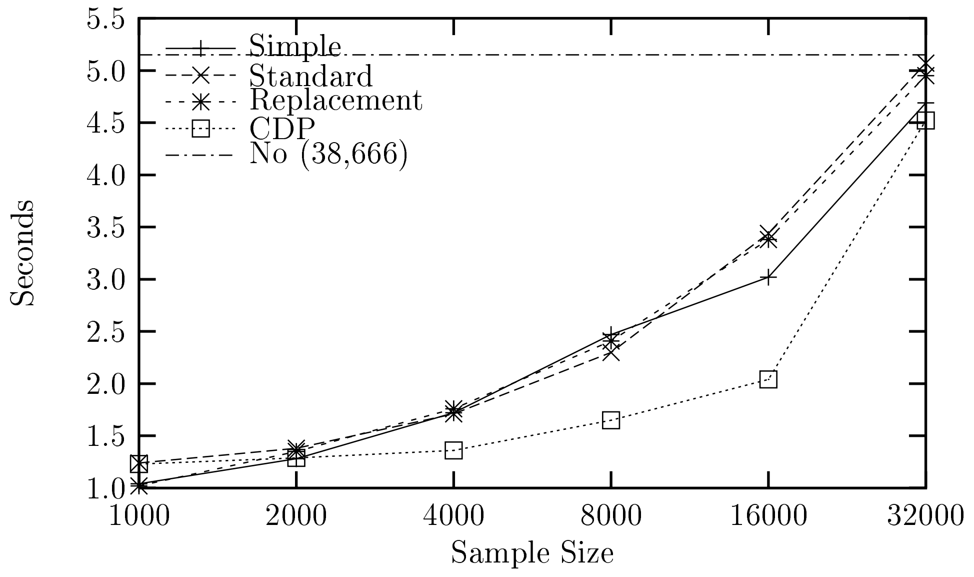

Figure 5.31: Mean model building time for sub-sampling variants on the Shuttle data set

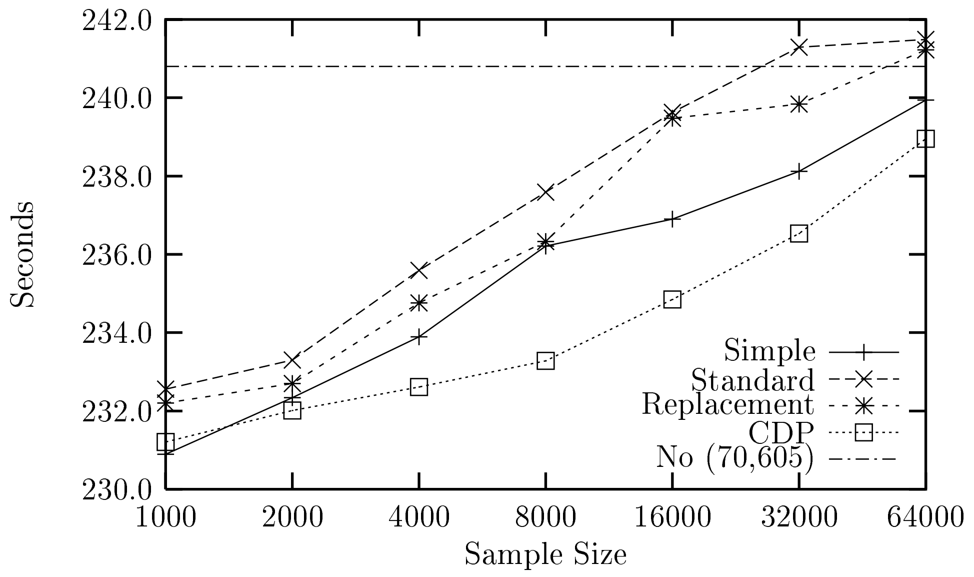

Figure 5.32: Mean model building time for sub-sampling variants on the Sleep data set

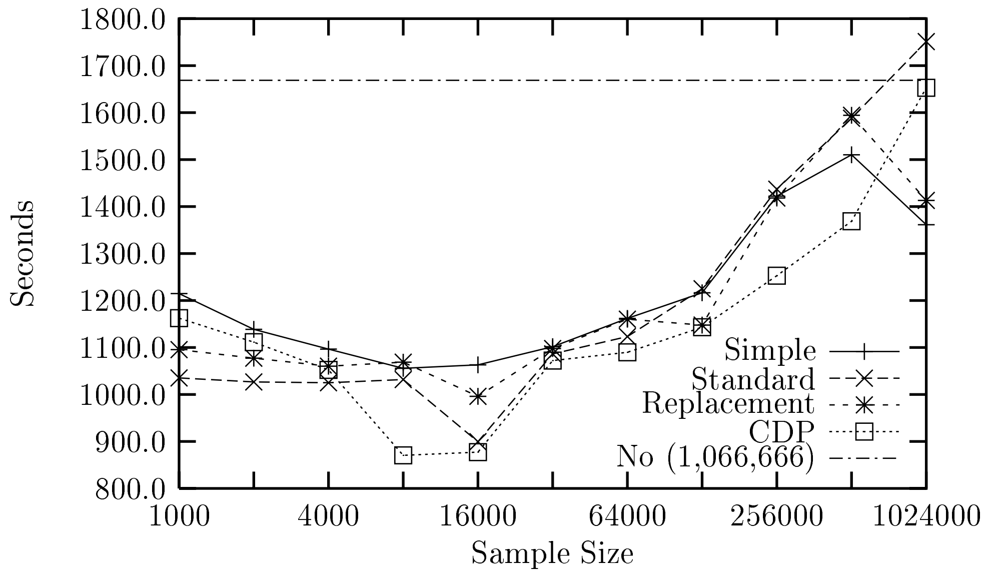

Figure 5.33: Mean model building time for sub-sampling variants on the Waveform data set

Table 5.3: Effect of Sub-sampling Variants on Execution Time

	Standard	Non-Replacement	DP	Simple
Standard	—	29:62	14:77	19:72
Simple	72:19	58:33	34:57	—

The table shows that both non-disproportionate sampling and sampling with replacement can be expected to reduce execution time.

The results show that simple sub-sampling often requires less execution time than standard sub-sampling. Of the eleven data sets, five show simple sub-sampling to be quicker than standard sub-sampling at every sample size. Comparing each sample size shows simple sub-sampling is quicker than standard sub-sampling on 72 occasions and slower on 19. Occasions where simple sub-sampling takes longer than standard sub-sampling generally equate to a small increase in execution time.

Since the problem at hand is execution time, it is important to make note of those occasions where standard sub-sampling is slower than no sampling. Restricting the view to these occasions shows that of the 28 occurrences, simple sub-sampling reduces the execution time compared to standard sub-sampling 24 times, and increases it only four times. On eight of these occasions, execution time is reduced to less than that of no sampling.

At the smallest sample size, simple sub-sampling is faster than standard sub-sampling on all but the Waveform data set. At the largest sample size, simple sub-sampling is faster than standard sub-sampling on each data set.

It is interesting that simple sub-sampling generally takes longer to execute than DP sub-sampling. This was investigated, and found to be due to two reasons. When the execution time of simple sub-sampling is a substantial proportion larger than that of DP sub-sampling, it is because DP sub-sampling infers many fewer nodes. When the execution time of the sub-sampling variants is close, the number of nodes in the unpruned models are similar. However, DP sub-sampling tends to have more leaves around the middle of the tree than simple sub-sampling. This may result in greater execution time higher in the tree, and less lower in the tree.

Table 5.4 shows the execution time required to achieve accuracy within a given bounded loss for simple and standard sub-sampling. Many entries are consistent across different bounds because, as shown in Table 5.2, all bounds were often achieved with the same sample size.

The table shows simple sub-sampling is quicker than standard sub-sampling with one per cent accuracy loss on eight data sets, and slower on three. With both two and five per cent accuracy loss, simple sub-sampling is quicker on ten data sets, and slower on only the Waveform data set. Comparing the execution time taken to build the smallest model after which accuracy loss was consistently no more than one per cent for both types of sampling, shows simple sub-sampling

Table 5.4: Execution time required to achieve bounded accuracy loss

	1 per cent loss		2 per cent loss		5 per cent loss	
Data Set	Std	Simple	Std	Simple	Std	Simple
Adult	12.68	12.44	12.77	12.32	12.77	12.32
Census Income	100.18	96.51	100.18	96.51	100.18	96.51
Connect-4	14.59	20.39	34.43	31.88	29.68	29.32
Cover Type	1603.14	1621.56	1603.14	1526.72	1603.14	1387.83
Distinct Boundary	232.02	209.53	232.02	209.53	232.02	209.53
Fuzzy Boundary	127.07	82.80	127.07	82.80	127.07	82.80
IPUMS	73.65	71.03	73.65	71.03	73.65	71.03
Random Binary	140.20	133.77	140.20	133.77	140.20	133.77
Shuttle	1.24	1.04	1.24	1.04	1.24	1.04
Sleep	232.56	230.89	232.56	230.89	232.56	230.89
Waveform	1035.46	1214.73	1035.46	1214.73	1035.46	1214.73

takes, on average, 1.5% less time than standard sub-sampling. With two per cent bounded accuracy loss, simple sub-sampling takes 6.5% less time, and with five per cent, 5.9% less time.

5.2.4 Summary

The modifications made to the way in which samples are selected with sub-sampling often have little impact on accuracy. Occasionally, simple sub-sampling produced models with greater accuracy, but more often than not resulted in a decrease in accuracy. However, the purpose of these experiments was to investigate the effects of replacement and disproportionate sampling on sub-sampling. The results suggest that non-disproportionate sampling can be expected to have a greater detriment on accuracy than sampling with replacement. Managing replacement also appears to require more execution time than disproportionate sampling. This should not be taken as suggesting that disproportionate sampling and sampling without replacement are not valuable tools, but rather that the need for them should be carefully considered.

These results also show that the results of Chapter 4 are not substantially affected by the method used to select samples. Although it is reasonable to expect that the method of selecting a sample had some influence on the results of the previous Chapter, the results presented here show the magnitude of the effects are insufficient to substantially alter the results of Chapter 4.

5.3 Sampling Using Variable Proportions

One of the issues with standard sub-sampling is that the same sample size is used at each node. Although this is a straight-forward methodology, and, as evidenced in Section 4.5.1, can produce accurate models, the implications of using a static sample size may be undesirable. For example, given two nodes of the same depth, it may be unreasonable to use the same sample size if the number of instances in the available data of these nodes differs markedly. Rather than selecting a number of instances, it may be more sensible to select a proportion of instances.

However, selecting a fixed proportion of instances is just as undesirable as selecting a fixed number of instances. This is because, as depth increases, the number of instances in the available data decreases. Therefore, using the same proportion of instances low in the tree as used high in the tree could result in unreasonably small sample sizes. For example, consider two nodes: one high in the tree with one million instances in the available data; the other lower in the tree with 1,000 instances in the available data. Selecting one per cent of the data at the higher node would result in 10,000 instances being used to determine the split. This may be a large enough sample to allow a reasonable split to be found.

However, at the lower node, selecting one per cent of instances would result in the split being determined using only 10 instances. Such a small sample is much more likely to be insufficient to determine an appropriate split.

Also, it can be expected that the effect of selecting a poor split at a lower node may be more detrimental to the accuracy of the final model than selecting a poor split at a higher node. If a poor split is selected high in the tree, it is likely that each branch of that node will contain substantial amounts of data. This allows an opportunity for the poor split to be corrected by lower nodes. However, such an opportunity is unlikely to be available at lower nodes, where there is less data.

It is therefore reasonable that, as depth increases, a greater proportion of the available data should be used so as to reduce the possibility of a poor split being selected. Such a methodology will hereafter be referred to as variable proportion sampling.

In effect, a similar phenomenon occurs with standard sub-sampling, since a fixed sample size is used and the amount of available data necessarily decreases with increasing depth. However, with standard sub-sampling, the practitioner only has control over the proportion of instances used at the root node.

It is also unreasonable to use a proportion of the available data when the amount of data is small. Hence, a minimum size, below which sampling is not performed, should be enforced.

The use of an increasing proportion of available data for inference may not only be more intuitive than using a static size, but also allows practitioners greater control over the workings of sub-sampling. However, this additional control also means the practitioner must specify additional parameters.

Calculating Sample Size

Given that an increasing proportion of instances will be used, the problem remains of how to determine sample size. Two methods for calculating sample size s are proposed:

Linear:

$$s = np(d + 1)$$

Geometric:

$$s = np2^{d+1}$$

where n is the number of instances in the available data, d is the depth of the node with the root node being depth zero, and p is the proportion of instances to use at the root node. If s is less than a practitioner specified minimum sample size, all available data will be used.

5.3.1 Experiments

Experiments were performed to compare proportional sampling to both standard sub-sampling and no sampling. The proportion of instances to use at the root node will hereafter be referred to as the *initial proportion*. The initial proportions investigated were one, two, five, and ten per cent, using both linear and geometric calculations of sample size. A minimum sample size of 100 instances was used in all experiments.

The experiments were performed using ten-times three-fold cross-validation. All data sets used in previous experiments were used in these experiments.

5.3.2 Results

Tables compare the accuracy and execution time of variable proportion sampling to that of no sampling. Since there is no direct mapping of sample sizes used in previous sub-sampling experiments to these experiments, it is inappropriate

Table 5.5: Accuracy of variable proportion sampling using a Linear progression with varying initial proportions

Data Set	No	Standard	1%	2%	5%	10%
Adult	0.8602	0.8589	0.8484	0.8462	0.8444	0.8439
Census Income	0.9526	0.9526	0.9456	0.9447	0.9482	0.9517
Connect-4	0.7981	0.7999	0.7824	0.7842	0.7860	0.7962
Cover Type	0.9375	0.9406	0.9369	0.9390	0.9379	0.9375
Distinct Boundary	0.9990	0.9991	0.9987	0.9989	0.9990	0.9990
Fuzzy Boundary	0.9985	0.9985	0.9985	0.9985	0.9985	0.9985
IPUMS	0.9282	0.9284	0.9283	0.9287	0.9286	0.9281
Random Binary	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
Shuttle	0.9996	0.9997	0.9991	0.9991	0.9994	0.9995
Sleep	0.7299	0.7310	0.7289	0.7288	0.7293	0.7300
Waveform	0.9857	0.9857	0.9850	0.9849	0.9856	0.9857

to compare variable proportion sampling to all sample sizes used with standard sub-sampling. Hence, only the sample size with the greatest predictive accuracy in Section 4.5.1 is used for comparison. In the event that the greatest accuracy is achieved with more than one sample size, the smallest sample size with that accuracy is used for comparisons of execution time.

Accuracy

Tables 5.5 and 5.6 show the accuracy of variable proportion sampling, using a linear and geometric progression respectively. The results are compared to the accuracy of no sampling, and the highest accuracy achieved using standard sub-sampling. The latter two results are taken from Chapter 4.

The results show that variable proportion sampling with a linear progression is generally less accurate than no sampling. Linear variable proportion sampling

Table 5.6: Accuracy of variable proportion sampling using a Geometric progression with varying initial proportions

Data Set	No	Standard	1%	2%	5%	10%
Adult	0.8602	0.8589	0.8502	0.8557	0.8585	0.8588
Census Income	0.9526	0.9526	0.9515	0.9519	0.9515	0.9519
Connect-4	0.7981	0.7999	0.7964	0.7965	0.7966	0.7979
Cover Type	0.9375	0.9406	0.9376	0.9376	0.9377	0.9375
Distinct Boundary	0.9990	0.9990	0.9990	0.9990	0.9990	0.9990
Fuzzy Boundary	0.9985	0.9985	0.9985	0.9985	0.9985	0.9985
IPUMS	0.9282	0.9284	0.9278	0.9275	0.9283	0.9282
Random Binary	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
Shuttle	0.9996	0.9997	0.9993	0.9993	0.9995	0.9995
Sleep	0.7299	0.7310	0.7300	0.7301	0.7296	0.7302
Waveform	0.9857	0.9857	0.9856	0.9857	0.9857	0.9857

with an initial proportion of one per cent is less accurate than no sampling eight times and more accurate only once, with two ties. With two per cent initial proportion, the linear progression is less accurate seven times and more accurate twice, with two ties. With five per cent, it is less accurate six times and more accurate twice, with three ties. With ten per cent, it is less accurate five times and more accurate once, with five ties. Comparing variable proportion sampling with a linear progression to the most accurate result with standard sub-sampling is even less favourable. An initial proportion of one per cent is less accurate nine times and never more accurate, with two ties. Starting proportions of two and five per cent both result in variable proportion sampling being less accurate eight times and more accurate once, with two ties. An initial proportion of ten per cent is less accurate eight times, with three ties.

The results when using a geometric progression show a similar trend. Compared to no sampling, an initial proportion of one per cent is more accurate twice and less accurate six times, with three ties. Starting proportions of two and five per cent are both more accurate twice and less accurate five times, with four ties. Ten per cent is more accurate once and less accurate four times, with six ties. Compared to the most accurate standard sub-sampling, an initial proportion of one per cent is less accurate eight times and never more accurate, with three ties. Starting proportions of two and five per cent are less accurate seven times and never more accurate, with four ties. Ten per cent is less accurate seven times and never more accurate, with four ties.

The results suggest that the use of variable proportion sampling is likely to cause a loss in accuracy when compared to no sampling. However, the difference in accuracy is relatively small. Comparing levels of accuracy loss from no sampling shows that both progressions of variable proportion sampling perform well. With a linear progression, initial proportions of one, two, and five per cent perform within one per cent bounded accuracy loss on nine occasions, and outside this twice. An initial proportion of ten per cent is within this bound ten times, and outside it once. When looking at two per cent bounded accuracy loss, all initial proportions are within this bound on all data sets. The trend is stronger when a geometric progression is used. There is only one occasion with all initial proportions where accuracy is not within one per cent of that obtained with no sampling. This occurs with one per cent initial proportion on the Adult data set.

Comparing initial proportions of linear and geometric progressions to each other shows that a geometric progression generally produces more accurate models. With an initial proportion of one per cent, the geometric progression has greater accuracy on nine data sets, and never has less accuracy, with two ties. With an initial proportion of two per cent, the geometric progression has greater accuracy seven times, with two ties and two losses. With an initial proportion of five per cent, geometric has six wins, three ties, and two losses. With an initial proportion of ten per cent, it has five wins and six ties, with no losses.

Thus, it may be said that a geometric progression produces models with greater predictive accuracy than a linear progression. Both generally produce models with small accuracy loss, although a geometric progression is slightly more consistent in this measure.

Comparing the accuracy obtained using differing initial proportions with a linear progression shows that larger initial proportions generally result in more accurate models. Compared to one per cent, five per cent is more accurate eight times and less accurate once, with two ties. Compared to two per cent, five per cent is more accurate six times and less accurate three times, again with two ties. Compared to one per cent, ten per cent is more accurate seven times and less accurate twice, with two ties. Compared to two per cent, ten percent is more accurate six times and less accurate three times, with two ties. Compared to five per cent, it is more accurate five times and less accurate three times, with three ties.

The results show a similar pattern with a geometric progression. Here, an initial proportion of five per cent is more accurate than one per cent six times and less accurate once, with four ties. Compared to two per cent, five per cent is more accurate five times and less accurate twice, with four ties. Compared to one per cent, ten per cent is more accurate seven times and less accurate once, with three ties. Compared to two per cent, it is more accurate five times and less accurate once, with five ties. Compared to five per cent, it is more accurate four times and less accurate twice, with five ties.

The trend for larger initial proportions to build models with greater predictive accuracy might at first be taken as evidence that selecting the correct split high in the tree is more important than suggested in previous discussions. However, this is not necessarily the case. With a linear progression, an initial proportion of one per cent will not result in all the available data being used until nodes of depth 99 are built, assuming there is still a greater number of available instances than the minimum sample size. With two per cent initial sample size, this occurs at depth 49. Producing decision trees of these sizes requires either exceptionally complex concepts in the data, high noise, or repeated selection of poor splits. Therefore, it is likely that the condition forcing use of all available data will be the imposition of a minimum sample size, rather than dictation by a linear progression. However, this phenomenon is likely to be less strong when using an initial proportion of five per cent, as all available data will be used from depth 19 onward. With a initial proportion of ten per cent, it occurs at depth nine.

This argument may also explain why the trend for five per cent initial proportion to be more accurate is not as strong when using a geometric progression. A geometric progression results in all available data being used at shallower depths than a linear progression: depth 7 for one per cent initial proportion; depth 6

Table 5.7: Mean nodes inferred by variable proportion sampling using a Linear progression

Data Set	No	Standard	1%	2%	5%	10%
Adult	600.4	486.8	556.2	207.3	215.6	184.3
Census	1634.5	1536.8	210.3	171.3	168.8	675.5
Connect-4	4708.2	3302.8	2796.1	2815.6	2856.4	3151.7
Cover Type	23730.4	22141.2	23049.9	22633.9	23601.8	23862.6
Distinct	1430.7	1444.9	1299.9	1372.8	1438.8	1461.7
Fuzzy	454.9	454.7	424.3	456.1	483.3	473.8
IPUMS	1414.4	1418.9	1310.3	1329.7	1331.9	1416.3
Random	78838.3	78801.3	79076.5	79055.4	78952.3	78850.4
Shuttle	46.7	37.3	48.1	52.7	55.6	51.3
Sleep	9212.3	9131.7	9167.2	9186.6	9133.9	9203.7
Waveform	4107.5	4015.9	6173.3	7434.9	4340.5	4103.8
VPS fewer than No Sampling			8:3	7:4	5:6	5:6
VPS fewer than Standard Sub-sampling			5:6	5:6	5:6	4:7

for two per cent; depth 5 for five per cent; and depth 4 for ten per cent.

Model Complexity

Tables 5.7 and 5.8 show the mean number of nodes inferred by variable proportion sampling, using a linear and geometric progression respectively. The results are compared to those of no sampling, and the result of standard sub-sampling with the greatest accuracy, taken from Chapter 4.

The results show that variable proportion sampling with a linear progression infers fewer nodes than no sampling 25 times, and more 19 times. The reverse is true when compared to standard sub-sampling. When using a geometric

Table 5.8: Mean nodes inferred by variable proportion sampling using a Geometric progression

Data Set	No	Standard	1%	2%	5%	10%
Adult	600.4	486.8	188.7	257.9	338.5	427.5
Census	1634.5	1536.8	694.9	769.5	646.9	898.0
Connect-4	4708.2	3302.8	3194.2	3242.0	3237.0	3286.5
Cover Type	23730.4	22141.2	23843.4	23804.5	23740.7	23800.5
Distinct	1430.7	1444.9	1439.3	1439.9	1434.5	1430.3
Fuzzy	454.9	454.7	467.3	467.8	454.7	454.5
IPUMS	1414.4	1418.9	1435.9	1465.5	1395.5	1418.0
Random	78838.3	78801.3	78827.5	78872.3	78819.5	78873.2
Shuttle	46.7	37.3	51.9	57.3	53.5	50.1
Sleep	9212.3	9131.7	9235.5	9191.9	9226.1	9143.5
Waveform	4107.5	4015.9	4167.0	4060.4	4028.3	4020.8
VPS fewer then No Sampling			4:7	4:7	7:4	7:4
VPS fewer then Standard Sub-sampling			4:7	4:7	5:1:5	6:5

Table 5.9: Comparison of number of nodes for different initial proportions with a linear progression

	1%	2%	5%	10%
1%	—	7:4	6:5	8:3
2%	4:7	—	6:5	7:4
5%	5:6	5:6	—	6:5
10%	3:8	4:7	5:6	—

progression, variable proportion sampling infers both fewer and more nodes than no sampling 22 times. Compared to standard sub-sampling, geometric variable proportion sampling infers fewer nodes 19 times and more 24, with one tie.

Comparing a linear progression to a geometric progression shows that a linear progression generally infers fewer nodes. With a initial proportion of one per cent, linear infers fewer nodes eight times and more three times; with a initial proportion of two per cent, linear infers fewer nodes nine times and more twice; with five per cent fewer five and more six times; with ten per cent fewer six times and more five.

Tables 5.9 and 5.10 compare the number of nodes inferred with different initial proportions using a linear and geometric progression respectively. The tables show win:loss records, where wins relate to the initial proportion in the row inferring fewer nodes than the proportion in the column. The tables show a weak trend for smaller initial proportions to infer smaller trees with a linear progression, but no such trend for a geometric progression.

Execution Time

Tables 5.11 and 5.12 show the execution time of variable proportion sampling, using a linear and geometric progression respectively. The results are compared to no sampling, and the result of standard sub-sampling with the greatest accu-

Table 5.10: Comparison of number of nodes for different initial proportions with a geometric progression

	1%	2%	5%	10%
1%	—	8:3	3:8	4:7
2%	3:8	—	2:9	4:7
5%	8:3	9:2	—	6:5
10%	7:4	7:4	5:6	—

racy, taken from Chapter 4.

The results show that when a linear progression is used, execution time is generally less than that of no sampling and the most accurate standard sub-sampling. A geometric progression is also generally quicker than no sampling, but is slower than the most accurate standard sub-sampling on more occasions than it is quicker.

Comparing the same initial proportion between linear and geometric progressions shows the linear progression is generally quicker. With an initial proportion of one or two per cent, linear is faster than geometric eight times and slower three. With an initial proportion of five per cent, linear is quicker six times and slower five. With ten per cent, linear is quicker eight times and slower three. This is not to be unexpected, as a geometric progression results in more instances being used to determine a split higher in the tree.

Tables 5.13 and 5.14 compare different initial proportions for linear and geometric progressions respectively. The tables show win:loss records, where wins relate to the initial proportion in the row having less execution time than the proportion in the column. Note that there was one tie when comparing initial proportions of one per cent and two per cent with a linear progression.

The tables show that smaller initial proportions generally result in less execution time. Although the only results that can be considered statistically

Table 5.11: Execution time of variable proportion sampling using a Linear progression

Data Set	No	Standard	1%	2%	5%	10%
Adult	13.26	14.50	11.86	12.13	12.78	12.74
Census	86.61	98.13	96.04	100.53	104.74	106.83
Connect-4	6.74	6.42	25.82	20.39	12.80	9.26
Cover Type	1818.31	1899.43	1897.77	1979.06	1964.17	1914.67
Distinct	289.65	232.02	207.08	222.33	240.97	260.44
Fuzzy	359.93	127.07	111.72	123.56	188.23	247.41
IPUMS	86.19	85.53	71.35	74.31	69.99	76.39
Random	181.70	140.20	125.71	108.64	137.74	155.38
Shuttle	5.15	1.38	1.15	1.15	1.66	2.24
Sleep	240.80	237.59	231.32	195.95	227.77	235.76
Waveform	1688.54	1024.99	1436.20	1760.54	1050.16	1025.70

Table 5.12: Execution time of variable proportion sampling using a Geometric progression

Data Set	No	Standard	1%	2%	5%	10%
Adult	13.26	14.50	12.09	13.67	14.25	13.66
Census	86.61	98.13	91.37	90.86	80.41	94.06
Connect-4	6.74	6.42	13.00	11.98	10.52	8.98
Cover Type	1818.31	1899.43	2042.73	2067.22	1739.75	1885.19
Distinct	289.65	232.02	240.60	243.35	239.57	262.31
Fuzzy	359.93	127.07	214.49	227.76	269.17	278.31
IPUMS	86.19	85.53	85.81	88.21	77.74	79.41
Random	181.70	140.20	145.43	147.69	161.93	165.18
Shuttle	5.15	1.38	1.39	1.64	2.42	3.18
Sleep	240.80	237.59	234.99	235.86	232.97	238.24
Waveform	1688.54	1024.99	1125.21	1178.27	1014.64	1070.50

Table 5.13: Comparison of execution times for different initial proportions with a linear progression

	1%	2%	5%	10%
1%	—	7:3	7:4	9:2
2%	3:7	—	7:4	8:3
5%	4:7	4:7	—	7:4
10%	2:9	3:8	4:7	—

Table 5.14: Comparison of execution times for different initial proportions with a geometric progression

	1%	2%	5%	10%
1%	—	9:2	4:7	7:4
2%	2:9	—	4:7	6:5
5%	4:7	7:4	—	9:2
10%	7:4	5:6	2:9	—

significant at the 0.05 level with a one-tailed binomial sign test are those with ratios of 9:2 ($p = 0.0327$), it is clear that increasing initial proportion when using a linear progression generally increases execution time. The trend also exists with a geometric progression, although it is not as strong. This behaviour is likely due to the reduced time required at nodes high in the tree.

The results for the Connect-4 data set again show the sampling methodology under investigation has much greater execution time than no sampling. For both types of progressions, an increase in initial proportion relates to a decrease in execution time.

5.3.3 Summary

The results show that variable proportion sampling can be expected to perform with accuracy less than, but close to, that obtained without sampling. The type of progression makes little difference, although geometric is slightly better. Taking execution time into account shows both to be generally quicker than no sampling, with linear being slightly quicker than geometric.

In terms of accuracy, variable proportion sampling fares a little worse against the best result for standard sub-sampling than it does against no sampling. This is because the bar is raised slightly by using the best standard sub-sampling.

However, although accuracy is generally worse, the difference is small. In terms of execution time, linear is often quicker, while geometric is often slower.

Comparing types of progression shows that a geometric progression is generally more accurate than a linear progression, while a linear progression is generally quicker.

Larger initial proportions also generally result in more accurate models. This is not to be unexpected, as increasing the initial proportion reduces the difference in behaviour between variable proportion sampling and no sampling.

Further experimentation into varying different parameters may prove useful. Ways in which sample size is calculated, either via different types of progression or different rates of increase could be investigated, as could the effect of changing the minimum sample size. However, the multitude of potential parameters makes such experimentation unwieldy at best. The parameters investigated above relate to the very nature of variable proportion sampling.

5.4 Statistically Determining Sample Size

The previous experiments showed variable proportion sampling to generally be slightly less accurate than standard sub-sampling, while often taking less execution time. Although variable proportion sampling somewhat alleviated the deficiencies of using a static sample size, problems still exist. It is still possible for the sample to be too small to determine an appropriate split, or larger than necessary, therefore wasting execution time.

This section investigates statistical measures with which sample size could be determined dynamically. Issues with these methods are discussed, and an algorithm that implements an appropriate method is introduced. Experiments are performed, and the results compared to other sampling methodologies.

5.4.1 Desirable Qualities

One of the properties common to simple sub-sampling and variable proportion sampling is that the way in which sample size is determined is computationally inexpensive. With simple sub-sampling, size specified as a parameter. Variable proportion sampling requires only a simple, low-cost calculation. However, a method which employs a statistical determination of sample size is likely to not have this luxury. Therefore, before investigating potential methods to statistically determine sample size, it may be useful to identify necessary and desirable qualities of the determination method. These qualities are outlined below.

1. *Use the right amount of data.* The purpose of statistically determining sample size is to attempt to use as much data as necessary to find an appropriate split, but not more than required. Using less data than necessary is likely to lead to inappropriate split selection, while using more will waste execution time.

2. *Not increase time.* Wherever possible, determination of sample size should not increase the time required to build a node to more than using all available data. In other words, the cost of selecting the sample plus inferring the split should not exceed the cost of learning from all available data.

3. *Allow control over the trade-off between accuracy and time.* Whenever sampling is used, it can be expected there will be a trade-off between predictive accuracy of the final model and execution time. Rather than making this trade-off static, the practitioner should be able to have input into the trade-off they are willing to accept. For example, increasing confidence in the chance of the most appropriate split being selected at a node is likely to come at the cost of increased execution time. It is sensible that the user of the algorithm should be able to decide how important these factors are in relation to each other and to the task at hand.

4. *Deal with continuous attributes.* Many data sets contain continuous attributes. An algorithm which ignores continuous attributes limits its practicability dramatically. Alternatively, continuous attributes could be discretised. However, this compromises the potential descriptiveness of inferred models. Also, if discretisation is performed prior to input to the learning algorithm, it is possible that potentially fruitful cut-points are forever lost. None of these situations are desirable.

5.4.2 Methods for Determining Sample Size

While much research has been performed into methods for determining appropriate sample size when using pre-sampling, little has been done to determine appropriate sample size when using sub-sampling. However, it may be possible to implement a method used in contexts other than sub-sampling within a sub-sampling framework. A discussion of methods previously used to determine appropriate sample size follows. Issues regarding implementation of these methods within a sub-sampling framework are also discussed.

Learning Curves

A popular method used to determine sample size within a pre-sampling context is the estimation of learning curves. A learning curve is a plot of model accuracy against sample size. The premise behind learning curves is the assumption that an increase in sample size will lead to an increase in the accuracy of inferred models. A point will eventually be reached after which adding extra instances to the sample has negligible or no effect on predictive accuracy. This point is the beginning of the learning curve "plateau," and can be considered the optimal sample size as it finds the balance between minimising loss in accuracy due to sampling and maximising execution time saved by sampling. This optimal sample size can then be used to train the final model.

Within a sub-sampling context, samples of increasing size could be taken at a node, with a utility measure recorded for each attribute. Each attribute could be measured until the utility measure had converged, and the attribute with the best measure selected for the split. In this way, splits may be determined using less than the available data set, resulting in a reduction in time.

However, the drawback of such a method is that it requires a utility measure that is monotonic with increasing sample size. Measures such as the information gain used by C4.5 are not monotonic in the number of instances [97]. Learning curves can therefore not be used if such measures are to be employed.

Clustering and Data Squashing

Rather than determine an appropriate sample size, methods such as clustering [36] and data squashing [34] determine an appropriate sample data set. The premise behind these methods is that an unsampled data set can be adequately represented using fewer, weighted instances. Clustering selects a subset of the input data to represent the entire data set, giving each instance in that subset a weight corresponding to the number of instances it represents. Data squashing, on the other hand, creates pseudo-instances. These pseudo-instances are not necessarily instances in the original data set. Their purpose is to allow for compression of the data while reducing information loss, achieved by ensuring that averages and moments of the original data are maintained in the squashed data. Clustering does not ensure this.

Clustered and squashed data sets can produce accurate models [32, 34]. However, the drawback of these methods is that they are computationally expensive. Although they may be useful in a pre-sampling context, their cost is unlikely to enable them to be used in a sub-sampling context.

Adaptive Sampling

Another method that is commonly used to determine sample size is to make a statistical calculation of the sample size required for accurate estimation of a concept. Procedures such as adaptive or sequential sampling [28, 99, 92, 60] dynamically calculate the number of instances required to estimate a parameter of a data set within a given margin of error with a given confidence level. In a sub-sampling context, the parameter to be estimated could be a measure of utility of an attribute.

The drawback of many adaptive sampling methods is that they require the process with which the parameter is estimated be incremental. By its very nature, computing utility via an entropy based measure is not incremental when continuous attributes are involved, as inclusion of extra instances requires re-sorting of attribute values. Although some adaptive sampling methods do not require incrementalism, they may remain inappropriate, as continual re-sorting of continuous attributes is still required.

Alternatively, continuous attributes could be discretised. As mentioned in the desiderata outlined in Section 5.4.1, global discretisation is undesirable. Thus, attributes must be discretised locally, before the adaptive sampling process begins. Although this discretisation could be done with little cost, any split that is subsequently found relates to the discretised attribute, and not the original continuous attribute. This has not satisfied the desiderata.

Statistical Measures

Another alternative is to use a statistical measure of how similar the sample is to the data from which it was selected. There are a number of well defined statistical measures to do this. However, many of these are unsuitable for the required purpose.

Firstly, the chi-square goodness-of-fit test cannot be used for two reasons.

The first is that rather than measuring whether a sample represents a known distribution, as desired here, this test measures whether a hypothesised distribution fits the data. In the desired context, the distribution from which the sample was selected will be known. The second reason is that the test will always show the distribution is not evidenced by the sample, if given a large enough sample [22, page 191]. Since sub-sampling is designed for large training sets, this rules out possible use of this measure.

Contingency tables also cannot be used. Contingency tables can be used to determine whether two data sets are likely to come from the same source. However, one of the assumptions of contingency tables is that the data sets compared are independent samples. Comparing a sample to the population from which it was selected violates this critical assumption.

Another deficiency of the chi-square goodness-of-fit test and contingency tables is that they require continuous values to be discretised. Rank-based methods, such as the Wilcoxon test, overcome this problem, but introduce others. First, rank-based methods require continuous values to be sorted. Within a sub-sampling context, this would defeat the purpose of performing sampling, as the process of sorting all available data so that the test can be performed, selecting a sample, performing the test, then inferring a split, would almost certainly result in the whole process taking longer than using all the available data to infer a split. Measures of the Kolmogorov type also suffer from this problem. The second problem with rank-based methods is they also assume that sampled values are drawn from a normal distribution. This clearly cannot be guaranteed in any real-world data set.

Regardless of these issues, John and Langley [60] use a chi-square test to determine if frequencies of discrete values in a sample are sufficiently similar to those from which the sample was drawn. To determine whether sampled continuous values are sufficiently similar, a comparison of large-sample means is performed. If all attributes are determined to be sufficiently similar, the sample is then considered sufficiently similar. However, there are two problems with this method in the current desired context. First, there is no guarantee that any sample will be large enough to accurately determine the true mean of a continuous attribute. Second, the way in which to allow a trade-off between sample sufficiency and execution time is not clear. It is possible to relax the bounds for each attribute, but the cumulative effect of this may be greater than expected. Alternatively, it may be not be necessary for all attributes to be considered sufficiently similar. This then introduces the problem that a split may be selected based on an attribute for which the sample is not representative of the available data. This could be remedied by requiring that splits can only be based on attributes that are considered sufficiently similar, but the problem then exists that an attribute not considered sufficiently similar may be the correct attribute on which to split at that node.

Methods such as the Smirnov test and the Cramér-von Mises test for two samples also can be used to determine if two data sets are drawn from the same distribution. However, as with contingency tables, these measures require that the samples being compared are independent.

5.4.3 Information Divergence and Sample Quality

A method suitable for the problem of determining sample size in progressive sampling has been proposed by Gu, Liu, Hu and Huan [50]. The paper introduces the Statistical Optimal Sample Size — a method for determining the minimum sample size that should be employed when estimating learning curves using a geometric sampling progression [84]. The method gathers information about the full training set (i.e. distribution of attribute values) and compares this to a random sample, producing a measure of the "quality" of the sample. If the quality of the sample is less than a supplied threshold, a larger sample is taken and the process repeated. If the sample quality is above the threshold, the sample is considered to be sufficiently similar to the full training set. This is then considered the minimum sample size that should be investigated, as models built from samples smaller than this size are unlikely to produce accurate models.

The quality of the sample as compared to the whole data set is defined as

$$Q(S) = e^{-J} \tag{5.1}$$

where

$$J = \frac{1}{r}\sum_{k=1}^{r} J_k(S, D) \tag{5.2}$$

S and D represent the sample and full data set respectively and r is the number of attributes [50]. $J_k(S, D)$ is taken as the measure of information divergence [68] between the sample and population for attribute k, defined as

$$J_k(1, 2) = \sum_{j=1}^{c}(p_{1jk} - p_{2jk}) \ln(p_{1jk}/p_{2jk}) \tag{5.3}$$

where c is the number of discrete values for attribute k, and p_{ijk} is the probability of the j-th value of attribute k occurring in population i. This probability can be estimated by taking a count of each value in the data set.

Although the above method is designed to eliminate the need to build unnecessary small models when estimating a learning curve, a similar process could be applied in a sub-sampling context. Samples of increasing size can be taken until a sample sufficiently similar to the available data is found. This would provide a minimum size with which to sample at a node. Although this only provides a minimum size with which to sample, this may be sufficient in sub-sampling.

This measure can be viewed as rejecting samples that are not similar enough to the available data. Note that since a sample must be drawn to determine whether it is sufficiently similar, determining the sample size also provides the sample. This precludes the possibility of using the sample quality measure to pre-determine sample size, as sample quality can only be evaluated with respect to a specific sample. However, since the measure is to be applied within a sub-sampling context, this is acceptable.

The method also provides a means for practitioners to manage the trade-off between accuracy and time via the threshold of sufficient quality. A lower threshold can be expected to produce less accurate models in less time, as smaller samples are then more likely to satisfy sufficiency.

Sub-sampling using the sample quality measure to determine sample size at each node shall henceforth be referred to as SQ sampling.

5.4.4 Modifying Sample Quality for Continuous Attributes

A drawback of the sample quality measure as presented by Gu et al. [50] is that it is designed for only discrete attributes. Continuous attributes must be discretised. As mentioned previously, it is undesirable to do this on a global scale (i.e. before data is input to the learning algorithm) as discretisation reduces information contained within data. Discretisation would also make the method unusable with algorithms that use only continuous attributes, such as OC1.

Therefore, to meet the desiderata of Section 5.4.1, the sample quality measure must be able to use continuous attributes without requiring global discretisation. This can be achieved in SQ sampling by discretising continuous attributes locally at each node, and only for the purpose of determining if the sample is sufficiently similar. Once this has occurred, the discretisation can be discarded and the original continuous values used.

A necessary consideration of SQ sampling is the time required by sampling overhead. To see how this can be managed, consider SQ sampling implemented within C4.5. The information divergence measure requires collecting information about the available data to determine if a sample is sufficiently similar. Since this information is independent of the sample, this should be performed before a sample is drawn, so that it can be re-used, if required, for further samples. Determining the utility of a split should also be postponed until a sample is selected.

When determining the utility of a split for a discrete attribute, C4.5 iterates through all instances, taking a count of the values of the attributes and class. This requires one pass over the available data. Since the information divergence measure requires knowledge of the available data, it must perform a similar process, even before a sample is considered. Once a sample is selected, it must then undergo the C4.5 process of determining split worth. Therefore, SQ sampling cannot reduce execution time required for determining the value of a split of a discrete attribute. However, not including discrete attributes when measuring sample quality would defeat the purpose of taking the measurement. It can be expected that using SQ sampling on a data set that contains only discrete attributes would take longer than using default C4.5.

Since it is not possible for SQ sampling to be quicker with discrete attributes, any reduction in execution time must occur with continuous attributes. When determining the worth of a split for a continuous attribute, C4.5 sorts the attribute values. This requires multiple passes over the node data. Sorting is an expensive operation, and reducing the number of values to be sampled can save substantial execution time. This provides an excellent opportunity for SQ sampling to reduce execution time.

As previously mentioned, the information divergence measure requires continuous values to be discretised. There are many ways this can be performed [38, 105, 39]. All discretisation methods work by placing continuous values into "bins" and assigning a discrete value to each bin. Different methods use different ways of determining the number and size of bins. However, methods that determine bins based on measures of entropy or error require sorting of the data (e.g. [38]), negating gain in execution time from using a smaller set to determine split worth. Methods that use equal frequency intervals (e.g. [105]) also require sorting.

One of the simplest methods of determining bins is to use intervals of equal width. Here, a pre-selected number of bins is used with the width of each bin calculated by

$$width = \frac{max\,value - min\,value}{number\,of\,bins}$$

Such a method does not require sorting of attribute values.

Fixed-width binning is not the best way to perform discretisation. Research has shown the Fayyad and Irani entropy method [38] tends to result in classifiers with higher accuracy than other discretisation methods [33, 65]. However, the difference between this method and fixed-width binning is not great. The accuracy of models inferred after discretisation using the Fayyad and Irani method and two types of fixed-width binning has been compared [33], and although the Fayyad and Irani method produces models with greater accuracy than fixed-width binning, the difference is seldom statistically significant. When using C4.5 as the induction algorithm, it has been shown that Fayyad and Irani's discretisation produces more accurate models than fixed-width binning using $2\log l$ bins ten times, and less accurate models five times [33]. The result of a one-tailed sign test is $p = 0.1509$. Using ten bins, the ratio is 12:4 ($p = 0.0384$). When using Naïve-Bayes as the induction algorithm and $2\log l$ bins, the ratio is 6:9 ($p = 0.8491$) and with ten bins 9:7 ($p = 0.4018$).

Although the purposes of discretisation in the above research and this research are different, the above research provides an indication that fixed width binning may be suitable for SQ sampling. This is because fixed-width binning has small computational requirements, while maintaining reasonable performance. Since discretisation will be used in this research to determine similarity of distributions rather than to reduce error, fixed-width binning should be adequate.

5.4.5 SQ Sampling

Thus far, it has been argued that the most appropriate way of selecting a sufficient sample is with Kullback's information divergence measure. A method that extends this measure to allow use of continuous attributes without loss of information in the final models has also been proposed. The method of discretisation has been discussed, with the conclusion drawn that fixed-width bins are likely to provide the best balance between informative bins and execution time.

However, there are still issues that must be resolved. First, the way in which sample size will be increased when a sufficient sample has not been found must be decided . Although it is possible to determine sufficiency by increasing sample size one instance at a time, and this will guarantee the minimum sample size which can be determined sufficient will be found, doing such is likely inefficient. Therefore, some sort of progression of sample sizes must be used. Following the work of Gu et al. [50] and Provost, Jensen and Oates [84], it is proposed that a geometric progression is used. The results of Section 5.3, although in a slightly different context, also support the use of a geometric progression.

The second issue regards enforcing a minimum sample size. Similar to variable proportion sampling, it is unreasonable to expend the resources required for determining sufficiency if the available data at a node consists of only a small number of instances. In these circumstances, the overhead of performing the sampling and determining sufficiency is likely to be greater than the cost of using all available data. Therefore, a minimum sample size should be employed, below which sampling is not performed.

5.4.6 Experiments

Experiments were performed to compare the performance of SQ sampling to standard sub-sampling and no sampling in terms of accuracy, model complexity, and execution time. The methodology and data sets used are the same as described in Section 4.4. Sampling is performed randomly, without replacement. This was performed as the purpose of the quality measure is to evaluate how close the attribute value distribution of a sample is to that of the population. Replacement, and stratified and disproportionate sampling alter this distribution in the sample.

Three parameters are required for SQ sampling. The minimum sample size and the starting sample size were both set to 1,000 instances for all experiments. Six sample quality settings were used, ranging from 0.95 to 0.9999. Local discretisation was performed using four fixed-width bins.

5.4.7 Results

Following from Section 5.3.2, tables compare results for SQ sampling to no sampling and the most accurate standard sub-sampling. Again, since there is no direct mapping between standard sub-sampling and SQ sampling, only the most accurate result achieved with standard sub-sampling is used.

Accuracy

Table 5.15 shows the accuracy of SQ sampling for the sample quality settings used. Results are compared to no sampling, and the greatest accuracy obtained using standard sub-sampling.

The results show that SQ sampling is consistently less accurate than no sampling. The only occasion on which SQ sampling is more accurate than the most accurate standard sub-sampling is with a sample quality of 0.9999 on the Adult data set.

Table 5.15: Accuracy of SQ sampling

Data Set	No	Standard	0.95	0.99	0.995
Adult	0.8602	0.8589	0.8468	0.8469	0.8469
Census Income	0.9526	0.9526	0.9463	0.9463	0.9462
Connect-4	0.7981	0.7999	0.7789	0.7789	0.7788
Cover Type	0.9375	0.9406	0.9121	0.9121	0.9121
Distinct Boundary	0.9990	0.9991	0.9983	0.9984	0.9984
Fuzzy Boundary	0.9985	0.9985	0.9979	0.9979	0.9982
IPUMS	0.9282	0.9284	0.9278	0.9278	0.9283
Random Binary	1.0000	1.0000	0.5019	0.5019	0.5019
Shuttle	0.9996	0.9997	0.9992	0.9992	0.9992
Sleep	0.7299	0.7310	0.7304	0.7307	0.7298
Waveform	0.9857	0.9857	0.9848	0.9849	0.9851
Data Set	No	Standard	0.999	0.9995	0.9999
Adult	0.8602	0.8589	0.8495	0.8521	0.8610
Census Income	0.9526	0.9526	0.9473	0.9484	0.9505
Connect-4	0.7981	0.7999	0.7786	0.7836	0.7984
Cover Type	0.9375	0.9406	0.9121	0.8977	0.9257
Distinct Boundary	0.9990	0.9991	0.9988	0.9989	0.9990
Fuzzy Boundary	0.9985	0.9985	0.9984	0.9985	0.9985
IPUMS	0.9282	0.9284	0.9283	0.9275	0.9281
Random Binary	1.0000	1.0000	0.7343	0.9834	1.0000
Shuttle	0.9996	0.9997	0.9995	0.9995	0.9994
Sleep	0.7299	0.7310	0.7305	0.7303	0.7296
Waveform	0.9857	0.9857	0.9855	0.9856	0.9857

Table 5.16: Number of times SQ sampling is within bounded accuracy losses

Bounded	Required Sample Quality					
Loss	0.95	0.99	0.995	0.999	0.9995	0.9999
1%	7:4	7:4	7:4	7:4	8:3	10:1
2%	8:3	8:3	8:3	8:3	10:1	11:0
5%	10:1	10:1	10:1	10:1	11:0	11:0

Table 5.16 presents win:loss records for sample quality settings with one, two, and five per cent bounded accuracy loss. The table shows the two highest quality settings often achieve accuracy within one per cent bounded loss. The sole exception for the highest quality setting is the Cover Type data set. It is interesting that the records for the lowest four qualities are identical. This suggests that varying the quality setting between 0.95 and 0.999 makes little systematic difference to accuracy.

The results for the Random Binary data set are interesting. On this data set, all sample quality settings of 0.995 and less produced models that perform only marginally better than chance. An investigation into SQ sampling with this data set showed that a sample with sufficiently high quality was always found at the root node with the minimum sample size, i.e. 1,000 instances. This explains the poor performance, as both pre-sampling and standard sub-sampling showed similar behaviour with this sample size and data set in Chapter 4. Changing the minimum sample size to 2,000 instances was investigated, and resulted in accuracy of 1.000 for all sample quality settings.

Model Complexity

Table 5.17 shows the mean number of nodes inferred by SQ sampling for the sample qualities used. Results are compared to no sampling, and the greatest accuracy obtained from standard sub-sampling.

Table 5.17: Nodes Inferred by SQ sampling

Data Set	No	Standard	0.95	0.99	0.995
Adult	600.4	486.8	229.6	217.5	211.3
Census Income	1634.5	1536.8	259.7	261.0	297.0
Connect-4	4708.2	3302.8	2942.3	2942.3	2997.9
Cover Type	23730.4	22141.2	20420.5	20420.5	20420.5
Distinct Boundary	1430.7	1444.9	1302.7	1291.1	1310.0
Fuzzy Boundary	454.9	454.7	326.1	328.3	367.5
IPUMS	1414.4	1418.9	1424.6	1424.6	1378.7
Random Binary	78838.3	78801.3	1.0	1.0	1.0
Shuttle	46.7	37.3	52.1	50.5	51.5
Sleep	9212.3	9131.7	9191.9	9146.8	9190.0
Waveform	4107.5	4015.9	4437.5	4516.4	4347.8

Data Set	No	Standard	0.999	0.9995	0.9999
Adult	600.4	486.8	516.7	411.9	453.6
Census Income	1634.5	1536.8	549.6	352.0	610.7
Connect-4	4708.2	3302.8	2950.0	2806.1	3273.0
Cover Type	23730.4	22141.2	19309.9	19745.9	20979.3
Distinct Boundary	1430.7	1444.9	1376.9	1400.1	1427.9
Fuzzy Boundary	454.9	454.7	411.9	417.7	432.8
IPUMS	1414.4	1418.9	1417.7	1483.3	1435.7
Random Binary	78838.3	78801.3	36749.5	76202.9	78846.9
Shuttle	46.7	37.3	48.1	46.8	46.7
Sleep	9212.3	9131.7	9097.9	9200.3	9194.7
Waveform	4107.5	4015.9	4350.7	4134.1	4045.4

The results show that SQ sampling generally infers less complex models than both no sampling and the most accurate standard sub-sampling. A sample quality of 0.9999 compared to the most accurate sub-sampling is the only case where this is not true, with SQ sampling inferring smaller models five times and larger six times.

Increasing sample quality shows no clear trend of an impact on model complexity. Comparing adjacent sample quality settings over all data sets shows that increasing quality results in an increase in model complexity 29 times, and a decrease 20 times, with six occasions where there was no difference in complexity.

Execution Time

Table 5.18 shows the time taken for SQ sampling for the sample qualities used. Results are compared to no sampling, and the greatest accuracy obtained from standard sub-sampling.

The results show that SQ sampling generally takes less execution time than no sampling. Only the highest sample quality setting is slower than no sampling more times than it is faster. SQ sampling is also generally quicker than the most accurate standard sub-sampling. However, the two highest quality settings are slower than standard sub-sampling more often than they are faster.

The results for a sample quality setting of 0.9999 on the Distinct Boundary, Fuzzy Boundary, and Waveform data sets show significant decreases in execution time compared to no sampling. It is interesting that this quality setting produced models with accuracy equal to that of no sampling on these data sets.

Again, the execution time required for the Connect-4 data set is much greater than that of no sampling. However, the results show that the sample quality setting has little impact on execution time.

Table 5.18: Execution time of SQ sampling

Data Set	No	Standard	0.95	0.99	0.995
Adult	13.26	14.50	13.00	12.89	13.15
Census Income	86.61	98.13	121.47	103.64	130.09
Connect-4	6.74	6.42	38.73	38.16	39.38
Cover Type	1818.31	1899.43	1366.46	1467.94	1377.29
Distinct Boundary	289.65	232.02	215.70	178.04	227.71
Fuzzy Boundary	359.93	127.07	85.87	72.36	117.73
IPUMS	86.19	85.53	70.27	81.26	78.34
Random Binary	181.70	140.20	13.53	11.77	99.95
Shuttle	5.15	1.38	1.57	1.57	3.10
Sleep	240.80	237.59	233.39	195.57	236.59
Waveform	1688.54	1024.99	940.10	1139.75	1090.16
Data Set	No	Standard	0.999	0.9995	0.9999
Adult	13.26	14.50	13.51	13.70	14.59
Census Income	86.61	98.13	130.09	129.77	115.78
Connect-4	6.74	6.42	47.80	46.30	10.31
Cover Type	1818.31	1899.43	1377.29	1504.61	1606.11
Distinct Boundary	289.65	232.02	227.71	232.42	210.53
Fuzzy Boundary	359.93	127.07	117.73	126.42	134.28
IPUMS	86.19	85.53	78.34	82.22	89.26
Random Binary	181.70	140.20	99.95	199.19	199.03
Shuttle	5.15	1.38	3.10	4.03	5.23
Sleep	240.80	237.59	236.59	237.85	211.92
Waveform	1688.54	1024.99	1090.16	978.15	1088.14

5.4.8 Summary

SQ sampling results in models being built with less predictive accuracy than no sampling or the most accurate standard sub-sampling. This generally comes with a decrease in execution time, with higher sample quality settings generally taking less execution time than lower sample qualities.

An analysis of sample sizes used shows that a sample of 1,000 instances is generally considered sufficiently close to the available data, regardless of the characteristics of the data set. This may suggest that higher levels of quality setting are necessary.

It is difficult to suggest that SQ sampling is effectively solving the problem of large data. Although accuracy close to that obtainable from the full training set can be achieved, the time savings in doing so are often marginal. However, there are significant savings with three of the four largest data sets. This may suggest that SQ sampling could provide larger savings in execution time with larger data sets.

5.5 Conclusions

Experiments were performed to investigate potential methods of improving sub-sampling. Comparisons were made to the existing methodologies of standard sub-sampling, and using the full training set.

An investigation into the effects of sampling without disproportionate sampling and with replacement was performed. This showed non-disproportionate sampling to result in greater loss of accuracy than sampling with replacement. Non-disproportionate sampling also results in a reduction in execution time less frequently than sampling with replacement. Although there are some occasions where substantial amounts of time are saved, the time savings are generally marginal.

Two new methodologies for determining sub-sample size were also introduced. Variable proportion sampling produces models with accuracy close to that obtainable with no sampling. Although accuracy is rarely greater than no sampling, it is often within one per cent of that of no sampling, and always within two per cent. This cost in accuracy brings with it a saving in execution time. Using a linear rather than geometric progression for determining sample size generally results in a reduction of both accuracy and execution time. This difference in execution time is generally small.

SQ sampling was introduced to attempt to statistically determine appropriate sample size on a node-by-node basis. Models produced using SQ sampling generally suffer from a loss in accuracy. However, model accuracy generally stays within one per cent bounded accuracy loss, especially with higher sample quality settings. Substantial reductions in execution time can also be gained, without loss of accuracy, when used on large data sets.

Finally, the method of obtaining the statistically optimal sample size has been extended from the ability to use only discrete attributes to also include continuous attributes. This removes the need for global discretisation, and thus losing information, from continuous attributes.

Chapter 6

The Problem with Learning Curves

A learning curve is a plot of accuracy against training set size for a given learning algorithm and data distribution. It is widely accepted that learning curves usually conform to a standard shape, and that accuracy usually increases as training set size increases. The general shape of a learning curve introduces an interesting expectation: after a certain data and algorithm dependent training set size is reached, accuracy either does not increase, or increases only negligibly, with the addition of further instances to the training set. The size at which this accuracy "plateau" begins can be viewed as the optimal training set size for efficient learning as it minimises expected accuracy loss while also minimising the size of the training set required to achieve that accuracy.

Learning curves therefore provide what is often considered a viable method for a practitioner to reduce execution time while maintaining accuracy equal or acceptably close to that obtainable when learning from the full training set. If the optimal size can be found or closely estimated, then there is little to be gained by expending the resources required to learn from larger training sets.

Much research has been performed into methods that use learning curves to improve efficiency by either directly or indirectly calculating this optimal size [53, 71]. However, a significant problem lies in the fact that developing an accurate learning curve method by creating models for every training set size is impractical. Therefore, for a learning curve method to reduce execution time, the curve *must* be estimated. As with any estimation, an estimated learning curve can be expected to contain some error. Research has therefore been performed to investigate methods that attempt to mitigate the cost, both in terms of loss of accuracy and expense of computational resources, of estimation of learning curves and determination of the optimal sample size.

The contention of the current research is that a number of these methods involve higher risk than is currently understood. This risk can manifest itself as increased execution time over the full training set or substantial loss of accuracy compared to that obtainable with the full training set. This chapter investigates the conditions under which learning curves may not be a viable method of reducing execution time while maintaining acceptable accuracy.

6.1 The Case Against Learning Curves

To be usable for the purpose of reducing execution time, a learning curve must be estimated. This requires induction of a number of models using training sets of varying size. The progression of sample sizes used is considered a sampling schedule. Regardless of the method used to estimate the learning curve and the sampling schedule used, the purpose is to find the minimum training set size after which including more instances will not be beneficial. This section discusses methods used to find this size, and outlines potential problems with each method.

6.1.1 Curve Extrapolation

Possibly the least computationally expensive method of estimating a learning curve is to find the accuracy of a small number of models built with small sample sizes, and extrapolate a curve. Extrapolation is performed by fitting parameters of a function that has the general form of a learning curve so that error between the function and measured accuracies is minimised. The optimal training set size can then be found by inspecting the curve to find the smallest training set size that is acceptably close to the projected accuracy obtainable from the full training set.

There are a number of types of functions which fit the general form of a learning curve. Both logarithmic and exponential functions can fit learning curves well, although it has been shown that a power function of the form $E = xS^{-y}$, where E is the error rate, S is the sample size, and x and y are parameters fit to observed accuracies, provides a more accurate model [42]. However, a three-parameter power law of the form $E = z - xS^{-y}$ provides an even better fit [49].

However, the problem with methods of this type is that learning curves generally consist of three phases [84]. The first phase is a steep rise in accuracy with little increase in sample size. The second is a much less steep rise in accuracy with reasonable increase in sample size. The third is a long plateau, where accuracy does not increase substantially with additional training data. Although these phases are not disjoint, they are distinct, and hence it will be difficult to fit a single function to accurately estimate all three phases. This will be exacerbated if the observed accuracies used to fit the data are drawn from the first two phases only, and even more so if drawn from only the first phase. However, since the purpose of estimating the curve is to reduce resource costs and to find the optimal sample size, including observed accuracies from the plateau phase defeats the purpose of using this method of estimating learning curves.

It may be argued that extrapolation methods could be iteratively used with increasing samples sizes to provide better estimations of the curve, until a model is built with accuracy close to that of the full training set. However, the problems with this approach are two-fold. As will be seen in Section 6.2, such a method does not guarantee efficiency. A larger problem, though, is that such a method relies on the projected accuracy of the full training set being reliable. In their experiment analysis, Gu, Hu and Liu [49] show the difference between the extrapolated three-parameter power law estimation of the learning curve and the measured accuracy of the full training set can be more than five per cent. Compounding this problem is the unpredictability of whether the extrapolated curve under-estimates or over-estimates actual obtainable accuracies. These factors raise doubts as to the utility of curve extrapolation methods.

6.1.2 Curve Interpolation

A more accurate method of estimating a curve than extrapolation is interpolation. Rather than the estimated points lying outside observed accuracies as with extrapolation, interpolation requires that estimated points are bounded on each side by observed accuracies. However, this suffers a similar fate to extrapolation of a curve in that effectiveness of the method requires observed accuracies contained within the plateau phase of the curve. Again, due to the resources required to achieve this, it is likely that interpolating a curve will defeat the purpose of using a learning curve.

6.1.3 Gradient of Tangent

A third method of finding the optimal sample size does not attempt to estimate the curve itself. Rather, it attempts to calculate when the plateau has been reached by analysing the gradient of the tangent to the curve. When the gradient is sufficiently close to zero, the curve is considered to have "converged" and the plateau reached.

Such estimation can be performed by analysing the gradient on either a global or local scale. When analysing global gradients, widely disparate sample sizes are compared, with the gradient determined between neighbouring samples. This has the advantage that relatively few samples must be taken, and small variations in accuracy due to the specific sample used for the training set are unlikely to have significant impact on the gradient. However, the disadvantage is that it if the learning curve does not plateau until large training set sizes are reached, it is possible that convergence will not be detected and the full training set must be used. Global scale gradient analysis is used in the work of John and Langley [60].

When analysing local gradients, a number of points are sampled around a selected sample size and the accuracies of models inferred from these additional samples used to find a line-of-best-fit. Again, if the gradient of this line is sufficiently close to zero, the curve is considered to have converged. However, the drawback of this method is that it relies on the learning curve being strictly monotonic until the plateau is reached. If the curve is not strictly monotonic, the method may incorrectly detect global convergence when it has found a local accuracy plateau. For example, the results presented in Sections 4.5.1 and 5.2.3 regarding the Random Binary data set showed that a sample size of 1,000 instances consistently produces accuracy rates of 0.5019. It is likely that inferring models from slightly different sample sizes will have little, if any, impact on accuracy, as the results suggest that a critical mass of instances must be reached before accuracy improves. Therefore, a method that analyses local gradients will likely accept that the curve has converged, even though larger sample sizes produce models with much greater accuracy.

A second problem with analysing local gradients is that the precision of estimates of local gradients can be expected to suffer from variations in accuracy due to the specific set of instances selected for a sample. These variations could potentially result in the curve appearing flatter or steeper than it actually is, leading to possible false detection of convergence or false non-detection of convergence respectively. To alleviate this problem, averaged results of many samples of the same size must be used, substantially increasing required computational resources and execution time.

The following sections further investigate the risks of using learning curves when using gradient of tangent methods. Since problems with global gradient analysis have been thoroughly discussed in other work [84], these sections focus on local gradient analysis methods. Specifically, the sections investigate the Linear Regression with Local Sampling (LRLS) method [84]. An analysis of computational requirements is performed, as are experiments investigating the reliability of convergence detection using analysis of local gradients of the curve.

6.2 Conditions Required for Efficiency of Learning Curves

To be computationally efficient, a learning curve method must take less execution time than learning from the full training set. It could even be argued that a learning curve method should take substantially less time than learning from the full training set, in order for the risk of a potential accuracy sacrifice to be worthwhile. Given this risk, a learning curve method should also be more computationally efficient than simply selecting and learning from a large sample of the data, with the knowledge that there is a reasonable chance that such a large sample will provide acceptable accuracy.

Extrapolation and interpolation methods work by estimating the learning curve, then selecting the appropriate training set size with which to build a

Table 6.1: Computational complexity required for varying training set sizes

Size	Step	Cumulative
1,000	12,882	12,882
2,000	33,297	46,179
4,000	86,064	132,243
8,000	222,450	354,693
16,000	574,969	929,662
32,000	1,486,125	2,415,787
64,000	3,841,196	6,256,983
128,000	9,928,363	16,185,346
256,000	25,661,897	41,847,243
512,000	66,328,450	108,175,693
1,024,000	171,439,519	279,615,212

model. Gradient of tangent methods work in the opposite way: a model is built, then the curve estimated to determine if the plateau has been reached. This methodology eliminates the possibility of selecting a sample size that is likely to produce a model within an acceptable bound of the full training set, as the accuracy of the full training set is not estimated.

It is therefore important to understand the potential computational cost of using gradient of tangent methods if convergence is not detected. Table 6.1 compares the computational cost of learning from the full training set to the cost of learning using the geometrically progressing sampling schedule of Chapter 4. Complexity is calculated with the measurement given by Provost, Jensen and Oates [84], where C4.5 is shown to be O($n^{1.37}$) for the Waveform data set. The column labelled “Step” presents the cost of inducing a single model with the corresponding training set size, and the “Cumulative” column sums costs up to that size.

The table shows an interesting phenomenon. Assume that the full training set consists of 1,024,000 instances. Then for a geometric sampling progression to be less expensive than learning from the full training set, the maximum sample size used with this sampling schedule must be 512,000 instances as this is the largest cumulative size less than the cost of learning from the full training set. However, this is true only for methods that detect convergence with global analysis of the gradient. Methods that employ local analysis must build many models of similar size. LRLS selects 10 additional samples around the scheduled sample size. Therefore, the cost at each step, and hence the cumulative cost, must be multiplied by 11 (the scheduled size plus 10 additional samples) to obtain the true cost for LRLS. Thus, to be more efficient than learning from the full training set, LRLS must detect convergence at a maximum of 64,000 instances. Even if the number of additional samples is reduced to four, the greatest sample size that can be used while maintaining efficiency increases to only 128,000 instances. Using only one additional sample still requires convergence to be detected with a sample of at most 256,000 instances, however, a limit of only one additional sample would also preclude the use of linear regression to find the local gradient.

Thus, to be more efficient than learning from the full training set, the LRLS algorithm must detect convergence with at most one-sixteenth of the full training set. Previous research [71], as well as the results reported in Chapter 4, show this is unlikely. Given that for five of the eleven data sets used in Chapter 4 pre-sampling could not reach accuracy within one per cent of that obtainable with the full training set, the utility of such a method of detecting convergence is questionable. It must also be remembered that Table 6.1 does not include the cost of measuring accuracy or detecting convergence, but only of inferring models. Including such costs will further reduce efficiency.

A general formula for calculating the maximum number of samples that can be selected using a geometric progression while ensuring greater efficiency than using the full training set can be derived as follows.

Let $\mathbf{n} = n_0, n_1, ..., n_N$ be any sampling schedule such that $n_N < n_{full}$, where n_{full} is the number of instances in the full training set. n_N is then the maximum sample size of the schedule that is less than the full training set. Assuming computational complexity can be represented as an exponential function, an efficient schedule requires

$$\sum_{i=0}^{N} A(n_i^x) < n_{full}^x \tag{6.1}$$

where x is the computational complexity of learning from a data set, and A is the number of models built at each scheduled sample size. For global analysis algorithms, A will be 1 as no additional points are sampled. For local analysis algorithms such as LRLS, A will be at least 3, so that linear regression calculating the gradient of the tangent can be performed. Note that although the size of the additional sample points is not precisely the same as the scheduled point, the difference in computational complexity is negligible.

For a geometric sampling schedule, n_i can be defined as

$$n_i = Br^i \tag{6.2}$$

where B is the initial sample size and r is the rate of increase of the sampling schedule. Then,

$$\sum_{i=0}^{N} A(Br^i)^x < n_{full}^x \tag{6.3}$$

The left hand side of the inequality can be substituted with the partial sum of a geometric series, giving

$$\frac{B^x[r^{x(N+1)} - 1]}{r^x - 1} < n_{full}^x A^{-1} \tag{6.4}$$

Solving for N results in the inequality

$$N < \frac{1}{x} \log_r(1 + \frac{r^x - 1}{AB^x} n_{full}^x) - 1 \tag{6.5}$$

This provides an upper bound on the number of geometrically progressing sample sizes that can be used while ensuring greater efficiency than learning from the full training set. The maximum sample size can be calculated by substituting $i = N$ into Equation 6.2.

It must be noted that Inequality 6.5 applies only to geometrically progressing sampling schedules. Geometric progressions have been shown to be more efficient than both linear progressions and dynamically programmed schedules [84], however, better estimates of the likelihood of detecting convergence may reduce or reverse the difference with dynamic programming.

Figures 6.1 – 6.4 plot the maximum number of samples allowable while remaining more efficient than learning from the full training set for varying computational complexity, with differing numbers of samples taken at each scheduled point. Each graph represents a different rate of increase of the schedule. Referring to Inequality 6.5, N is represented on the vertical axis, and x on the horizontal. Rate of increase r differs with each graph, and the six curves on each graph relate to differing values of A. B and n_{full} are set to the constants 1,000 and 1,024,000 respectively. The graphs are limited to computational complexity exponents between 1.0 and 3.0. Algorithms with smaller exponents than 1.0 are likely to result in poor predictive classifiers, as extremely little attention can be paid to the data. Algorithms with larger exponents then 3.0 are likely to be prohibitively expensive and therefore rarely used.

The figures show that as computational complexity decreases, the maximum number of samples that can be used while being more efficient than learning from the full training set also decreases. Thus, rather than allowing more samples to be taken as might be expected, using more efficient algorithms to estimate learning curves actually demands that fewer samples are scheduled. This result also impacts on methods based on curve extrapolation by requiring that either fewer samples are used, therefore reducing the reliability of the estimated curve,

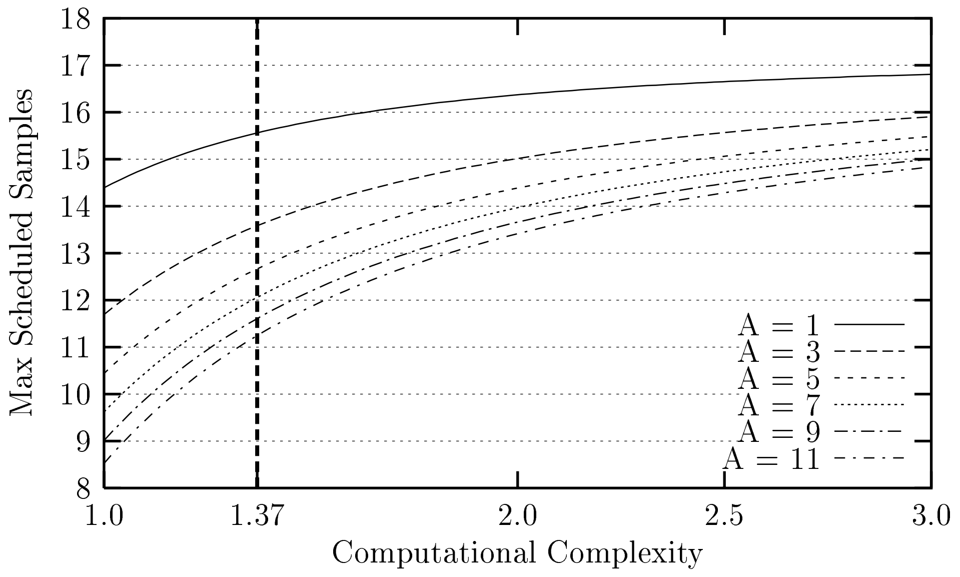

Figure 6.1: Maximum number of samples for varying computational complexity, with rate of increase $r = 1.5$

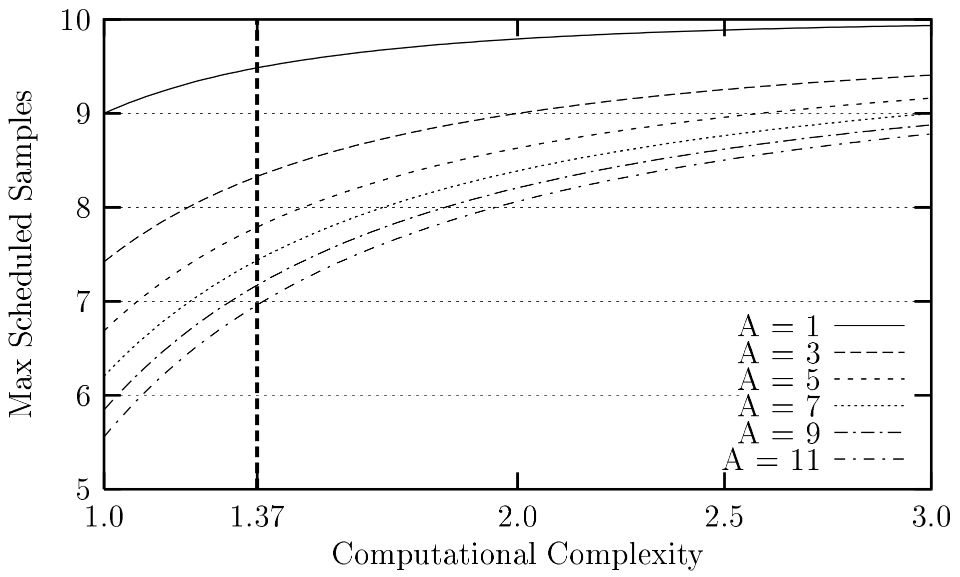

Figure 6.2: Maximum number of samples for varying computational complexity, with rate of increase $r = 2.0$

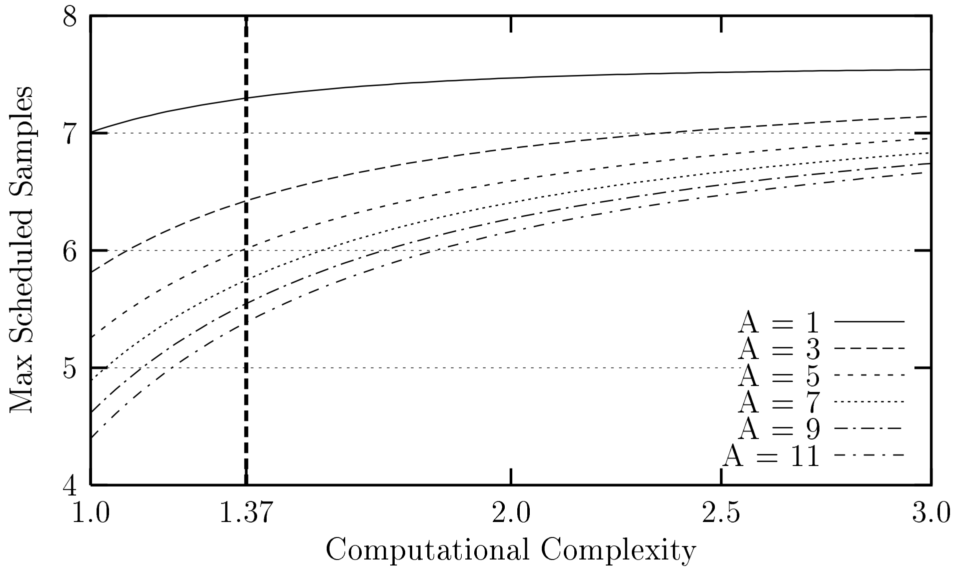

Figure 6.3: Maximum number of samples for varying computational complexity, with rate of increase $r = 2.5$

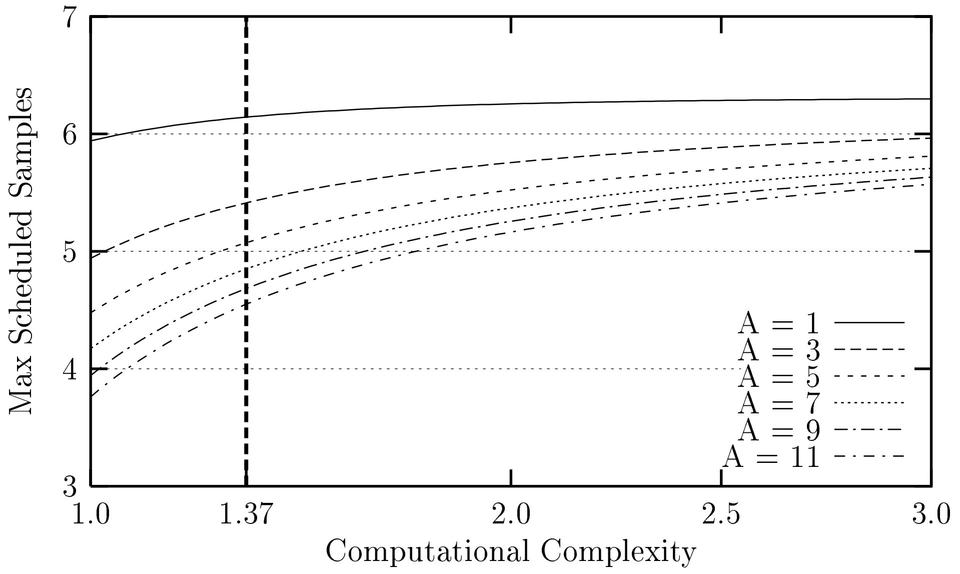

Figure 6.4: Maximum number of samples for varying computational complexity, with rate of increase $r = 3.0$

or increasing the risk of having greater computational cost than learning from the full training set. This result should not be taken as suggesting increasing learning algorithm efficiency will result in making learning curve estimation less efficient, but that if an efficient algorithm is to be used, it may be more sensible to use the full training set, rather than taking the risk of increasing execution time by using learning curves and not detecting convergence early enough.

The range on the horizontal axis precludes the graphs showing the existence of an asymptote of the number of samples with increasing computational complexity. These limits are 17.095, 10.000, 7.565, and 6.309 samples for $r = 1.5$, 2.0, 2.5, and 3.0 respectively. These limits show that as the rate of increase of the geometric progression increases, fewer samples can be used. This is not unexpected, as increasing the rate of increase necessarily means that fewer samples will be scheduled if the size of the full training set and the initial sample size remain fixed. However, the difference between these limits is somewhat surprising. Being able to use a maximum of 17 scheduled points for $r = 1.5$ may allow sufficient observed accuracies to accurately estimate the learning curve. Increasing the rate of increase to $r = 2.0$ almost halves the number of samples that may be scheduled. This could have a dramatic effect on the accuracy of curve estimation.

6.3 Effectiveness of Convergence Detection of Local Gradient of Tangent Methods

Although computational complexity and execution time are problems of importance when learning curves are employed, it is possibly even more important that an acceptably accurate model is built. Methods of detecting convergence that analyse local gradients introduce the risk that local accuracy plateaus may be found if the curve is not strictly monotonic. This may result in convergence being detected with an accuracy much smaller than the global maximum.

Evidence of the non-monotonicity of learning curves already exists. Previous research has shown learning curves that are not monotonic [78], occasions where accuracy decreases with increasing sample size [26], and learning curves that clearly contain local maxima [80, 29]. Figure 4.7 also showed accuracy to decrease on the IPUMS data set when moving from a sample of 8,000 to 16,000 instances.

Although these examples show only local accuracy maxima and not local accuracy plateaus, they may have the same effect. At a maximum, the tangent to the curve is zero. Local maxima, as well as local plateaus, may therefore result in detection of convergence.

6.3.1 Experiments

Rather than knowing it is possible for local plateaus to occur, it is more important to know whether they affect convergence detection when using local gradient analysis. Experiments were performed to investigate whether analysis of the gradient of the tangent can be expected to detect convergence only when further accuracy increase is negligible.

Methodology

Experiments employed the same geometric sampling schedule as used in previous chapters. Instances for a sample were randomly selected without regard to class distribution, and without replacement. Additional points were then sampled around the scheduled point. As the way in which LRLS selects these points is not clear, it is assumed that the same number of points are selected on each side of the scheduled sample size, with a difference of one instance between each sample size. Following LRLS, linear regression was performed on these points, with convergence considered detected if the 95% confidence interval of the

Table 6.2: Minimum sample size at which convergence was detected

	No. of Additional Samples				
Data Set	2	4	6	8	10
Adult	1,000	1,000	2,000	2,000	2,000
Census Income	1,000	1,000	1,000	1,000	1,000
Connect-4	1,000	2,000	1,000	2,000	2,000
Cover Type	1,000	1,000	1,000	1,000	2,000
Distinct Boundary	1,000	1,000	1,000	1,000	1,000
Fuzzy Boundary	1,000	1,000	1,000	1,000	1,000
IPUMS	1,000	1,000	1,000	1,000	1,000
Random Binary	1,000	2,000	2,000	2,000	4,000
Shuttle	1,000	1,000	4,000	2,000	1,000
Sleep	1,000	1,000	1,000	1,000	1,000
Waveform	1,000	1,000	1,000	4,000	4,000

gradient of the regressed line contained zero. To mitigate the effect of variance of measured accuracy due to the specific samples used, accuracy was averaged over ten runs of three-fold cross-validation. The eleven data sets used in previous chapters were employed for these experiments.

6.3.2 Results

Table 6.2 shows the minimum sample size at which convergence was detected on each data set for differing numbers of additional samples.

The table shows that with only two additional samples, convergence is detected at a sample of 1,000 instances for each data set. Increasing the number of additional samples delays convergence detection for some data sets. However, even with ten additional samples, convergence is still detected with a sample

Table 6.3: Accuracy of the scheduled sample size when convergence was detected

	Full	No. of Additional Samples				
Data Set	Train Set	2	4	6	8	10
Adult	0.8602	0.8365	0.8365	0.8425	0.8425	0.8425
Census Income	0.9526	0.9386	0.9386	0.9386	0.9386	0.9386
Connect-4	0.7981	0.6761	0.6945	0.6761	0.6945	0.6945
Cover Type	0.9375	0.6623	0.6623	0.6623	0.6623	0.6870
Distinct Boundary	0.9990	0.9688	0.9688	0.9688	0.9688	0.9688
Fuzzy Boundary	0.9985	0.9942	0.9942	0.9942	0.9942	0.9942
IPUMS	0.9284	0.9209	0.9209	0.9209	0.9209	0.9209
Random Binary	1.0000	0.5047	0.5095	0.5095	0.5095	0.5184
Shuttle	0.9996	0.9946	0.9946	0.9969	0.9957	0.9946
Sleep	0.7299	0.6553	0.6553	0.6553	0.6553	0.6553
Waveform	0.9857	0.9653	0.9653	0.9653	0.9753	0.9753

of 1,000 instances on six of the eleven data sets. This suggests that such a convergence detection method is prone to detect convergence early in the sampling schedule. This is unsurprising, especially with few additional samples, as small differences in training set size are unlikely to make significant differences in predictive accuracy.

Table 6.3 shows the accuracy of the scheduled sample size at which convergence was detected for varying numbers of additional samples. The accuracy obtained when using the full training set is also shown.

All training sets with all numbers of additional samples show a loss of accuracy compared to the full training set, with the most dramatic losses on the Cover Type and Random Binary data sets. The Cover Type data set loses approximately 30% of the accuracy obtainable from the full training set, while the loss is close to 50% with Random Binary. The Sleep data set also loses

Table 6.4: Number of data sets within accuracy loss bounds

Accuracy	Additional Samples				
Loss Bound	2	4	6	8	10
1%	3	3	3	3	3
2%	4	4	4	5	5
5%	7	7	7	7	7

approximately 10%.

Table 6.4 shows the number of data sets that were within given accuracy loss bounds at the sample size where convergence was detected. As can be seen, only three data sets obtained accuracy within one per cent of that obtainable with the full training set, regardless of the number of additional samples. These were the Fuzzy Boundary, IPUMS, and Shuttle data sets. Perhaps more importantly, with each number of additional samples, four data sets were not within five per cent accuracy bounds. This suggests that using this method to detect convergence may result in substantial accuracy loss, as convergence is prematurely detected.

The results call into question previous results concerning LRLS [84], where the start of the global curve plateau was considered to be the size at which accuracy was within one per cent of that of the full training set. These results show few of the data sets used converged within this bound. However, it is claimed that the LRLS local gradient of tangent method is effective in determining convergence with accuracy close to that obtainable with the full training set [84]. The above table provides evidence contrary to that conclusion.

6.4 Conclusions

Conditions under which learning curve methodologies will be more efficient than learning from the full training set were investigated. A formula for determining the maximum number of sample points for which learning curve methods will be more efficient than learning from the full training set has been introduced in Section 6.2 for a geometric sampling schedule. Graphs show that as algorithm efficiency is increased, the number of samples that can be used while maintaining efficiency decreases. This suggests that if highly efficient algorithms are to be employed, using learning curves may be a risky proposition, as relatively few sample sizes can be used in curve estimation. Potential effects of this could be poor estimation of the learning curve or strict bounds on the number of samples that can be explored before convergence must detected to ensure efficiency.

Experiments were performed to investigate the effectiveness of methods that determine convergence by analysing the local gradient of the tangent to the curve. The results show that this method often detects convergence when local accuracy is far from that obtainable with the full training set. Accuracy of the sample size at which convergence was detected was within one per cent of that obtainable with the full training set for only three of the eleven data sets, suggesting that this method is not reliable in detecting the global accuracy plateau. This may be due to the apparent frequency of local accuracy plateaus or local accuracy maxima.

Chapter 7

Conclusions

This dissertation has investigated issues relating to the use of classification learning algorithms with large data sets, and methods to overcome problems which have been identified. A bias/variance framework has been proposed for analysing the performance of learning algorithms at differing data set sizes. Existing methods have been compared and new methods proposed, developed, and evaluated.

Section 7.1 summarises the work detailed in each chapter. Section 7.2 discusses directions for future research. Software developed to facilitate the experiments of this dissertation are outlined in Section 7.3. Section 7.4 re-iterates the main conclusions presented in previous chapters.

7.1 Summary

The dissertation investigated a number of aspects relating to learning from large data.

Chapter 3 described experiments investigating whether different types of algorithm are required with large data sets than with small data sets. An analysis of the bias plus variance decomposition of error was performed. This focused on how the profile of predictive error changes as training set size increases, leading to the conclusion that bias reduction is most important at large data set sizes. The bias reduction potential of decision tree grafting was therefore investigated, as was the potential of increasing representational power. These results may provide useful insights with regard to the development of future algorithms specifically designed for use with large data sets.

Chapter 4 provided a rigorous comparison of the traditional notions of sampling, here termed pre-sampling and sub-sampling. Comparisons were made in terms of predictive accuracy, model complexity, and execution time. The way in which these vary as training set size increases was also analysed. This research may aid practitioners in making appropriate choices when faced with large data.

Chapter 5 investigated ways of improving sub-sampling. The effects of disproportionate sampling and replacement on inferred models were analysed. Two new methods of selecting instances were also proposed. Variable Proportion Sampling selects an increasing percentage of available instances as tree depth increases. SQ Sampling dynamically determines sample size at a node using a measure of sample quality. These methods were compared to sub-sampling as used in Chapter 4. SQ Sampling was shown to infer models from the largest data sets used without loss of accuracy, but with a substantial saving of time. It may thus provide a viable means of using sub-sampling with large data sets.

Chapter 6 analysed the feasibility of using learning curves to determine optimal sample size. The analysis consisted of two parts. The first part investigated limitations on the sampling scheme to ensure a reduction in execution time as compared to learning from the full training set. The second part investigated the reliability of methods that determine convergence of the learning curve by calculating the gradient of the tangent to the curve, and their susceptibility to local accuracy plateaus. The results call into question the general applicability of learning curve estimation methods, and the reliability of gradient of tangent estimation to produce models without substantial loss of accuracy.

7.2 Further Research

There is wide scope for future research following from insights gained in this thesis.

Chapter 3 showed that bias becomes an increasingly dominant factor of error as training set size increases. Learning algorithms that focus on bias management rather than variance management could therefore have the potential to perform exceptionally well with large data sets. One way in which increased bias management may be achieved is to select the best of a number of algorithms, similar to stacking. However, stacking will increase execution time drastically. An alternative may be to employ a hybrid of stacking and pasting [11], such that models are built using a number of algorithms on small training sets, with the best model at each step kept for inclusion in the committee. The process can then be repeated, allowing for the possibility of a committee of a number of different types of models. Although this would likely lead to an increase in execution time, using small training sets would mitigate any increase.

Given the high accuracy of sub-sampling, more efficient sub-sampling methodologies may prove useful. Reducing the execution time of sub-sampling could increase the viability of using sub-sampling with large data sets.

A pasting-like approach could be employed at each node of a decision tree. Numerous small samples of available data could be selected, with the best split found for each sample. The attribute that wins a majority vote could then be selected as the attribute on which to split at that node. If the attribute is discrete, no further search must be performed. If the attribute is continuous, then either the split with the highest utility could be used, or a search performed on the selected attribute only to find the best potential split. This may reduce execution time while having the advantage of building a single model.

Further research into different methods of performing sub-sampling could prove fruitful. The results concerning SQ Sampling suggest it is possible to infer models with accuracy close to that obtainable on the full training set while determining sample size at each node. Further research into ways of speeding this up may be beneficial, especially into ways of making the calculation of sufficiency more computationally efficient.

7.3 Software Developed

A number of modifications to C4.5 Release 8 were required to perform this research. These were:

1. Extension to allow pre-sampling.
2. Extension to allow sub-sampling.
3. Implementation of sub-sampling variants to allow samples to be selected with and without replacement and with and without disproportionate sampling.
4. Extension to include Variable Proportion Sampling and SQ sampling.

The OC1 induction algorithm was also extended to collect results regarding bias and variance.

7.4 Conclusions

The principal conclusions of this thesis are:

1. Variance has been shown to decrease and bias to become a larger proportion of error as training set size increases. This can be expected to occur regardless of the bias plus variance profile of the induction algorithm. Bias has been shown to generally decrease as training set size increases, but whether or not this occurs may be dependent on the bias plus variance

profile of the algorithm. The increasing proportion of bias error is shown to be due to the natural trend for variance to decrease with larger training sets. Together, these results suggest that bias management becomes more important with larger training sets.

2. The cause of bias and variance has been analysed. Two causes of bias — erroneous extrapolation of a concept and lack of representational power — have been identified as areas in which algorithms may deliberately target bias reduction.

3. Decision tree grafting becomes less effective as a bias reduction method as training set size increases, although it remains useful for reducing variance.

4. An increase in representational power will reduce the bias component of error when learning from large training sets, but does not necessarily result in an overall decrease in error. The behaviour of bias reduction mechanisms has also been shown to alter with increased training set size.

5. Experimental comparisons of popular sampling methodologies showed sub-sampling to produce substantially more accurate and more descriptive models than pre-sampling when using the same sample size. Sub-sampling was found to produce models with greater predictive accuracy than those obtainable without sampling using the full training set, without decreasing model complexity. Sub-sampling allows a practitioner to have high confidence that an acceptably accurate model will be produced, with a likely saving of execution time over learning from the full training set. Pre-sampling almost guarantees a saving of execution time, but does so at a high risk of substantially reduced accuracy.

6. An investigation into the effects of selecting disproportionate sub-samples and selecting sub-samples without replacement showed disproportion-

ate sampling to have a stronger influence on model accuracy than non-replacement. Simplifying the sampling process by using replacement more frequently resulted in a decrease in execution time than did removing disproportionate sampling.

7. Determining sub-sample size by selecting samples of variable proportions was shown to produce models with accuracy comparable to that of no sampling and with a saving of execution time.

8. Sample Quality (SQ) sampling showed determining sub-sample size by using a statistical measure of sample quality to produce models with accuracy very close to that obtainable without sampling. On the largest data sets used, SQ sampling substantially decreased execution time without loss of accuracy.

9. The evaulation of the statistical quality of a sample was extended to include continuous attributes.

10. The viability of learning curves was investigated and a formula derived to determine the maximum number of sample points that can be used while ensuring a reduction of execution time over learning from the full training set. An analysis showed that the more computationally efficient an algorithm is, the fewer samples it can use to estimate a learning curve while maintaining reduced execution time.

11. The reliability of methods that determine learning curve convergence through analysis of the gradient of the tangent to the curve was investigated. Experiments showed that, due to the non-monotonicity of the curves, such methods often detect convergence before the global accuracy maximum is reached by finding local accuracy plateaus. Combined with the potential inefficiency of learning curve methods, this result suggests significant risk in the use of learning curves for estimating optimal sample size.

Classification learning from large data sets is likely to be a continual problem for machine learning and data mining practitioners. It can be expected that data set sizes and computing power will continue to increase, but it is unlikely that the size of main memory or the speed of processors will advance sufficiently quickly to reduce the strain increased data set size will put on computing resources. This research has investigated ways in which the problem of large data can be mitigated. The comparison of sampling methodologies provides practitioners with insight into how sampling can best be employed to suit their needs, and the value and cost of replacement and disproportionate sampling. The analysis into the efficiency and reliability of learning curve methods has shown surprising results concerning the number of samples that can be used to estimate a learning curve, and that methods that detect convergence by analysing the gradient of the tangent to the curve can result be substantially less accurate models than previous work might suggest. Finally, the future design of algorithms may be aided by evidence that different types of algorithm to those commonly used may be required when learning from large data sets. As data sets continue to grow, it may become increasingly important to use the right sort of algorithm with large data sets.

Bibliography

[1] Khaled Alsabti, Sanjay Ranka, and Vineet Singh. CLOUDS: A decision tree classifier for large datasets. In *Proceedings of the Fourth ACM SID-KDD International Conference on Knowledge Discovery and Data Mining*, pages 2–8, New York, NY, August 1998. ACM Press.

[2] Yali Amit and Donald Geman. Shape quantization and recognition with randomized trees. *Neural Computation*, 9(7):1545–1588, October 1997.

[3] John M. Aronis and Foster J. Provost. Increasing the efficiency of data mining algorithms with breadth-first marker propagation. In David Heckerman, Heikki Mannila, Daryl Pregibon, and Ramasamy Uthurusamy, editors, *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining*, pages 119–122, Newport Beach, CA, August 1997. AAAI Press.

[4] Eric Bauer and Ron Kohavi. An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. *Machine Learning*, 36(1/2):105–139, July/August 1999.

[5] Stephen D. Bay. Multivariate disretization for set mining. *Knowledge and Information Systems*, 3(4):491–512, November 2001.

[6] Stephen D. Bay and Michael J. Pazzani. Detecting change in categorical data: Mining constrast sets. In *Proceedings of the Fifth ACM SIGKDD*

International Conference on Knowledge Discovery and Data Mining, pages 302–306, San Diego, CA, August 1999. ACM Press.

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

[8] Paul S. Bradley, Usama M. Fayyad, and Cory Reina. Scaling clustering algorithms to large databases. In *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining*, pages 9–15, New York, NY, August 1998. AAAI Press.

[9] Leo Breiman. Arcing classifiers. Technical Report 460, Statistics Department, University of California, Berkeley, 1996.

[10] Leo Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, August 1996.

[11] Leo Breiman. Pasting small votes for classification in large databases an on-line. *Machine Learning*, 36(1/2):85–103, July 1999.

[12] Leo Breiman. Random forests - random features. Technical Report 567, Statistics Department, Univeristy of California, Berkeley, CA, 1999.

[13] Leo Breiman, Jerome H. Friedman, Richard A. Olshen, and Charles J. Stone. *Classification and Regression Trees.* Wadsworth International, Monterey, CA, 1984.

[14] Tom Brijs and Koen Vanhoof. Cost-sensitive discretization of numeric attributes. In *Proceedings of the Second European Symposium on Principles of Data Mining and Knowledge Discovery*, pages 102–110, Nantes, France, September 1998. Springer-Verlag.

[15] Jason Catlett. *Megainduction: Machine Learning on Very Large Databases.* PhD thesis, University of Sydney, 1991.

[16] Jason Catlett. Peepholing: Choosing attributes efficiently for megainduction. In *Proceedings of the Ninth International Conference on Machine Learning*, pages 49–54, Aberdeen, Scotland, July 1992. Morgan Kaufmann.

[17] Philip K. Chan and Salvatore J. Stolfo. Learning arbiter and combiner trees from partitioned data for scaling machine learning. In *Proceedings of the First International Conference on Knowledge Discovery and Data Mining*, pages 39–44, Montreal, Canada, August 1995. AAAI Press.

[18] Philip K. Chan and Salvatore J. Stolfo. Toward scalable learning with non-uniform class and cost distributions: A case study in credit card fraud detection. In *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining*, pages 164–168, New York, NY, August 1998. AAAI Press.

[19] Jaturon Chattratichat, John Darlington, Moustafa Ghanem, Yike Guo, Harald Hüning, Martin Köhler, Janjao Sutiwaraphun, Hing Wing To, and Dan Yang. Large scale data mining: Challenges and responses. In *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining*, pages 143–146, Newport Beach, CA, August 1997. AAAI Press.

[20] Nitesh V. Chawla, Lawrence O. Hall, Kevin W. Bowyer, Jr. Thomas E. Moore, and W. Philip Kegelmeyer. Distributed pasting of small votes. In *Proceedings of the Third International Workshop on Multiple Classifier Systems*, pages 52–61, Cagliari, Italy, June 2002. Springer.

[21] Peter Clark and Tim Niblett. The CN2 induction algorithm. *Machine Learning*, 3(4):261–283, March 1989.

[22] W. J. Conover. *Practical Nonparametric Statistics.* John Wiley & Sons, New York, second edition, 1980.

[23] Corinna Cortes and Daryl Pregibon. Giga-mining. In *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining*, pages 174–178, New York, NY, August 1998. AAAI Press.

[24] Thomas M. Cover and Peter E. Hart. Nearest neighbor pattern classification. *IEEE Transactions of Information Theory*, 13(1):21–27, January 1967.

[25] John Darlington, Moustafa M. Ghanem, Yike Guo, and Hing Wing To. Performance models for co-ordinating parallel data classification. In *Proceedings of the Seventh International Parallel Computing Workshop*, Canberra, Australia, September 1997.

[26] Thierry Van de Merckt. Decision trees in numerical attribute spaces. In *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence*, pages 1016–1021, Chambery, France, August 1993. Morgan Kaufmann.

[27] Thomas G. Dietterich. An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting and randomization. *Machine Learning*, 40(2):139–157, August 2000.

[28] Carlos Domingo, Ricard Gavaldá, and Osamu Watanabe. Adaptive sampling methods for scaling up knowledge discovery algorithms. In *Proceedings of the Second International Conference on Discovery Science*, pages 172–183. Springer-Verlag, December 1999.

[29] Pedro Domingos. Linear-time rule induction. In Evangelos Simoudis, Jia Wei Han, and Usama Fayyad, editors, *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, pages 96–101, Portland, OR, August 1996. AAAI Press.

[30] Pedro Domingos. Using partitioning to speed up specific-to-general rule induction. In *Proceedings of the AAAI-96 Workshop on Integrating Multiple Learned Models*, pages 29–34, Portland, OR, 1996. AAAI Press.

[31] Pedro Domingos and Geoff Hulten. Mining high-speed data streams. In *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 71–80, Boston, MA, August 2000. ACM Press.

[32] Pedro Domingos and Geoff Hulten. A general method for scaling up machine learning algorithms and its application to clustering. In *Proceedings of the Eighteenth International Conference on Machine Learning*, pages 106–113, Williamstown, Maryland, June 2001. Morgan Kaufmann.

[33] James Dougherty, Ron Kohavi, and Mehran Sahami. Supervised and unsupervised discretization of continous features. In *Proceedings of the Twelfth International Conference on Machine Learning*, pages 114–119, Tahoe City, CA, July 1995. Morgan Kaufmann.

[34] William DuMouchel, Chris Volinskyand Theodore Johnson, Corinna Cortes, and Daryl Pregibon. Squashing flat files flatter. In *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 6–15, San Diego, CA, August 1999. ACM Press.

[35] Andrzej Ehrenfeucht, David Haussler, Michael Kearns, and Leslie Valiant. A general lower bound on the number of examples needed for learning. *Information Computing*, 82(3):247–261, 1989.

[36] Martin Ester, Hans-Peter Kriegel, and Xiaowei Xu. A database interface for clustering in large spatial databases. In *Proceedings of the First In-*

ternational Conference on Knowledge Discovery and Data Mining, pages 94–99, Montreal, Canada, Montreal, Canada 1995. AAAI Press.

[37] Usama Fayyad and Padhraic Smyth. From massive data sets to science catalogs: Applications and challenges. *Statistics and Massive Data Sets, Report to the Committee on Applied and Theoretical Statistics*, 1996.

[38] Usama M. Fayyad and Keki B. Irani. Multi-interval discretization of continuous-valued attributes for classification learning. In *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence*, pages 1022–1027, Chambery, France, August 1993. Morgan Kaufmann.

[39] Eibe Frank and Ian H. Witten. Making better use of global discretization. In *Proceedings of the Sixteenth International Conference on Machine Learning*, pages 115–123, Bled, Slovenia, June 1999. Morgan Kaufmann.

[40] Yoav Freund and Robert E. Schapire. Experiments with a new boosting algorithm. In *Proceedings of the Thirteenth International Conference on Machine Learning*, pages 148–156, Bari, Italy, July 1996. Morgan Kaufmann.

[41] Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of online learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, August 1997.

[42] Lewis J. Frey and Douglas H. Fisher, Jr. Modeling decision tree performance with the power law. In *Proceedings of the Seventh International Workshop on Artificial Intelligence and Statistics*, pages 59–65, Fort Lauderdale, FL, January 1999. Morgan Kaufmann.

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

[44] Jerome H. Friedman, Ron Kohavi, and Yeogirl Yun. Lazy decision trees. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, pages 717–724, Portland, Oregon, August 1996. AAAI Press.

[45] Johannes Fürnkranz. Integrative windowing. *Journal of Artificial Intelligence Research*, 8:129–164, January 1998.

[46] Johannes Gehrke, Raghu Ramakrishnan, and Venkatesh Ganti. Rainforest - a framework for fast decision tree construction of large datasets. In *Proceedings of the Twenty-fourth International Conference on Very Large Databases*, pages 416–427, New York, NY, August 1998. Morgan Kaufmann.

[47] David Gibson, Jon M. Kleinberg, and Prabhakar Raghavan. Clustering categorical data: An approach based on dynamical systems. *The VLDB Journal*, 8(3–4):222–236, February 2000.

[48] Jonathan Gratch. Sequential inductive learning. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, pages 779–786, Portland, OR, August 1996. AAAI Press.

[49] Baohua Gu, Feifang Hu, and Huan Liu. Modelling classification performance for large data sets. In *Proceedings of the Second International Conference on Advances in Web-Age Information Management*, pages 317–328, Xi'an, China, July 2001. Springer-Verlag.

[50] Baohua Gu, Bing Liu, Feifang Hu, and Liu Huan. Efficiently determining the starting sample size for progressive sampling. In *Proceedings of the Twelfth European Conference on Machine Learning*, pages 192–202, Freiburg, Germany, September 2001. Springer-Verlag.

[51] Pierre Guerts. Some enhancements of decision tree bagging. In *Proceedings of the Fourth European Conference on Principles of Data Mining*

and Knowledge Discovery, pages 136–147, Lyon, France, September 2000. Springer.

[52] Sudipto Guha, Rajeev Rastogi, and Kyuseok Shim. ROCK: A robust clustering algorithm for categorical attributes. *Information Systems*, 25(5):345–366, 2000.

[53] David Haussler, Michael Kearns, H. Sebastian Seung, and Naftali Tishby. Rigorous learning curve bounds from statistical mechanics. *Machine Learning*, 25(2-3):195–236, 1996.

[54] Tin Kam Ho. The random subspace method for constructing decision forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(8):832–844, August 1998.

[55] Chun-Nan Hsu, Hung-Ju Huang, and Tzu-Tsung Wong. Why discretization works for naive bayesian classifiers. In *Proceedings of the Seventeenth International Conference on Machine Learning*, pages 399–406, Austin, TX, June 2000. Morgan Kaufmann.

[56] M. E. C. Hull, Danny Crookes, and P. J. Sweeney. *Parallel Processing: The Transputer and Its Applications.* Addison-Wesley, Reading, Maryland, 1994.

[57] Geoff Hulten, Laurie Spencer, and Pedro Domingos. Mining time-changing data streams. In *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 97–106, San Francisco, CA, August 2001. ACM Press.

[58] Gareth James and Trevor Hastie. Generalizations of the bias/variance decomposition for prediction error. Technical report, Department of Statistics, Stanford University, February 1997.

[59] Gareth M. James. Variance and bias for general loss functions. *Machine Learning*, 51(2):115–135, May 2003.

[60] George H. John and Pat Langley. Static versus dynamic sampling for data mining. In Evangelos Simoudis, Jiawei Han, and Usama M. Fayyad, editors, *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, pages 367–370, Portland, Oregon, U.S.A., August 1996. AAAI Press.

[61] Dimitrios Kalles and Athanassios Papagelis. Stable decision trees: Using local anarchy for efficient incremental learning. *International Journal on Artificial Intelligence Tools*, 9(1):79–95, 2000.

[62] Hillol Kargupta, Ilker Hamzaoglu, and Brian Stafford. Scalable, distributed data mining-an agent architecture. In David Heckerman, Heikki Mannila, Daryl Pregibon, and Ramasamy Uthurusamy, editors, *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining*, pages 211–214, Newport Beach, CA, August 1997. AAAI Press.

[63] Kenji Kira and Larry A. Rendell. A practical approach to feature selection. In *Proceedings of the Ninth International Conference on Machine Learning*, pages 249–256, Aberdeen, Scotland, July 1992. Morgan Kaufmann.

[64] Ron Kohavi. Scaling up the accuracy of Naive-Bayes classifiers: a decision-tree hybrid. In Evangelos Simoudis, Jia Wei Han, and Usama Fayyad, editors, *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, pages 202–207, Portland, OR, August 1996. AAAI Press.

[65] Ron Kohavi and Mehran Sahami. Error-based and entropy-based discretization of continous features. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, pages 114–119, Portland, OR, August 1996. AAAI Press.

[66] Ron Kohavi and David H. Wolpert. Bias plus variance decomposition for zero-one loss functions. In Lorenza Saitta, editor, *Proceedings of the Thirteenth International Conference on Machine Learning*, pages 275–283, Bari, Italy, July 1996. Morgan Kaufmann.

[67] Eun Bae Kong and Thomas G. Dietterich. Error-correcting output coding corrects bias and variance. In *Proceedings of the Twelfth International Conference on Machine Learning*, pages 313–321, Tahoe City, CA, July 1995. Morgan Kaufmann.

[68] Solomon Kullback. *Information Theory and Statistics.* Dover Publications, Mineola, NY, second edition, 1997.

[69] Pat Langley. *Elements of Machine Learning.* Morgan Kaufmann, 1996.

[70] Pat Langley, Wayne Iba, and K. Thompson. An analysis of bayesian classifiers. In *Proceedings of the Tenth National Conference on Artificial Intelligence*, pages 223–228, San Jose, CA, July 1992. AAAI Press.

[71] Aleksandar Lazarevic and Zoran Obradovic. Data reduction using multiple models integration. In *Proceedings of the Fifth European Conference on Principles of Data Mining and Knowledge Discovery*, pages 301–313, Freiburg, Germany, September 2001. Springer-Verlag.

[72] Huan Liu and Hiroshi Motoda. *Feature Selection for Knowledge Discovery and Data Mining.* Kluwer Academic Publishers, 1998.

[73] Shie Mannor and Ron Meir. On the existence of linear weak learners and applications to boosting. *Machine Learning*, 48(1–3), July/August 2002.

[74] M. C. Monard and G. E. A. P. A. Batista. Learning with skewed class distribution. In *Advances in Logic, Artificial Intelligence and Robotics*, pages 173–180, São Paulo, Spain, 2002.

[75] A. Moore and M. S. Lee. Cached sufficient statistics for efficient machine learning with large datasets. *Journal of Artificial Intelligence Research*, 8:67–91, January 1998.

[76] David S. Moore and George P. McCabe. *Introduction to the Practice of Statistics*. W. H. Freeman and Company, New York, New York, third edition, 1999.

[77] Ron Musick, Jason Catlett, and Stuart Russell. Decision theoretic subsampling for induction on large databases. In *Proceedings of the Tenth International Conference on Machine Learning*, pages 212–219, Amherst, MA., June 1993. Morgan Kaufmann.

[78] Vítor H. Nascimento and Ali H. Sayed. Are ensemble-average learning curves reliable in evaluating the performance of adaptive filters? In *Proceedings of the Asilomar Conference on Signals, Systems, and Computers*, pages 1171–1175, Pacific Grove, CA, October 1998.

[79] Raymond T. Ng and Jiawei Han. Efficient and effective clustering methods for spatial data mining. In *Proceedings of the Twentieth International Conference on Very Large Databases*, pages 144–155, Santiago, Chile, September 1994. Morgan Kaufmann.

[80] Tim Oates and David Jensen. The effects of training set size on decision tree complexity. In *Proceedings of the Fourteenth International Conference on Machine Learning*, pages 254–262, Nashville, TN, July 1997. Morgan Kaufmann.

[81] Tim Oates and David Jensen. Large datasets lead to overly complex models: an explanation and a solution. In *Proceedings of the Fourth Internataional Conference on Knowledge Discovery and Data Mining*, pages 294–298, New York, NY, August 1998. AAAI Press.

[82] Art B. Owen. Data squashing by empirical likelihood. *Data Mining and Knowledge Discovery*, 7(1):101–113, January 2003.

[83] Claudia Perlich, Foster Provost, and Jeffrey S. Simonoff. Tree induction vs. logistic regression: A learning-curve analysis. *Journal of Machine Learning Research*, To appear.

[84] Foster Provost, David Jensen, and Tim Oates. Efficient progressive sampling. In *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 23–32, San Diego, CA, August 1999. ACM Press.

[85] Foster Provost and Venkateswarlu Kolluri. Scaling up inductive algorithms: An overview. In David Heckerman, Heikki Mannila, Daryl Pregibon, and Ramasamy Uthurusamy, editors, *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining*, pages 239–242, Newport Beach, CA, August 1997. AAAI Press.

[86] Foster John Provost and John M. Aronis. Scaling up inductive learning with massive parallelism. *Machine Learning*, 23(1):33–46, 1996.

[87] J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1(1):81–106, 1986.

[88] J. R. Quinlan. *C4.5: Programs for Machine Learning.* Morgan Kaufmann, San Mateo, CA, 1993.

[89] J. R. Quinlan. Improved use of continous attributes in C4.5. *Journal of Artificial Intelligence Research*, 4:77–90, March 1996.

[90] Salvatore Ruggieri. Efficient C4.5. *IEEE Transactions on Knowledge and Data Engineering*, 14(2):438–444, March/April 2002.

[91] Robert E. Schapire, Yoav Freund, Peter Bartlett, and Wee Sun Lee. Boosting the margin: A new explanation for the effectiveness of voting methods. *The Annals of Statistics*, 26(5):1651–1686, 1998.

[92] Tobias Scheffer and Stefan Wrobel. Incremental maximization of non-instance-averaging utility functions with applications to knowledge discovery problems. In *Proceedings of the Eighteenth International Conference on Machine Learning*, pages 481–488, Williamstown, Maryland, June 2001. Morgan Kaufmann.

[93] John Shafer, Raskesh Agrawal, and Manish Mehta. SPRINT: A scalable parallel classifier for data mining. In *Proceedings of the Twenty-second International Conference on Very Large Databases*, pages 544–555, Mumbai, India, September 1996. Morgan Kaufmann.

[94] K. Murthy Sreerama, Simon Kasif, and Steven Salzberg. A system for induction of oblique decision trees. *Journal of Artificial Intelligence Research*, 2:1–32, August 1994.

[95] Robert Tibshirani. Bias, variance and prediction error for classification rules. Technical report, University of Toronto, November 1996.

[96] Hannu Toivonen. Sampling large databases for association rules. In *Proceedings of the Twenty-second International Conference on Very Large Databases*, pages 134–145, Mumbai, India, September 1996. Morgan Kaufmann.

[97] Ricardo Vilalta, Sheng Ma, and Joseph Hellerstein. Rule induction of computer events. In *Proceedings of the Twelfth International Workshop*

on Distributed Systems: Operations and Management, Nancy, France, October 2001. Springer.

[98] Slobodan Vucetic and Zoran Obradovic. Performance controlled data reduction for knowledge discovery in distributed databases. In *Proceedings of the Fourth Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 29–39, Kyoto, Japan, April 2000. Springer-Verlag.

[99] Osamu Watanabe. Simple sampling techniques for discovery science. Technical report, Dept. of Math. and Comp. Science, Tokyo Institute of Technology, 2000.

[100] Geoffrey I. Webb. Decision tree grafting from the all-tests-but-one partition. In *Proceedings of the Sixteenth International Joint Conference on Machine Learning*, pages 702–707, Stockholm, Sweden, 1999. Morgan Kaufmann.

[101] Geoffrey I. Webb. Multiboosting: A technique for combining boosting and wagging. *Machine Learning*, 40(2):1–38, August 2000.

[102] Geoffrey I. Webb and Damien Brain. Generality is predictive of prediction accuracy. In *Proceedings of the 2002 Pacific Rim Knowledge Acquisition Workshop*, pages 117–130, Tokyo, Japan, August 2002. Japanese Society for Artificial Intelligence.

[103] Geoffrey I. Webb and Paul Conilione. Estimating bias and variance from data. Pre-publication manuscript.

[104] B. Wuthrich. Probabilistic knowledge bases. *IEEE Transactions On Knowledge And Data Engineering*, 7:691–698, October 1995.

[105] Ying Yang and Geoffrey I. Webb. Proportional k-interval discretization for naive-bayes classifiers. In *Proceedings of the Twelfth European Conference*

on Machine Learning, pages 564–575, Freiburg, Germany, September 2001. Springer-Verlag.

[106] Mohammed Javeed Zaki, Ching Tien Ho, and Rakesh Agrawal. Parallel classification for data mining on shared-Memory multiprocessors. In *Proceedings of the Fifteenth International Conference on Data Engineering*, pages 198–205, Sydney, Australia, March 1999. IEEE.

[107] Zijian Zheng and Geoffrey I. Webb. Lazy learning of bayesian rules. *Machine Learning*, 41(1):53–84, October 2000.