Abstract

In the world today, we have witnessed an explosion in the amount of data produced. The deluge of data is clearly impossible to handle by present means. In order to address this problem Machine Learning research has focussed on developing automatic data analysis techniques. Most of the techniques developed so far are computationally intensive even for moderate sized datasets (20,000 examples with 200 features), let alone large datasets (100,000 to 1,000,000) with a large number of features. The undesirable consequence of the task is that the learning time becomes unbearably large. This can be addressed by the new idea of using a user-specified training set size. Previous research in the field has shown that using subsets of the data collection lowers the accuracy of the learner. This work addresses the issue of using ensemble learning techniques to overcome this limitation.

Acknowledgments

Firstly, I would like to thank my advisor, Dr. Rich Maclin, for all his help and guidance that he has given me for more than a year. I would like to express my gratitude to the members of my examination committee, Dr. Carolyn Crouch and Dr. Bruce Peckham. I would also like to thank the CS faculty for providing me support during these two years of my graduate study.

I would like to thank my parents in a special way as I would not be where I am without their help. I would also like to thank my fellow students who were so enduring of me, when they let me use their machines for my work. Finally, I thank all my friends for making my stay in Duluth a memorable one.

Contents

List of Figures

List of Tables

1 Introduction

This chapter begins with an example of the task addressed in this thesis. Section 1.2 is a statement of the thesis. Section 1.3 details the motivation behind choosing this particular topic. Section 1.4 describes the context of research for this work. The chapter concludes with a description of the structure of this thesis.

1.1 Example of the Task Addressed in this Thesis

Consider the following task faced by the credit manager of a bank. The details are hypothetical, but the generic task is representative of the task of any credit manager:

> A bank customer applies for a small loan and fills out a standard form that asks for details such as the balance of their principle savings account, their monthly expenses, and whether they are employed. The customer's request is then processed according to the information contained in this form. To make this decision, the bank has records of customers who had applied for credit over the years, plus information about which applications were succesful. Even if automating the process of using this information is difficult - it would certainly be worth the effort as the database should be able to provide a valuable basis to formulate the future lending policy of the bank.

A set of fictional loan applications is shown in Table 1. The various factors that contribute to the success of the application are called *attributes* (e.g., Balance, Employed). These attributes can either be *discrete* (having a fixed number of values) or *continuous* (having floating-point values). In the table, Account and Employed are examples of discrete attributes. Balance and Monthly Expense are examples of continuous attributes. The decision-making task of the manager can be called *classification.* It is possible to automate this process of classification by creating a classification procedure (or classifier) that will label any given example presented in the format of Table 1 as accept or reject (based on information from the form). For this classifier to replace the loan officer, it should be both reliable and accurate. The cost of running and maintaining the classifier should also be low compared to the expected profit from the transaction.

Machine Learning (ML) is an active field of research in which several methods for knowledge encoding have been developed. Computer programs that can do this by "learning" from examples in a dataset are called *inductive systems*. An inductive system is one that draws inferences based only on the training examples it has seen so far.

The classifier used in this thesis is the Naive Bayes Classifier. This classifier has currently experienced a renaissance in ML. Despite its simplicity, the naive Bayes learning scheme performs well on most classification tasks and is often significantly more accurate than most sophisticated methods.

Let us assume that the classifier has Table 1 as its training data, and it has to classify a new instance. It produces probabilistic estimates, comparing the new instance with all training data, for all the possible output classes. It then predicts the output class as the class with the largest posterior probability. Of course, the resulting prediction, which only has a small

Attributes				Output
Account	Balance	Employed	Monthly Expense	Class
bank	700	yes	200	accept
bank	300	yes	600	reject
none	0	yes	400	reject
other bank	1200	yes	500	accept
none	3000	no	300	accept
bank	600	no	300	reject
none	2000	no	100	accept

Table 1: A set of examples for the retail credit task.

amount of data to work with, will not be that accurate but it is a good estimate considering the amount of data available. For example, let us assume that we have the following new instance to classify:

$$bank, 640, yes, 500, class =? \tag{1}$$

Our task is to predict the target value (accept, reject) of the target concept Output Class. Calculating the probabilities of the different target values based on the frequency over the 7 training examples we have,

$$\begin{aligned} P(OutputClass = accept) = \frac{4}{7} &= .5714 \\ P(OutputClass = reject) = \frac{3}{7} &= .4285 \end{aligned}$$

Similarly, we can estimate the conditional probabilities. For example, those for Account = bank are

$$\begin{aligned} P(Account = bank \mid OutputClass = accept) = \frac{1}{4} &= .25 \\ P(Account = bank \mid OutputClass = reject) = \frac{2}{4} &= .50 \end{aligned}$$

When handling continuous attributes, we divide the range of values into a fixed number of bins. In this case, we assume the number of bins to be 5. In the above example, the range for the attribute Balance is {0-3000}, if we evenly divide the range, the 5 bins are {0-600},{600-1200}, {1200-1800},{1800-2400},{2400-3000}. Using probability estimates and similar estimates for the remaining attribute values, we calculate the posterior probabilities for each class:

$$P(Class = accept)P(Acc = bank \mid Class = accept)P(Bal = \{600 - 1200\} \mid Class \quad = \quad accept)$$

$$P(Empl = yes \mid Class = accept) P(MthlyExp = \{440 - 520\} \mid Class = accept) \quad = \quad 0.0$$

$$\begin{aligned} P(Class = reject) P(Acc = bank \mid Class = reject) P(Bal = \{600 - 1200\} \mid Class &= reject) \\ P(Empl = yes \mid Class = reject) P(MthlyExp = \{440 - 520\} \mid Class = accept) &= .01058 \end{aligned}$$

Thus, the naive Bayes classifier assigns the target value Output Class = reject to this new instance, based on the probability estimates learned from the training data. Furthermore, by normalizing the above quantities to sum to 1 we can calculate the conditional probability that the target value is no, given the observed attribute values. For the current example, this probability is $\frac{.01058}{.01058+0.0} = 1.0$. In the above case, the table is small and the classification of new instances can be done very quickly even on a small computer. But what would happen if many more examples were used? The method suggested above might end up using too much time and resources. In the real world a bank would easily accumulate hundreds of thousands of examples in a few years, but would it be desirable to use them all? Would it be profitable and less time-consuming at the same time to use just a sample of the training data instead of the entire data collection? This thesis attempts to provide answers to the above questions.

1.2 Thesis

This thesis investigates three main questions about the induction of classifiers from very large datasets.

Question 1: *Can a model that has been trained on samples taken from the collection of training data perform as well as the model that has been trained using the entire training data collection?*

Question 2: *Will this method yield accuracies that are comparable to the methods that use more resources and processing time?*

Question 3: *Can using the ensemble approach overcome the disadvantages that come with using sub-sampling techniques?*

1.3 Motivation

In the world today, we have witnessed an explosion in the amount of data produced. We are drowning in information but are starved for knowledge. The deluge of data is clearly impossible to handle with present means. Uncontrolled and unorganized information is no longer a resource in an information society. In order to address this problem machine learning research has focussed on developing automatic data analysis techniques. Most of the techniques developed so far are computationally intensive even for moderate sized datasets (20,000 examples with 200 features) let alone large datasets (100,000 to 1,000,000 examples) with a large number of features (over 1000). Hence, in this work I focus on developing fast and accurate machine learning techniques that are capable of handling large datasets.

Previous research in the field [Catlett, 1991] has shown that using small subsets of the data points available when building classifiers lowers the accuracy of the classifier. In this thesis I address this limitation by scaling up the inductive learning technique using *ensemble classifiers*. An ensemble classifier consists of a set of individual classifiers (components) whose predictions are combined using an appropriate combiner mechanism. Empirical evidence suggests that an ensemble learner almost always outperforms an average classifier from the ensemble [Maclin and Opitz, 1997, Quinlan, 1996].

The naive Bayes classifier is used for several reasons. Besides being simple and fast it compares surprisingly well to other more complex learning algorithms like decision tree learning, rule learning, and instance based learning [Langely and Sage, 1997, Cestnik et al., 1986, Domingos and Pazzani, 1997]. The naive Bayes learning algorithm is a stable learning algorithm – small changes in the training data do not affect the classifier much compared to the other unstable algorithms. In addition, naive Bayesian classifier learning is robust to noise (incorrect training examples) and irrelevant attributes (attributes that are not relevant to predicting the target value). Therefore it is interesting to explore the effects of different learning techniques using this stable classifier on large datasets.

1.4 Context of Research

In this section, I will cover the context of the fields of research from which the thesis draws ideas.

1.4.1 Why Inductive Learning Techniques?

In a general sense, the terms *induction* or *inductive reasoning* refer to any progression from specific facts to general rules. The converse, *deductive reasoning* or *deduction* refers to the opposite: the application of general rules to specific facts to derive new facts. Deduction guarantees that if the premises are correct then the conclusions drawn from them will also be correct. Induction provides no such guarantee. But if the tasks are well-defined then induction can be performed according to a precisely defined procedure. These inductive algorithms are fundamental to the field of ML and were mainly motivated by the desire to automate the process of *knowledge acquisition*. Data can be collected by an expert in the field and supplied to the inducer, which can the train on this data and predict the output of any unknown future point that it may encounter.

1.4.2 Why the Naive Bayes Classifier?

The Naive Bayes Classifier [Good, 1965, Duda and Hart, 1973, Langely et al., 1992], sometimes called the simple Bayes algorithm, builds a simple conditional independence classifier. A probability estimate is determined for each output class and the prediction is made for the class with the largest posterior probability. In this implementation of the classifier, continuous attributes were discretized using the entropy discretization technique [Fayyad and Irani, 1993]. Probabilities were estimated using frequency counts with an m-estimate Laplace correction [Cestnik, 1990] as described in [Becker et al., 1997b]. The Naive Bayes classifier is relatively simple but very robust to violation of its independence assumptions. It performs well for many real-world datasets [Ali and Pazzani, 1996, Kohavi and Sommerfield, 1995] and in [Langely and Sage, 1997], the authors show that the Naive Bayes classifier is excellent at handling irrelevant attributes.

1.4.3 Why Ensemble Learners?

The main issue in inductive learning is the question of *generalization*. Given a set of training examples (i.e., pairs of inputs and corresponding outputs produced according to some underlying but unknown rule) one wants to generate a classifier which generalizes well (i.e., makes accurate predictions for the outputs corresponding to inputs not contained in the training set). More recently, it has emerged that generalization performance can often be improved by training not just one classifier but rather by using an ensemble (i.e., a collection of a finite number of predictors all trained for the same task). This idea of improving generalization performance by combining the predictions of many different classifiers has been investigated extensively in the literature [Granger, 1989, Wolpert, 1992a, Breiman, 1989].

When classifying a point using an ensemble classifier, the point is given as input to all the component classifiers. Each of these classifiers then predicts an outcome for the input point independent of one another. The predictors are then merged by a combiner mechanism. Empirical evidence suggests that an ensemble classifier almost always outperforms a simple classifier [Clemen, 1989, Quinlan, 1996, Wolpert, 1992a, Zhang et al., 1992]. Empirical evidence further suggests that an ensemble classifier almost always outperforms an average classifier from the ensemble [Maclin and Opitz, 1997, Quinlan, 1996]. Often the ensemble classifier outperforms *all* of its component classifiers. Hence it would be interesting to study ensemble learners in the context of this thesis, as it is possible for ensembles to overcome the disadvantages of using subsets of training data.

1.4.4 Why use very Large Datasets?

The meaning of the phrase "very large" in the context of induction has been changing rapidly with time. It used to mean hundreds or thousands of examples and now it means tens or hundreds of thousands. The number of attributes is important too, but it does not seem to exhibit such a wide variation as training size.

In domains where there is perfect information available, even a small set of training data would prove sufficient to build a classifier that could predict with high reliability. But in some domains perfect accuracy is not attainable given the imperfections of the data available. If examples are cheaply available, then the most economical way of raising accuracy is to use all of those examples to train, provided the induction process remains economical. However, it should be acknowledged that larger datasets are not always the only or the best way to use CPU time to improve accuracy.

1.4.5 Why Measure Learning Time?

If only a few hundred training instances are available, the choice between taking .6 seconds to process 100 examples and taking 0.9 seconds for 200 instances is unlikely to require close calculations on learning time, but if the choice is between taking 6 CPU hours on a large computer to process 100,000 examples and 2 CPU weeks to process a million, the choice becomes economically significant. The question arises whether the small increase in accuracy to be gained from using a larger set will justify the large increase in cost. Learning time on large datasets must be measured because of the danger of it becoming uneconomical or unmanageable or unbearably large.

1.5 What Follows

In the following chapter, I discuss the Naive Bayes Classifier, ensemble learners, ensemble building techniques, ensemble combining techniques, Bagging and Boosting. In Chapter 3, I present a set of experiments, with the methodology following each experiment and the analysis of the results obtained. These experiments include the preliminary baselining experiments and the advanced experiments. In the last chapter, I draw conclusions for the thesis.

2 Background

In this chapter, I will provide background material for this thesis. In the first section, I give an introduction to the Naive Bayes classifier. In section 2.2, I discuss the working of ensemble classifiers. This machine learning technique has been shown to work well in general. In sections 2.3 and 2.4, I cover background material on the mechanisms that have been suggested for building and combining the component classifiers in an ensemble. Finally, in sections 2.5 and 2.6, I present two ensemble classifier mechanisms that I use in this work: Bagging and Boosting.

2.1 The Naive Bayes Classifier

The Naive-Bayes Classifier [Good, 1965, Duda and Hart, 1973, Langely et al., 1992], sometimes called Simple-Bayes [Domingos and Pazzani, 1997], builds a simple conditional classifier based on an assumption of independence. Formally, the probability of an output class label value y for an unlabeled instance x containing n attributes, $< A_1, ..., A_n >$ that describes the new instance is given by

$$\begin{aligned} P(y \mid x) &= P(x \mid y) \cdot P(y)/P(x) && \text{by Bayes rule} \\ &\propto P(A_1, ..., A_n \mid y) && \text{since P(x) is same for all label values} \\ &= \prod_{j=1}^{n} P(A_j \mid y) \cdot P(y) && \text{by the conditional independence assumption} \end{aligned}$$

The above probability is computed for each class and the prediction is made for the class with the largest posterior probability. The probabilities in the above formulae must be estimated from the training set. The basic algorithm is scaled up using methods to handle the zero contents in probabilities and by using entropy minimization techniques to calculate the intervals for continuous attributes. Conditional independence is a property defined as follows: if A_1, A_2 and A_3 are attributes with output class y independent of A_1 and A_2 given A_3, then we have $P(y|A_1, A_2, A_3) = P(y|A_3)$ since y is independent of both A_1 and A_2 given A_3. Conditional independence may not always be a good assumption because the attributes may not always be independent of each other. In Table 1, we saw that each instance (x) had attributes $(A_1, ... A_4)$ Account, Balance, Employed, Monthly Expense and it belonged to either output class (y) accept or reject. The conditional probabilities are computed for each value of the various attributes; for example, we compute $P(Account = bank|OutputClass = accept)$ which is estimated by the fraction $\frac{n_c}{n}$ where $n = 4$ is the total number of training examples for which $OutputClass = accept$, and $n_c = 1$ is the number of these for which $Account = Bank$.

2.1.1 Laplace-m Approach

The probabilities estimated so far are calculated by the fraction of times the event is observed to occur over the total number of opportunities. While this observed fraction provides a good estimate of the probability in many cases, it provides poor estimates when n_c is very small. To see this difficulty assume that n_c is 0, for example in our data from Table 1, we see that $P(MonthlyExpense = 440 - 520|OutputClass = accept) = 0$. This causes two difficulties. First, $f = \frac{1}{n}$ produces a biased underestimate of the probability. Second, since this probability estimate is zero, this probability term will dominate the Bayes classifier if the future query contains $Account = bank$. The reason is that the quantity calculated requires multiplying all the other probability terms by this (zero) estimate.

The class probabilities and the conditional probabilities in the above formulae were based on pure frequency counts. As shown in the example above, an attribute value A_j that does not occur together with a given class label value will produce a zero estimate for $P(A_j \mid y)$ for some attribute A_j, eliminating class y from consideration. To overcome this problem of a single value controlling the outcome, I used the Laplace approach. In this implementation, the probabilities are estimated using frequency counts with an m-estimate Laplace correction [Cestnik, 1990]. Given a predefined factor f, if there are n_c matches out of n instances for a k value problem, we estimate the probability as $(n_c+f)/(n+kf)$. Laplace-m sets the adjustment f to be $1/m$, where m is the number of instances in training data set, making it smaller as the number of instances increases.

2.1.2 Multiple Interval Entropy Discretization

With continuous attributes, we face the problem of dividing the range of the attribute values into appropriate intervals. Empirical evidence suggests that the entropy-based discretization of continuous attributes performs well compared to other methods [Kohavi and Sahami, 1996, Dougherty et al., 1995].

In order to fix a criterion for deciding whether to accept or to reject a partition, we choose the Minimum Description Length Principle Criterion (MDLPC). It is defined as the minimum number of bits required to uniquely specify the object out of the universe of all objects. By object, we mean any generic entity such as a concept, a function, a set, etc. In general, it is assumed that there is some probability distribution governing the occurance of objects in the universe. This allows us to have interesting universes, like ours, where not all entities are equally likely to make an appearance at any given time. Hence we make use of the MDLPC to make a guess at the hypothesis with the higher probability, given a fixed set of instances. Specifically, the MDLPC states that given a set of data and a hypothesis, we accept a hypothesis if the total cost of coding the hypothesis and the data is less than that of coding the data alone. This criterion is chosen as it reduces the arbitrariness that would be caused if some other heuristic for deciding when to refrain from further partitioning is chosen.

Let p denote a point that partitions the set S of N examples into subsets S_1 and S_2. Let there be k classes $C_1, ..., C_k$ and let $P(C_i, S)$ be the proportion of examples in S that have class C_i. The *class entropy* of a subset S is defined as:

$$Ent(S) = -\sum_{i=1}^{k} P(C_i, S) \log(P(C_i, S)) \tag{2}$$

When the logarithm is base 2, $Ent(S)$ measures the amount of information needed, in *bits*, to specify the classes in S. To evaluate the resulting class entropy after a set S is partitioned into two sets S_1 and S_2, we take the weighted average of their resulting class entropies. For an example set S attribute A, and cut point which is a threshold value, T: Let $S_1 \subset S$ be the subset of examples in S with A-values $\leq T$ and $S_2 = S - S_1$. The *class information entropy of the partition* induced by $T, E(A, T; S)$, is defined as

$$E(A, T; S) = \frac{\mid S_1 \mid}{\mid S \mid} Ent(S_1) + \frac{\mid S_2 \mid}{\mid S \mid} Ent(S_2) \tag{3}$$

A binary discretization for A is determined by selecting the cut point T_A (defined as the average of two consecutive values in the set S for which the output class value changes) for which $E(A, T_A; S)$ is minimal amongst all the candidate cut points. The information gain of a cut point is:

$$\begin{aligned} Gain(A, T; S) &= Ent(S) - E(A, T; S) \qquad (4) \\ Gain(A, T; S) &= Ent(S) - \frac{\mid S_1 \mid}{\mid S \mid} Ent(S_1) - \frac{\mid S_2 \mid}{\mid S \mid} Ent(S_2) \qquad (5) \end{aligned}$$

The cost for the two competing hypothesis {HT,NH} (i.e., the hypothesis induced by the partition and null-hypothesis) is calculated as follows:

$$Cost(NH) = N \cdot Ent(S) + k \cdot Ent(S) \tag{6}$$

$$\begin{aligned} Cost(HT) &= \log_2(N-1)+\mid S_1 \mid \cdot Ent(S_1)- \mid S_2 \mid \cdot Ent(S_2) \\ &\quad + \log_2(3^k-2) + k_1 Ent(S_1) + k_2 Ent(S_2) \end{aligned} \tag{7}$$

where k_1 and k_2 are the number of classes in sets S_1 and S_2 respectively. The MDLPC prescribes accepts a partition iff $Cost(HT) < Cost(NH)$. Examining the condition under which $[Cost(NH) - Cost(HT)] > 0$:

$$\begin{aligned} 0 &< NEnt(S)- \mid S_1 \mid \cdot Ent(S_1)- \mid S_2 \mid \cdot Ent(S_2) - \log_2(N-1) \\ &\quad + kEnt(S) - \log_2(3^k-2) - k_1 Ent(S_1) - k_2 Ent(S_2) \end{aligned} \tag{8}$$

The above inequality, after dividing by N, reduces to

$$Gain(A,T;S) - \frac{\log_2(N-1)}{N} > \frac{\triangle(A,T;S)}{N} \tag{9}$$

where

$$\triangle(A,T;S) = \log_2(3^k-2) - [kEnt(S) - k_1 Ent(S_1) - k_2 Ent(S_2)] \tag{10}$$

Hence the MDLP Criterion - The partition induced by a cut point T for a set S of N examples - is accepted iff

$$Gain(A,T;S) > \frac{\log_2(N-1)}{N} + \frac{\triangle(A,T;S)}{N} \tag{11}$$

and it is rejected otherwise.

Now that the selection criterion is fixed for the intervals, I will describe the algorithm that splits the range of continuous attributes into multiple intervals. The procedure for selecting a fixed point (Equation 9) can be applied recursively to each of the binary subsets resulting from the selected cut point to yield multiple intervals as shown in the algorithm **Multi-Interval** (see Table 2). Given N examples, this procedure will partition the range of the attribute to anywhere from 2 to $N-1$ possible intervals. Algorithm **Interval-Extract** returns a set of intervals for attribute A and example set S by calling the recursive algorithm **Multi-Interval**. The algorithm **Multi-Interval** adds as many cut values to the list Val-list as the MDLPC sees fit. The final product is a list of cut values. Algorithm **Interval-Extract** simply sorts this list of values in ascending order and constructs a list of intervals for attribute A. Each cut value, along with its immediate successor in the sorted Val-List, defines the lower and upper bounds of an interval, respectively. Prior to the calling of algorithm **Multi-Interval**, a binary interval partition is forced on the range of A.

Multi-Interval(A, S, Val-List)

A: A continuous-valued attribute.

S: a set of training examples on which A is defined.

Val-List: pointer to a list of cut values.

```
If (| $S$ |$< 2$) return;
$T \leftarrow$ Selected-Cut-Value$(A, S)$;
If(MDLPC accepts T) then
    Insert(T,Val-List);
    $S_1 \leftarrow e \in S \mid A(e) < T$;
    $S_2 \leftarrow S - S_1$;
    Multi-Interval$(A, S_1;$ Val-List);
    Multi-Interval$(A, S_2;$ Val-List);
```

Table 2: The Multi-Interval Algorithm for splitting a set of of continuous values into intervals.

To illustrate the discretization process, let us consider Table 1 There are two continuous attributes, Balance and Monthly Expense. The ranges are $\{0-3000\}$ and $\{100-600\}$ for the

Interval-Extract(A, S)

A: A continuous-valued attribute.

S: a set of training examples on which A is defined.

```
$T \leftarrow$ Selected-Cut-Value$(A, S)$;
$S \leftarrow e \in S \mid A(e) < T$;
$S_2 \leftarrow S - S_1$;
Val-List $\leftarrow (T)$;
Multi-Interval$(A, S_1;$ Val-List$)$;
Multi-Interval$(A, S_2;$ Val-List$)$;
Val-List $\leftarrow$ Sort(Val-List);
$x \leftarrow$ First(Val-List);
Val-List $\leftarrow$ Rest(Val-List);
Intervals $\leftarrow (-\infty, x)$;
While (Val-List is not Empty) Do
        $y \leftarrow$ First(Val-List);
        Intervals $\leftarrow$ Intervals $\cup [x, y]$;
        $x \leftarrow y$;
Intervals $\leftarrow [y, \infty]$;
Return(Intervals);
```

Table 3: The Interval-Extract Algorithm for splitting the continuous range of an attribute into intervals.

attributes respectively. In order to minimize the time taken for the process, we make use of the fact that the selected cut-points are always boundary points (*i.e.*, points where the Output class value changes). In the loan example we have the cut-point 650 for the Balance attribute and cut-points 250, 350, 450 and 550 for the Monthly Expense attribute. Since 650 is the only cut-point for attribute Balance, it is chosen. Hence the range for attribute Balance is now split into two: $\{0-650\}$ and $\{650-3000\}$. For the second attribute, Monthly Expense, we calculate the entropies of the cut-points 250, 350, 450 and 550.

$$\begin{aligned} E(cut-point=250) &= 0.69 \\ E(cut-point=350) &= 0.86 \\ E(cut-point=450) &= 0.98 \\ E(cut-point=550) &= 0.79 \end{aligned}$$

So we choose the cut-point that gives the lowest entropy value (250). Since this is the first cut-point, we force it as a partition. Now there are two intervals $\{100-250\}$ and $\{250-600\}$. Since there are no other cut-points in the first partition, we consider the remaining interval $\{250-600\}$. We calculate the entropies for the cut-points 350, 450 and 550.

$$\begin{aligned} E(cut-point=350) &= 0.95 \\ E(cut-point=450) &= 0.95 \\ E(cut-point=550) &= 0.8 \end{aligned}$$

Now we choose cut-point 550 as it has the lowest entropy value. Checking if the MDLPC holds for the selected cut-point, we compute $Gain(A,T;S) = 0.17$ and $\triangle(A,T;S) = 2.87$.

Since $Gain(A, T; S) < 0.97$, we reject the partition as it fails to satisfy the MDLPC.

$$Gain(A, T; S) > \frac{\log_2(N-1)}{N} + \frac{\triangle(A, T; S)}{N} \tag{12}$$

Hence we have selected two intervals, $\{100-250\}$ and $\{250-600\}$, for the attribute Monthly Expense.

2.2 Ensemble Learners

The main issue of inductive learning is the question of *generalization*: given a set of training examples (*i.e.,* pairs of inputs and corresponding outputs produced according to some underlying but unknown rule), one wants to produce, by a suitable training algorithm, a classifier which generalizes (*i.e.*, makes accurate predictions for the outputs corresponding to inputs not contained in the training set).

More recently, it has emerged that generalization performance can often be improved by training not just one classifier but rather by using an ensemble (*i.e.*, a collection of a (finite) number of predictions, all trained for the same task). This idea of improving generalization performance by combining the predictions of many different classifiers has been investigated extensively in literature [Granger, 1989, Wolpert, 1992a, Breiman, 1989].

Consider the real-life situation in which we are trying to predict the next day's weather. Ten copies of the same weather forecast may contain the same amount of information as just one copy. By obtaining ten *different* forecasts, however, it may actually be possible to predict the next day's weather more accurately, even if the forecasts are all based on the same satellite data, as the aspects not given importance in some forecasts may be covered by others and the "good" aspects of the predictions will hopefully be reinforced. The same is true quite generally for ensemble learning. But only when the classifiers in an ensemble

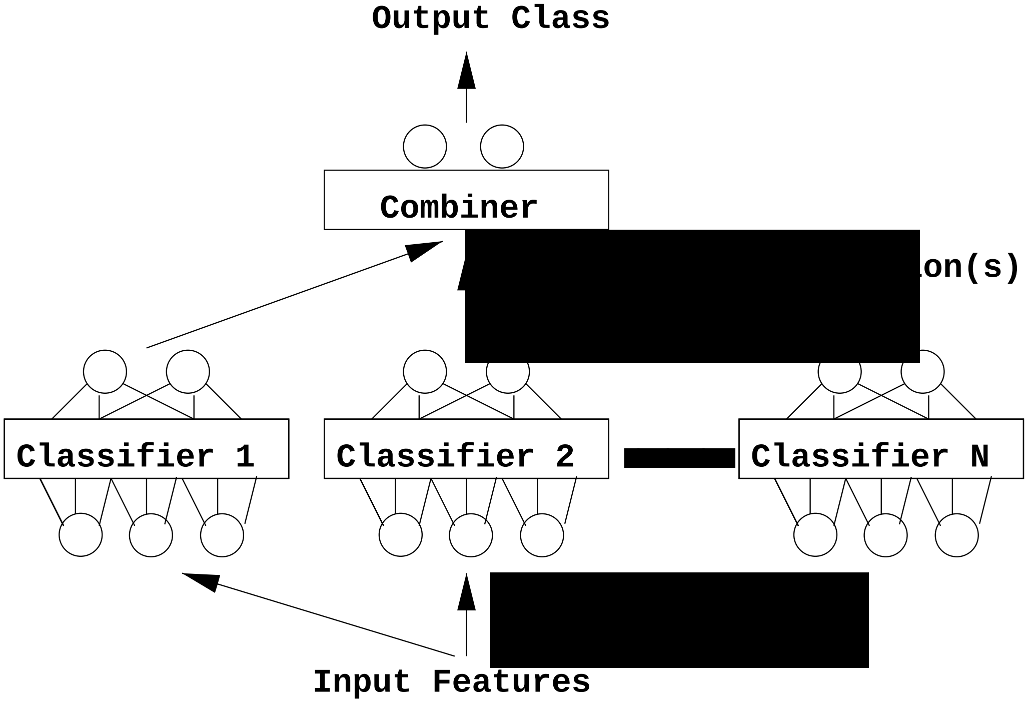

Figure 1: A classifier ensemble made up of N component classifiers and a combine mechanism.

are different is there something to be gained from using an ensemble.

An ensemble classifier consists of a set of individual classifiers (see Figure 1). The predictions of these classifiers, produced by different techniques, are combined by a suitable combiner mechanism. When classifying a point using an ensemble classifier, the point is given as input to all the component classifiers. Each of these classifiers then predicts an outcome for the input point independent of one another. The predictors are then merged by a combiner mechanism. Empirical evidence suggests that an ensemble classifier almost always outperforms an individual classifier [Clemen, 1989, Quinlan, 1996, Wolpert, 1992b, Zhang et al., 1992]. Often the ensemble classifier outperforms *all* of its component classifiers. Numerous methods have been suggested for the creation of component classifiers and for the combining their predictions. These methods will be discussed in Sections 2.3 and 2.4 in detail.

In order to increase the *disagreement* between the component classifiers, the classifiers could be trained using the same method with different training sets in each case. This method tries to produce members that are more *diverse*, so as to reduce the ensemble error more than the error of the individual members [Krogh and Vedelsby, 1995]. The above methodology opens up a plethora of ideas for the creation of optimal ensemble learners. The disagreement between the ensemble classifiers can be considered as the ambiguity in their classifications. When an ensemble of classifiers is considered it was shown that the generalization error of the entire ensemble is equal to the weighted average error of the individual classifiers minus the ensemble ambiguity (the weighted average of the individual ambiguities) [Krogh and Vedelsby, 1995].

2.3 Ensemble Building Techniques

The classifiers in the ensemble are usually trained independent of each other before combining their predictions. Examples of mechanisms used to build component classifiers include: (i) using different training parameters with a single learning method [Alpaydin, 1993, Drucker et al., 1994, Maclin and Shavlik, 1995]; (ii) using different subsets of training data with a single learning method [Breiman, 1996, Freund and Schapire, 1996]; (iii) using different learning methods [Zhang et al., 1992]; and (iv) explicitly searching for a set of classifiers that is both accurate and diverse [Opitz and Shavlik, 1996]. Bagging [Breiman, 1996] and Boosting [Freund and Schapire, 1996] are two of the most successful methods used in ensemble learning that create component classifiers by using different subsets of the training data.

2.3.1 Single Training Method - Different Training Parameters

This method involves the use of the same training method with different training parameters to create an ensemble and has been investigated by many researchers. In [Drucker et al., 1994], the authors suggest a way to build ensembles of neural networks by using different initial weights for each network in the ensemble. The training sets chosen to build each of the networks were selected randomly. In order to ensure that each network is accurate, training was done until the mean square error reached a pre-defined minimum. In [Maclin and Shavlik, 1995], a combination of networks initialized via competitive learning is used. Competitive learning was used to partition training data into clusters, each representing one output category. Neural networks that have a hidden unit corresponding to every cluster in all the categories were created. These were then combined to create the ensemble classifier. The variability among the networks was achieved through randomly choosing seeds for the competitive learning algorithm and training the algorithm with random sequences of examples.

2.3.2 Single Training Method - Different Subsets of Training Data

This technique of building ensembles is followed in the popular methods of Bagging and Boosting. Bagging [Breiman, 1996] is a method that creates ensembles by randomly selecting training examples, with replacement, for each of its component classifiers. Boosting [Freund and Schapire, 1996], on the other hand, involves creating classifiers sequentially, applying them to re-weighted versions of the training data.

2.3.3 Single Dataset - Different Training Method

In [Zhang et al., 1992] a hybrid system is built using three different experts which are trained using the same dataset (see Figure 2). The training system consisted of a statistical model,

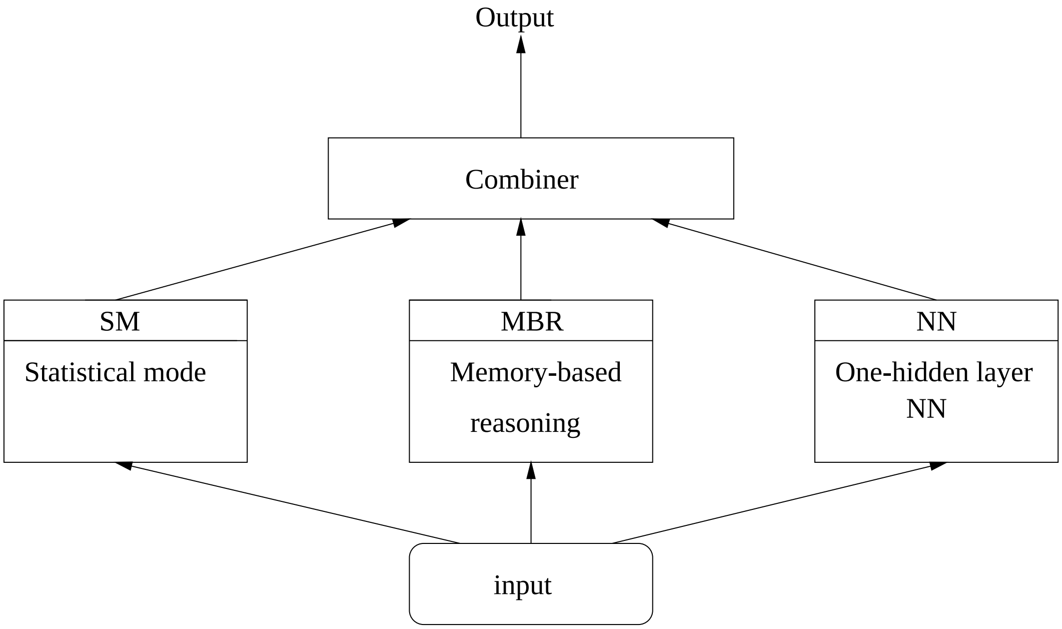

Figure 2: The hybrid system has 3 experts, a statistical module, a memory-based reasoning module and a NN module. The combiner combines the outputs to produce the final output [Zhang et al., 1992].

a memory-based reasoning model and a neural network model. The training of the whole system involves (1) training the 3 experts and (2) training the Combiner. In order to train the Combiner, half of the training data is used to train the 3 experts separately and the outputs of these trained experts on the second half of the training data are recorded. These outputs are then used as inputs to train the Combiner. After the training of the Combiner is completed, each expert is trained using the whole training set. These trained experts together with the trained Combiner form a trained hybrid system. Each expert is then allowed to make predictions completely independent of the other. The predictions were then combined using the trained Combiner to yield the final output.

2.3.4 Searching for Accurate and Diverse Classifiers

In [Opitz and Shavlik, 1996], a neural-network ensemble was built by explicitly looking for neural networks that committed errors on different parts of the input space. Genetic operators were applied continually to an initial population of networks to create new networks, keeping only the set of networks that were highly accurate and diverse in every generation.

2.4 Ensemble Combining Techniques

The performance of the learner can be fine-tuned to get the highest possible accuracy on a validation set. This fine-tuning can be a complex task when there are patterns on which even the best learner is not accurate enough when some other learner may be. By suitably combining these learners, performance can be improved. As there is no point in combining multiple learners that always make similar decisions, the aim is to be able to find a set of learners that differ in their decisions to complement each other.

Combining the component classifiers of an ensemble can be done in many ways. Examples of methods used for combining classifiers include: (i) voting [Hansen and Salamon, 1990]; (ii) simple averaging (in regression) [Lincoln and Skrzypek, 1989]; and (iii) weighted averaging [Freund and Schapire, 1996, Perrone, 1993, Rogova, 1994].

2.4.1 Notation for Combining Predictors

Let us say we have k learners. Let ϑ_j denote the estimate of learner j given input x. Let S be the training dataset with data $(y_n, x_n), n = 1, ..., N$, where the y's are class labels for the input x's. For now we ignore how ϑ_j are calculated and instead we concentrate on the ways in which they could be combined to find C, to get the final estimate as

$$C = g(\vartheta_1(x, S), \vartheta_2(x, S), ..., \vartheta_k(x, S) \mid \psi) \tag{13}$$

where g is the combiner function with ψ denoting its parameters. When there are c outputs, each learner gives c outputs which are combined. If we are doing classification, assuming that C_i estimates the posterior probability of class i and the zero-one loss function is used to minimize Bayesian risk, we choose the class with the highest probability

$$C^* = arg \max_i C_i \tag{14}$$

The combining of the classification powers of several classifiers is considered a general problem in many areas and a systematic investigation has been made. The following approaches for solving this problem are proposed, based on different methodologies.

2.4.2 Voting

The simplest way to combine multiple classifiers is by *voting* which corresponds to taking a linear combination of the learners. If g_j denotes the weight of the learner j, the final output is computed as

$$r = \sum_{j=1}^{m} g_j . \vartheta_j \tag{15}$$

satisfying the following conditions

$$\forall j, g_j \geq \ 0 \text{ and } \sum_{j=1}^{m} g_j = 1 \tag{16}$$

Simple Voting All the voters in this case have equal weight, i.e., $g_j = 1/m$. Hansen and Salamon [1990] used this technique to combine the predictions of multiple neural networks in the ensemble. The randomness in the networks was brought about by assigning different initial weights and sequencing of training examples to the networks. As each of these networks made generalization errors on different subsets of the input space, the collective decision made by them is less likely to be in the error than in the decision made by any of the individual networks. They concluded that the ensemble can be far less fallible than any one network.

Weighted Voting If the voters can supply additional information on how much they vote for each class, then these can be converted to certainties and used as weights in a *weighted voting* scheme, for example, proportional to the difference of the two highest outputs [Alpaydin, 1993]. Another possibility is to assess the accuracies of the classifiers on a separate cross-validation set and use that information to compute the weights. Xu et al. [1992] discuss various ways in which the outputs of several classifiers are combined. To compute the weights, they propose to use a *belief* measure or the Dempster-Schafer theory [Rogova, 1994]. Freund and Schapire [1996] combined the classifiers created by the Boosting technique by weighting them according to their performance over the training set. In [Lincoln and Skrzypek, 1989], the authors propose a way to learn the weights in a voting scheme. In [Alpaydin, 1993], the model complexities in a Bayesian framework are taken into consideration by giving larger weights to simpler models. In [Perrone, 1993] the author gives a number of didactic examples that depict the advantage of voting. He also shows that for a minimum square error, when the learners are unbiased and uncorrelated, weights should be inversely proportional to variances.

2.4.3 Voting sequence of Combiners

In [Asker and Maclin, 1997] a system called SEQUEL is introduced which suggests a novel way to combine classifiers. It implements a method for combining the predictions of k classifiers trained on n examples. The method assumes that each classifier f_k produces a probability estimate so that $f_k(x)$ gives the probability that x is an instance of a target concept C. Each classifier x is an instance of a threshold (τ_k), which is the probability given to the negative example with the highest probability. A classifier is considered competent for an example x if $f_k(x) \geq \tau_k$. The prediction of the ensemble $f^*(x)$ is the prediction of the best classifier multiplied by the product of the thresholds of all previous classifiers.

$$f^*(x) = \left(\sum_{i=1}^{k(x)-1} \tau_i \right) f_{k(x)}(x) \tag{17}$$

2.4.4 Learning a Combiner function

Wolpert [1992a], describes a stacking mechanism that extends voting in the sense that the output of the learners are combined through a combiner system that has also been trained. In the original terminology, the learners are called the level 0 generalizers and the combiner is the level 1 generalizer. Let's denote the level 0 generalizers as the ϑ_j and the level 1 generalizer as the combining function $g(\cdot \mid \psi)$ specified up to a parameter vector ψ. For example $g(\cdot)$ may be a multi-layer perceptron and ψ its connection weights

$$r = g(\vartheta_1, \vartheta_2, ..., \vartheta_m \mid \psi). \tag{18}$$

The level 1 generalizer learns what the correct output is when level 0 generalizers give a certain output combination. Thus level 1 needs be trained on data unused in training the level 0 generalizers. Wolpert proposes the use of the leave-one-out strategy though this is too costly and k-fold cross validation is more efficient when a larger sample exists. Unlike voting, in this case $g(\cdot)$ can also be non-linear. In [Zhang et al., 1992] stacking for protein secondary structure prediction was shown to significantly improve in accuracy. In their study, the level 0 generalizers are a statistical model, a memory-based learner and a one hidden layer network. The level 1 generalizer is another neural network with one hidden layer. Maclin [1998] discusses a method that uses a neural network to predict the accuracy of the predictions of each classifier in the ensemble. A k-Nearest Neighbor method for predicting the accuracy of the classifier, when combining, is also discussed.

2.5 Bagging

The **Bagging algorithm** (**B**ootstrap **agg**regat**ing**) by Breiman [1996] votes classifiers generated by different bootstrap samples (replicates). The algorithm is shown in Table 4. The classifier denoted by $\vartheta(x, S)$ predicts the output y, using x as the input. A sequence of learning sets S are generated, each consisting of N independent observations from the same underlying distribution S. Using these learning sets we generate a sequence of predictors $\{\vartheta(x, S_k)\}$. The predictions of these predictors are then combined using majority voting.

The replicate data sets $S^{(B)}$, each consisting of N cases, are drawn at random *but with replacement*, from S. Each (y_n, x_n) may appear repeated number of times or not at all in any particular $S^{(B)}$. Hence the $S^{(B)}$ are a replicate data set drawn from the bootstrap distribution approximating the distribution underlying S. A critical factor in whether Bagging

Input: training set S, Classifier ϑ, integer T (number of bootstrap samples)

```
for $i = 1$ to $T$
    $S' =$ bootstrap sample from $S$ (sample with replacement).
    $C_i = \vartheta(S')$
$C^*(x) = \arg\max_{y \in Y} \sum_{i:C_i(x)=y} 1$ (the most often predicted label y)
```

Output: classifier C^*

Table 4: The Bagging Algorithm votes classifiers generated using different bootstrap samples.

will improve accuracy is the stability of the predictor ϑ. If minor changes in the training data set causes major changes in the behavior of the predictor ϑ, then the predictor is considered unstable. Bagging is seen to work well with unstable predicting methods. If changes in S (i.e. a replicate S) produce small changes in ϑ, then ϑ_B will be close to ϑ. Improvement will result only otherwise. Some unstable procedures were found to include neural networks, classification and regression trees, and subset selection in linear regression, while k-nearest neighbor methods were found to be stable.

2.6 Boosting

Boosting was introduced by Schapire [1990] as a method for boosting the performance of a weak learning algorithm. After improvements by Freund[1990], **AdaBoost** (**Ada**ptive **Boost**ing) was introduced by Freund and Schapire [1995]. But later another version of the algorithm called AdaBoost.M1 [Freund and Schapire, 1996] was introduced, and Table 5 shows this algorithm. This algorithm generates classifiers sequentially, while Bagging can generate them in parallel. AdaBoost also changes the weights of the training instances provided as input to each inducer based on the classifiers that were previously built.

Given an integer k specifying the number of trials, k weighted training sets $S_1, S_2, ..., S_k$ are generated in sequence and k classifiers $C_1, C_2, ..., C_k$ are built. A final classifier C^* is formed using a weighted voting scheme where the weight of each classifier depends on its performance on the training set used to build it. The probability of picking an example is initially set to $\frac{1}{N}$ (where N is the number of original examples). After a classifier is added to the ensemble, the probabilities are adjusted by a factor based on ϵ_k, the sum of probabilities of the examples that were incorrectly classified by the previous classifier. The probability of selecting each of the misclassified examples is multiplied by $\frac{(1-\epsilon_k)}{\epsilon_k}$. Then all the probabilities are renormalized to sum to one. Hence the probabilities of misclassified examples are increased whereas the probabilities of correctly classified examples are decreased. The

```
Input: training set $S$, Classifier $\vartheta$, integer $T$ (number of trials)

1.  $S' = S$ with instance weights assigned to be 1.
2.  For $i = 1$ to $T\{$
3.      $C_i = \vartheta(S')$
4.      $\epsilon_i = \sum_{x_j \in S' : C_i(x_j) \neq y_j}$ weight$(x)$ (weighted error on training set).
5.      If $\epsilon_i > \frac{1}{2}$
6.          Set $S'$ to a bootstrap sample from $S$ with weight 1 for every instance.
7.          Goto step (3) (this loop is limited to 25 times after which we exit the loop).
8.      $\beta_i = \frac{\epsilon_i}{(1-\epsilon_i)}$
9.      For-each $x_j \in S'$, if $C_i(x_j) = y_j$ then weight$(x_j)$ =weight$(x_j) \cdot \beta_i$
10.     Normalize the weights of instances so the total weight of $S'$ is $m$.
11. $\}$
12. $C^*(x) = \arg\max_{y \in Y} \sum_{i : C_i(x) = y} \log \frac{1}{\beta_i}$
Output: classifier $C^*$
```

Table 5: The AdaBoost Algorithm weights the predictions of the classifiers trained using examples selected from a weighted training set.

predictions of the classifiers are not simply averaged because the later ones may be effective in classifying only a small subset of points. To address this problem the predictions of the classifiers are weighted by the factor $\log(\frac{(1-\epsilon_k)}{\epsilon_k})$. Hence classifiers with small values of ϵ_k will be weighted higher than those with large values, which results in fewer errors on the training set.

3 Experiments

This chapter describes the datasets and the experimental methods used in the empirical evaluations in this thesis. Results obtained for the various methods are presented and analyzed. Machine learning studies intelligent artifacts - systems created by the researchers who study them. Research in the field has thus tried to emphasize formal analysis and theoretical approaches. Indeed considerable progress has recently occured on the theoretical front, both in formalizing the nature of learning algorithms and characterizing their behavior. Despite this progress, some algorithms still remain too complex for formal analysis. In such cases, empirical studies of these algorithms must retain a central role. Hence, in this work I empirically evaluate the new technique of learning from sub-samples.

In any science, the goal of experimentation is to understand a class of behaviors and the conditions under which they occur. Formally defined, an experiment involves systematically varying one or more *independent* variables and examining their effect on some *dependent* variables. In this thesis the variables most often varied are the learning methods and the number of training instances. The dependent variables observed are error rate and learning time.

3.1 Datasets

The datasets were drawn from the UCI repository with emphasis on ones that were previously investigated by other researchers. This section overviews the characteristics typically used to describe datasets, explains the basis on which the datasets used in this thesis were selected, and gives a brief description of each. The following questions deal with analyzing the different characteristics of the datasets:

- How many training instances does the dataset comprise?
  - Indicates the availability of data for learning.

- How many classes are there?
  - Indicates the possible distribution of the instances among the output classes.
- What is their distribution (i.e., their relative frequencies)?
  - Gives a general idea about the frequency distributions of the instances among the output classes.
- How many attributes are there?
  - Gives an indication of how elaborate the description of every instance in the dataset is.
- Are the attributes all real-valued, all discrete, or mixed?
  - Provides an indication of the nature of the different attributes.
- What is the distribution of the values of the attributes?
  - Provides an indication of the distribution of the attribute values for each attribute.
- Are the values ever unknown?
  - Gives an idea about the uncertainty of the availability of data for the dataset.

A harder question to answer is: What is the Bayes optimal error rate (which is the error rate calculated knowing the estimates of the probability density functions of samples of classes)? The Bayes optimal error is the lowest error rate that can be possibly be obtained with any particular problem. This question is complicated by noise, which can make even a simple concept difficult to formulate. The phrase "the concept" seems to presume that a 100% correct concept exists, but perfect accuracy may not be achievable and the best concept may be unknown. The notion of complexity is bound up with the way the concept is found and represented; no independent metric of complexity of the concept has gained acceptance. However, the other descriptors listed above give an indication of the complexity of the dataset.

One last but important question concerns the origin and purpose of the data: is it a *natural* domain of interest to someone other than machine learning researchers (as opposed to *synthetic* (or *artificial*), contrived by those researchers to investigate algorithms rather than some external phenomenon)? A dataset can still be labeled as *natural* even though it was generated by a computer, such as the simulation data for the satellite dataset; the distinction is made on the basis of whether the dataset concerns some practical real-world task.

The datasets used in this thesis were chosen to "show real-world relevance" and "provide evidence of generality". I used almost all the natural domains of sufficient size (at least 5000 training instances) that I could find in the UCI repository. I have included the smaller datasets also in my experiments to study the nature of the standard techniques in a more complete way. Table 6 describes the datasets providing answers for most of the questions that were listed above.

3.2 Experimental Methodology

In this section I will describe the experiments that were performed, what they measured, and how the results were interpreted. First and foremost, I will elaborate on the general methodology followed in carrying out the various experiments. The details of the specific experiments are then given in later sections.

For supervised concept induction tasks, the first dependent variable to measure is the percentage of correctly classified future instances. Given a particular performance criterion, one must estimate this value in some fashion. For estimating accuracy, a model is built from a set of data called the *training data* and tested against a set of data set aside for test-

Data set	Dataset size	Attributes Continuous/Nominal	Unknown Attributes?	Classes
agaricus-lepiota	8124	0/22	yes	2
breast-cancer-wisconsin	699	10/0	yes	2
cleveland-heart	303	8/5	yes	2
credit-a	690	6/9	yes	2
credit-g	1000	7/13	no	2
DNA	21623	0/13	no	3
glass	214	9/0	no	7
hepatitis	155	6/13	yes	2
house-votes-84	435	0/16	yes	2
hypo	3772	7/22	yes	5
hypothyroid	3163	7/18	yes	2
ionosphere	351	34/0	no	2
iris	150	4/0	no	3
kr-vs-kp	3196	0/35	no	2
labor	57	8/8	yes	2
led-creator-+17	5000	0/24	no	10
letter-recognition	20000	16/0	no	26
nursery	12960	0/8	no	5
pima-indians-diabetes	768	8/0	no	2
promoters-936	936	0/57	yes	2
rbs	1877	0/49	no	2
satellite	6435	0/36	no	6
segmentation	2310	0/19	no	7
sick-euthyroid	3163	7/22	yes	2
sick	3772	7/18	yes	2
sonar	208	60/0	no	2
soybean-large	683	0/35	yes	19
splice	3190	0/60	yes	3
vehicle	846	18/0	no	4
waveform	5000	40/0	no	3

Table 6: The characteristics of the datasets used in the experiments are shown. For each dataset the size of the dataset, the number of continuous and nominal attributes, information about missing attributes and the number of classes are shown.

ing called the *testing data.* To produce an unbiased estimate, these sets should be disjoint and selected randomly from the available data. In the classification domain, the predictive accuracy of a learning algorithm depends on predicting properties of unseen instances, not on summarizing aspects of instances it has already processed.

In this thesis, I have adopted the 10-fold cross-validation technique. In this technique, the entire dataset is randomly divided into 10 equal parts and each of these 10 sets is used as the testing set against the classifier, keeping the remaining 9 sets as training sets each time. The results are then averaged over the observed results. These partitions are selected by random sampling, where the partitions are randomly generated and the error rate is averaged across them. Besides improving error estimates, there are a number of significant advantages to sampling. The goal of separating a sample of cases into a training set and testing set is to help produce an unbiased error estimate. With a single train and test partition, too few cases in the training group can lead to erroneous error estimates. The 10-fold cross validation technique allow for more accurate estimates as it guarantees testing on all instances in the dataset.

The other dependent variable measured was the time taken. Since the major goal of this thesis is to find methods of speeding up learning by a large factor without significantly altering accuracy, learning time is scrutinized closely along with changes in error rate.

3.3 Preliminary Experiments - Naive Bayes

In this section, I discuss the preliminary tests conducted with the Naive Bayes Classifier. These tests are run to study the behavior and efficiency of the base classifier. These results are used, in later sections, for efficiency comparisons with results obtained from more complex techniques implemented using this same base classifier.

3.3.1 Methodology

Our main concern with estimating accuracy is that the estimate should be precise. Hence, we use the 10-fold cross validation technique. As mentioned above, this technique is used to get better error estimates. The Naive Bayes classifier was trained on 9 folds each time leaving one fold to remain the test fold. These results give an idea of the performance and accuracy of the base classifier used throughout this thesis on unseen examples.

3.3.2 Results

The Naive Bayes classifier discretizes using entropy, estimating probabilities using the Laplace-m technique and ignoring unknown values during classification as described in Chapter 2. In Table 7, the results obtained using the Naive Bayes classifier are compared with standard C4.5 results [Quinlan, 1996, Becker et al., 1997a, Dietterich, 1998, Maclin and Opitz, 1997]. These results are averaged over ten standard 10-fold cross-validation experiments.

3.3.3 Analysis

The average error for C4.5 is 13.21% and for Naive Bayes it is 13.57%. The results in Table 7 show clearly that the accuracy of the Naive Bayes classifier is comparable to that of C4.5. If we ignore the large datasets (agaricus-lepiota, kr-vs-kp, letter-recognition, satellite and segmentation) C4.5's error is 15.61% and Naive Bayes's error is 14.29%. But this estimate will not be very useful as I am going to concentrate on the larger datasets in this work.

The Naive Bayes is an efficient an accurate algorithm. Clearly from the results we can see that its accuracy is very good on small datasets when compared with larger ones. This behavior can be attributed to the fact that the classifier may asymptote to a high error rate, making it less useful as a classifier for very large datasets.

Dataset	Naive Bayes Error	C4.5 Error	Sign Test (Naive Bayes vs. C4.5)
agaricus-lepiota	0.78	0.00	+
labor	11.90	19.12	-
breast-cancer-wisconsin	3.37	5.25	-
letter-recognition	26.35	12.36	+
cleveland-heart	17.52	24.02	-
pima-indians-diabetes	25.22	25.31	-
credit-a	13.82	14.55	-
promoters-936	5.81	12.8	-
credit-g	26.46	28.96	-
rbs	9.08	11.2	-
glass	29.99	33.17	-
satellite	18.67	13.8	+
hepatitis	15.82	20.75	-
segmentation	7.04	3.7	+
house-votes-84	9.98	5.06	+
sick	3.03	1.3	+
hypo	0.92	0.48	+
sonar	22.21	27.42	-
hypothyroid	1.42	0.73	+
soybean-large	7.69	8.20	-
ionosphere	10.09	10.79	-
splice	5.41	5.9	-
iris	8.33	5.20	+
vehicle	39.23	29.4	+
kr-vs-kp	12.55	0.75	+
waveform	20.0	23.41	-

Table 7: Results obtained for the datasets from Table 6 using the Naive Bayes classifier and C4.5. The third column, shows the difference sign for the results obtained from both the classifiers for statistical analysis using the Sign Test.

While the Naive Bayes classifier shows good performance on many of the datasets, it is still a very limited classifier. It is a "global classifier" and cannot make local predictions. Hence the Naive Bayesian inducer cannot be *consistent* in the statistical sense without additional assumptions (an inducer is consistent if the classifiers it produces approach the Bayes optimal error as the dataset grows to infinity).

Applying the sign test to the distribution of results, we assume that the probability of a type I error, denoted by α, is .05. Formulating the decision rule, we have:

Decision Rule: Reject the hypothesis of the identity of the two distributions if the number of positive differences among the matched pairs is between $0, \ldots 7$ and $19 \ldots 26$ (called the critical region).

In the case of comparing the Naive Bayes with C4.5, the number of positive differences computed is 11 out of 26 datasets, and since this is outside the critical region we do accept the hypothesis of the identity between the two distributions. Hence, we conclude that the results obtained from the Naive Bayes classifier and C4.5 are comparable to each other.

3.4 Preliminary Experiments - Bagging and Boosting

This section, presents results obtained by applying the techniques of Bagging and Boosting to the Naive Bayes classifier. These results serve as a baseline for the future experiments conducted with ensembles of classifiers.

3.4.1 Methodology

This set of experiments involved running the Bagging and the Boosting algorithms on the datasets. In these experiments, the 10-fold cross-validation technique was applied to the Bagging and Boosting algorithms presented earlier in Sections 2.5 and 2.6.

3.4.2 Results

Table 8 shows the results obtained for Bagging and Boosting. Figure 3 shows a bar-graph of the results in Table 8 for a few selected datasets. Since the later experiments are conducted on ensemble classifiers, the simple classifier Bagging and Boosting results will provide a baseline for the later experiments. These results will aid the reader in obtaining a better

Dataset	Base Error	Bagging Error	Boosting Error	Sign Test (Base/Bag)	Sign Test (Base/Boost)	Sign Test (Bag/Boost)
agaricus-lepiota	0.78	0.48	0.0	+	+	+
breast-cancer-wisconsin	3.37	3.14	3.74	+	-	-
cleveland-heart	17.52	17.38	18.31	+	-	-
credit-a	13.82	14.13	14.69	-	-	-
credit-g	26.46	24.64	26.38	+	-	-
DNA	37.96	38.04	36.87	-	+	+
glass	29.99	29.43	29.39	+	+	+
hepatitis	15.82	14.39	16.78	-	-	-
house-votes-84	9.98	9.87	4.77	-	+	+
hypo	0.92	0.90	0.84	+	+	+
hypothyroid	1.42	1.45	1.36	-	+	+
ionosphere	10.09	9.66	9.17	+	+	+
iris	8.33	6.33	7.33	+	+	-
kr-vs-kp	12.55	12.49	5.91	+	+	+
labor	11.90	10.19	8.63	+	+	+
led-creator-+17	25.54	25.64	25.88	-	-	-
letter-recognition	26.35	25.2	18.65	+	+	+
nursery	9.66	9.79	8.17	-	+	+
pima-indians-diabetes	25.22	24.34	24.38	+	+	-
promoters-936	5.81	5.76	5.78	+	+	-
rbs	9.08	8.21	9.99	+	-	-
satellite	18.67	18.29	18.19	+	+	+
segmentation	7.04	6.92	7.19	+	-	-
sick	3.03	2.88	2.91	+	+	-
sick-euthyroid	3.88	3.88	3.82	+	+	+
sonar	22.21	21.34	21.32	+	+	+
soybean-large	7.69	7.26	6.48	+	+	+
splice	5.41	4.45	5.82	+	-	-
vehicle	39.23	40.22	38.54	-	+	+
waveform	20.0	20.06	20.07	-	-	-

Table 8: The table shows the base results obtained for the datasets using the Bagging and the Boosting techniques. The last three columns present the difference scores, between the Naive Bayes, Bagging and Boosting results, for analysis using the Sign Test.

view of the results obtained, by comparison with the results of the base classifier. The results shown were averaged over ten standard 10-fold cross-validation trials.

3.4.3 Analysis

The Naive Bayes algorithm is a very stable algorithm, and Bagging being a variance reduction technique did not significantly reduce the error rates. The bar-graph in Figure 3 gives a general idea of the performance of Bagging and Boosting on a few selected datasets. The Boosting results proved to be better than Bagging overall, but it degraded performance for 12 datasets (breast-cancer-wisconsin, cleveland-heart, credit-a, credit-g, hepatitis, iris, led-creator-+17, pima-indians-diabetes, rbs, segmentation, splice and waveform).

In general, Bagging produced results with slightly lower error rates than the results obtained for the single classifier except for the credit-a, DNA, hypothyroid, led-creator-+17 and nursery. Except for the agaricus-lepiota, iris and splice datasets, Bagging did not significantly decrease the error rates when compared to the single classifier. The Boosting results were more extreme. It performed significantly worse than Bagging for the following datasets: breast-cancer-wisconsin, cleveland-heart, credit-a, credit-g, hepatitis, led-creator-+17, rbs, splice and waveform. At the same time, Boosting did exceptionally well for agaricus-lepiota, house-votes-84, kr-vs-kp, labor and vehicle.

Applying the sign test again to the distribution of results, assuming again the probability of a type I error (α) to be .05. Formulating the decision rule, we have:

Decision Rule: Reject the hypothesis of the identity of the two distributions if the number of positive differences among the matched pairs is between $0, \dots 9$ and $21 \dots 30$ (called the critical region).

Single vs. Bagging: Since the number of positive differences computed is 21 out of a 30, this lies in the critical region and we reject the hypothesis of the identity between the two distributions. From this we conclude that the results obtained from the Naive Bayes classifier and the Bagging technique are not comparable to each other.

Single vs. Boosting: Since the number of positive differences computed is 21 out of a

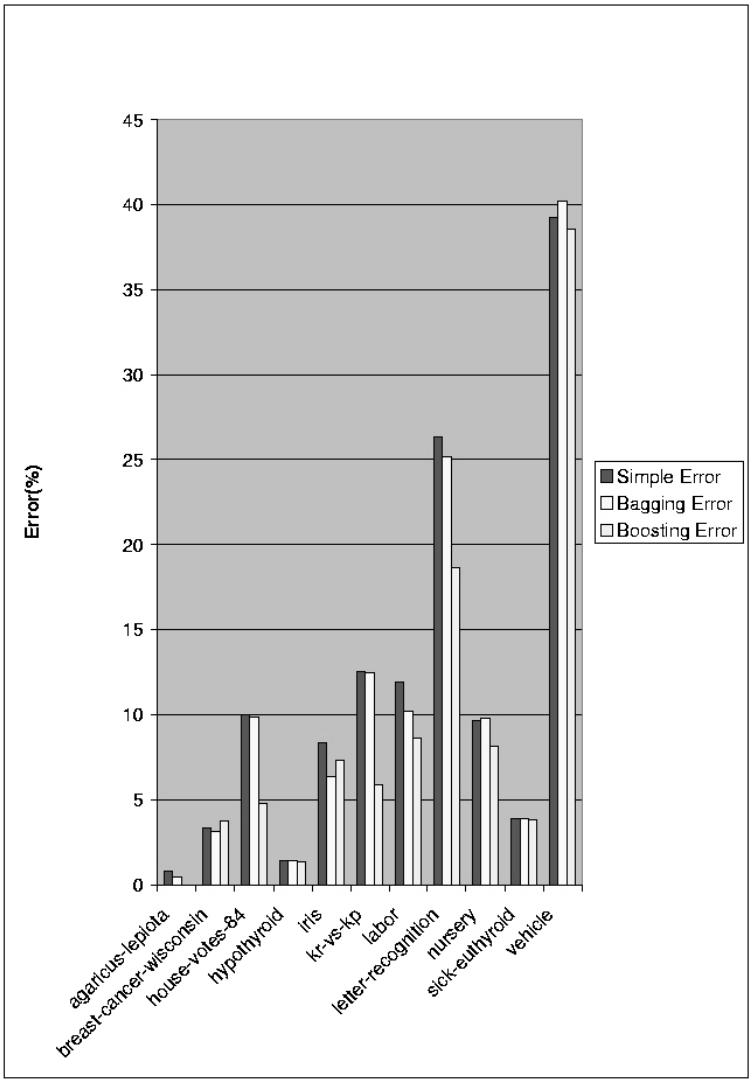

Figure 3: Datasets for which the errors for Bagging are uniformly lower than for Boosting.

30 this lies in the critical region and we reject the hypothesis of the identity between the two distributions. From this we conclude that the results obtained from the Naive Bayes classifier and the Boosting technique are not comparable to each other.

Bagging vs. Boosting: Since the number of positive differences computed is 16 out of a 30 this lies outside the critical region and we accept the hypothesis of the identity between the two distributions. From this we conclude that the results obtained from the Bagging technique and the Boosting technique are comparable to each other.

3.5 Advanced Experiments - Bagging using Training Samples

In this section, I present experiments that were conducted by using only sub-samples of data from the training set. This method has an obvious disadvantage of reducing accuracy of the learned concept when compared to methods that learn from the entire training data collection. In order to reduce this effect, I use the ensemble techniques of Bagging and Boosting. These techniques have certain advantages over single classifier learning in that they are capable of learning a concept in a more complete way. Hence in this section, unlike the experiments that were run in previous sections, training is done on sub-samples selected at random from the training set. Ensemble classifiers are described in detail in Sections 2.2, 2.3 and 2.4. Details on how the Bagging and the Boosting technique works is covered in Sections 2.5 and 2.6. The questions we are trying to answer in this section are:

Question 1: *Will the Sub-sample Bagging method yield accuracies that are comparable to the standard Bagging methods that use more resources and processing time?*

Question 2: *Can using the ensemble approach with Bagging overcome the disadvantages that come with using sub-sampling techniques?*

3.5.1 Methodology

Each classifier in the ensemble was built using only a random sub-sample of the dataset. The training samples used were of a fixed size containing 100, 200, 500 and 1000 samples. The Bagging technique was used to build the ensemble of classifiers (each with 25 component classifiers). The results were averaged over five standard 10-fold cross-validations.

3.5.2 Results

The results obtained for Bagging using sub-samples are shown in Figures 4 to 14 for the larger datasets (i.e., datasets with more than 3000 samples). For every dataset, the graph with plots representing the simple classifier results, the standard Bagging results and the sub-sample Bagging results are shown in the graph. The time taken (in minutes) for the experiments are shown in Table 9 for all the datasets. The first and second columns indicate the dataset name and sizes. The third column shows the time taken by the Bagging algorithm with a standard training set size to train an ensemble of size 25. The columns 3 to 7 show the time taken for the corresponding training set sizes shown in the appropriate columns for the same ensemble size of 25. The results were averaged over 5 standard 10-fold cross-validations. In order to enable uniformity of time comparisons, all of the following experiments were performed on Sun Ultra 1 machines.

The graphs shown in Figures 4 to 14 plot the error rates for Bagging Sub-samples of fixed sizes (100, 200, 500 and 1000). The plots for the base classifier result is indicated by a horizontal line marking the error rate. The standard Bagging plot is also included in every case.

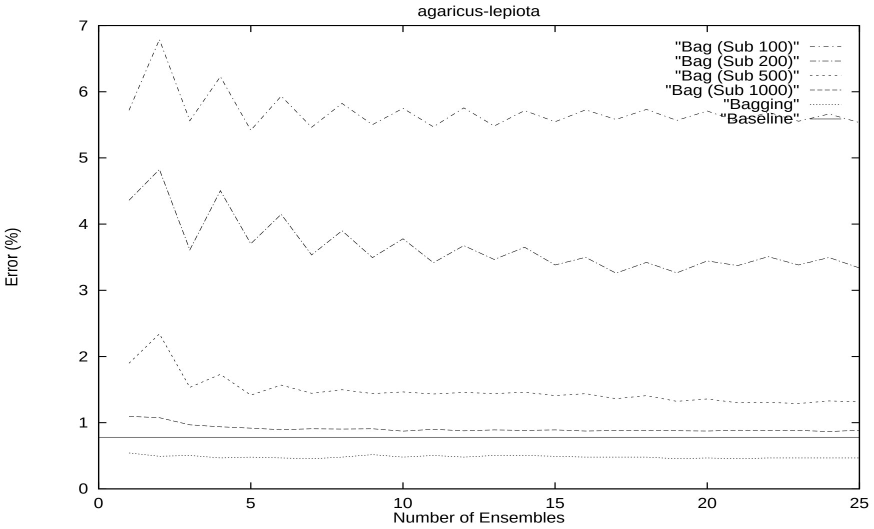

Figure 4: Graph for the agaricus-lepiota dataset showing the plots for Bagging Sub-samples using different fixed-sized datasets of 100, 200, 500 and 1000 instances. The standard Bagging plot is also shown. The plot for the single classifier results is shown as a plot independent of ensemble size.

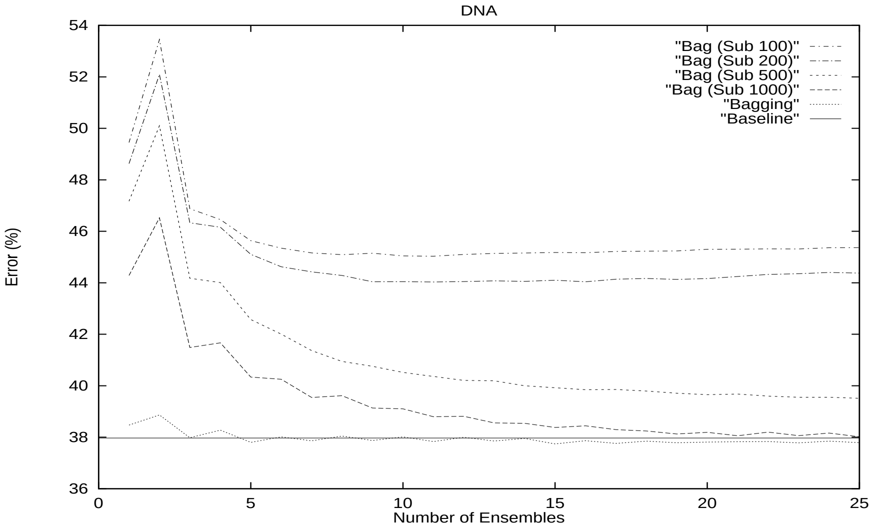

Figure 5: Graph for the DNA dataset showing the plots for Bagging Sub-samples similar to Figure 4.

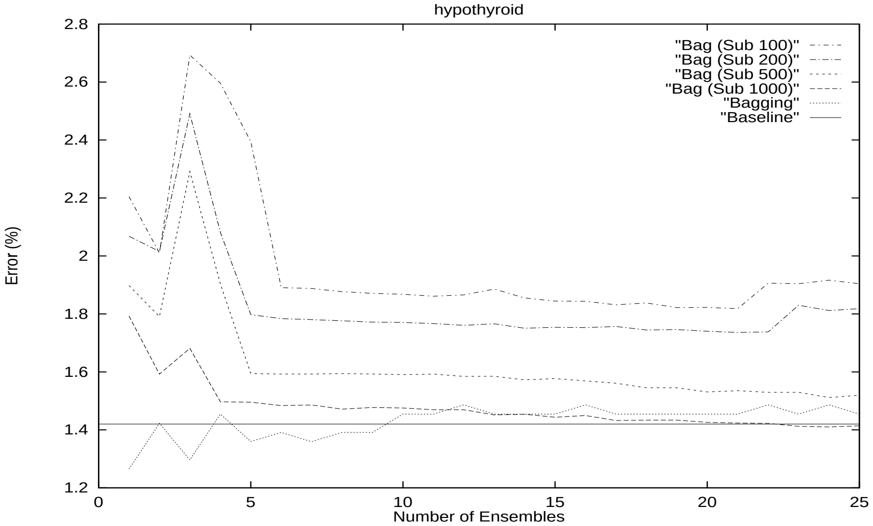

Figure 6: Graph for the hypothyroid dataset showing the plots for Bagging Sub-samples similar to Figure 4.

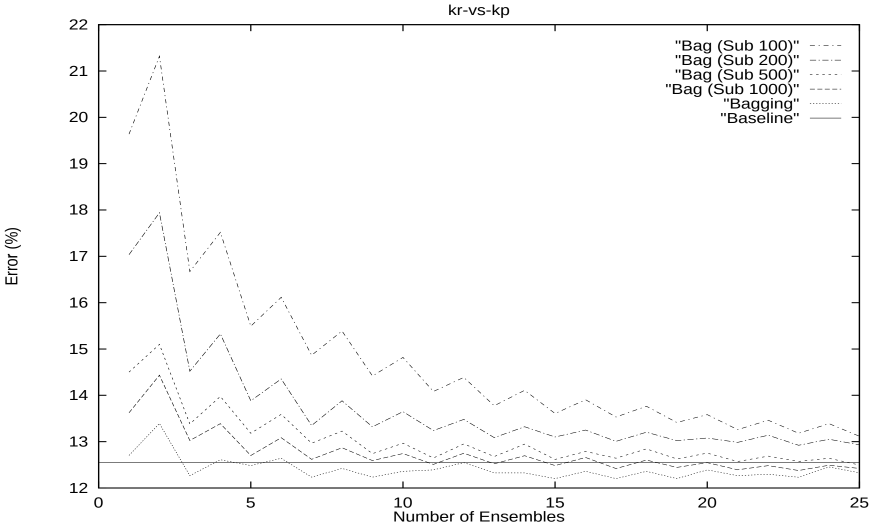

Figure 7: Graph for the kr-vs-kp dataset showing the plots for Bagging Sub-samples similar to Figure 4.

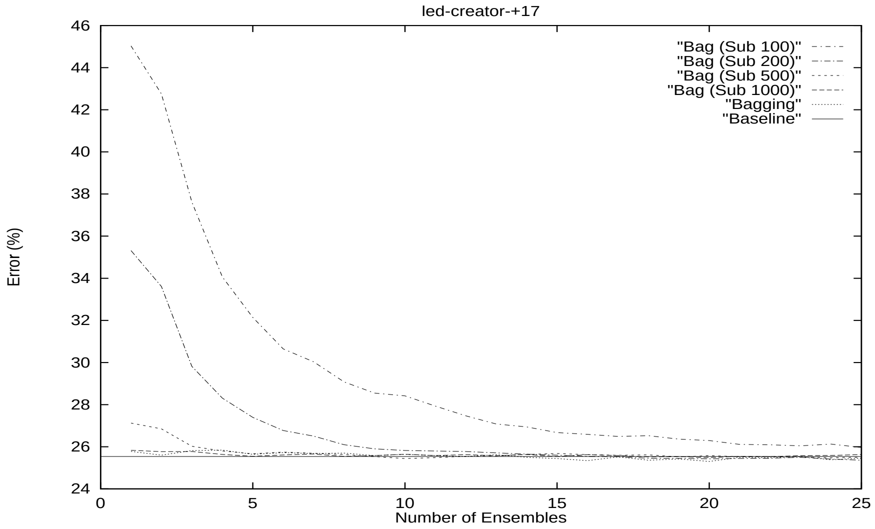

Figure 8: Graph for the led-creator-+17 dataset showing the plots for Bagging Sub-samples similar to Figure 4.

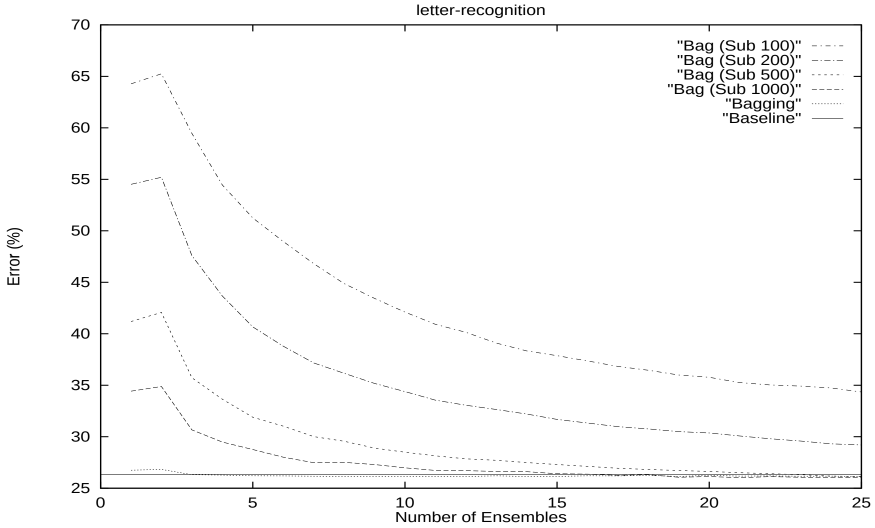

Figure 9: Graph for the letter-recognition dataset showing the plots for Bagging Sub-samples similar to Figure 4.

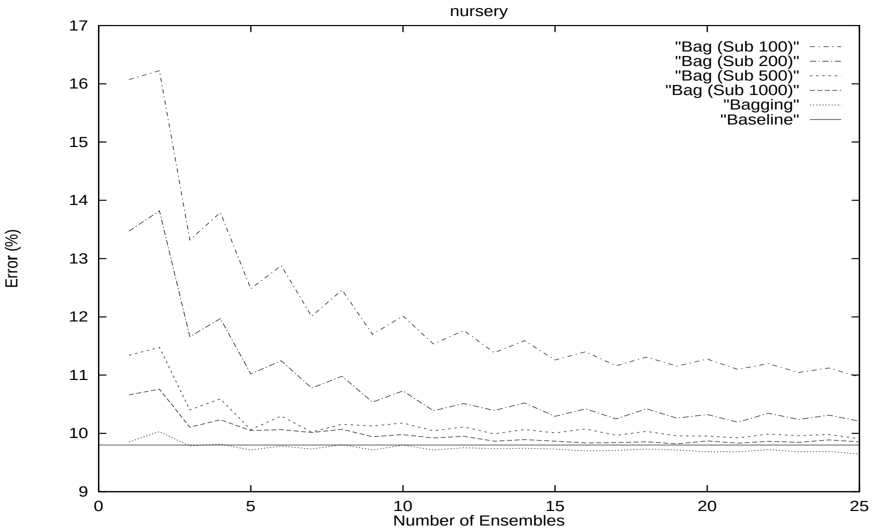

Figure 10: Graph for the nursery dataset showing the plots for Bagging Sub-samples similar to Figure 4.

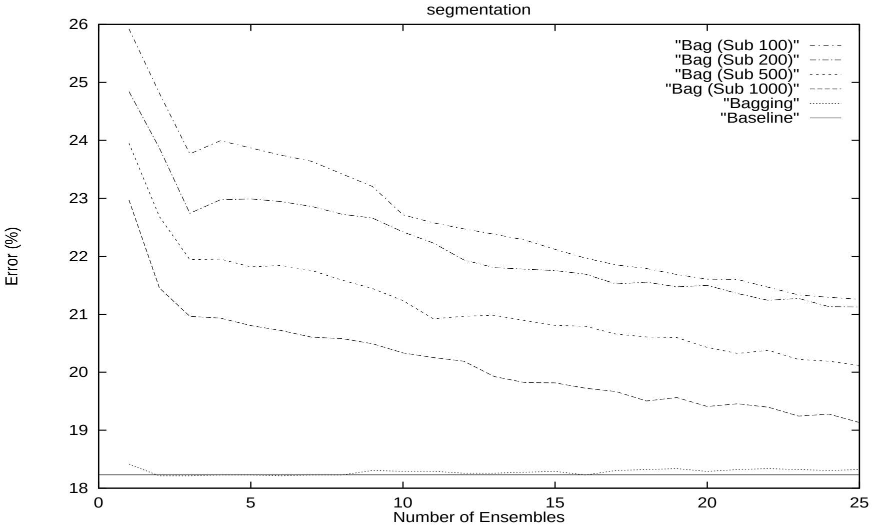

Figure 11: Graph for the satellite dataset showing the plots for Bagging Sub-samples similar to Figure 4.

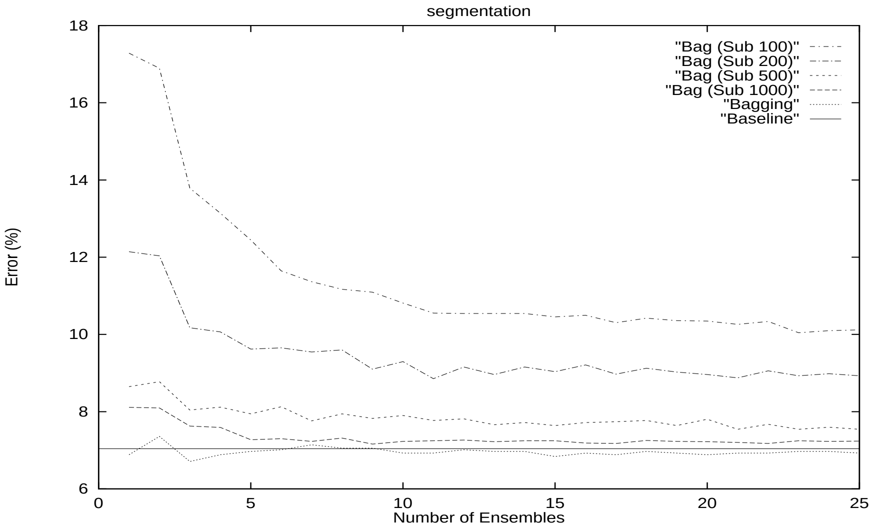

Figure 12: Graph for the segmentation dataset showing the plots for Bagging Sub-samples similar to Figure 4.

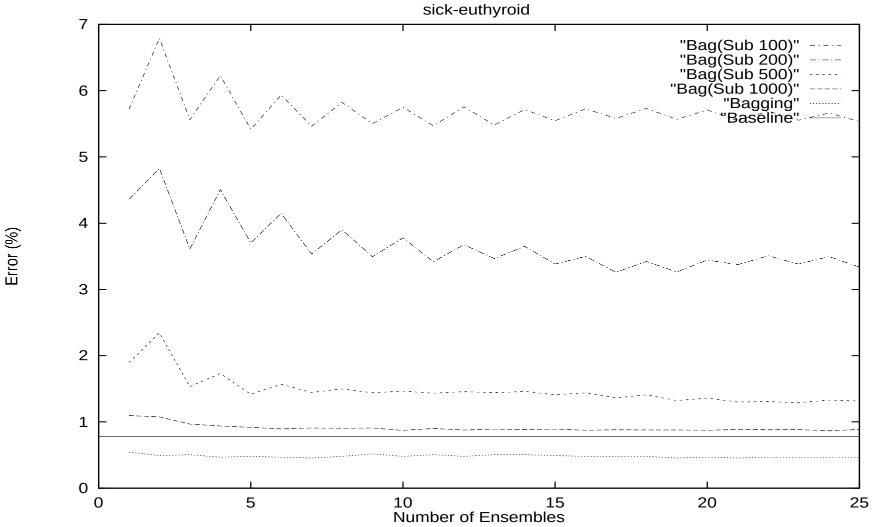

Figure 13: Graph for the sick-euthyroid dataset showing the plots for Bagging Sub-samples similar to Figure 4.

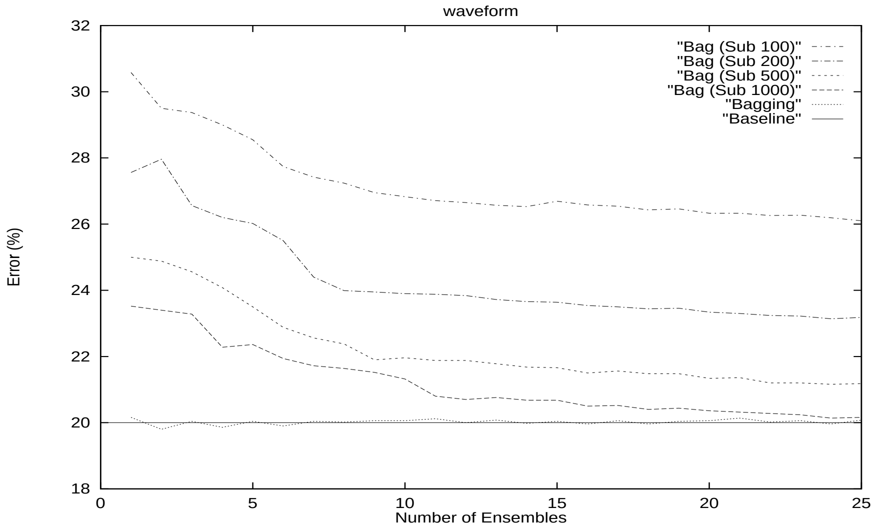

Figure 14: Graph for the waveform dataset showing the plots for Bagging Sub-samples similar to Figure 4.

Dataset	Dataset	**Time Taken(mins) with Time Reduction(%)**				
		Training Set Size				
	size	Standard	100	200	500	1000
agaricus-lepiota	8124	1423.87	21.16(98.51)	43.30(96.96)	90.87(93.62)	197.40(86.14)
DNA	21623	5701.60	33.33(99.42)	67.89(98.81)	144.78(97.46)	292.39(94.87)
hypothyroid	3163	354.89	16.06(95.47)	28.11(92.08)	64.67(81.77)	124.96(64.78)
kr-vs-kp	3196	612.17	18.58(96.96)	37.23(93.92)	100.45(83.59)	213.30(65.16)
led-creator-+17	5000	685.26	18.17(97.35)	34.82(94.91)	75.64(88.96)	152.28(77.78)
letter-recognition	20000	19594.26	134.32(99.31)	272.83(98.61)	560.52(97.14)	1088.57(94.44)
nursery	12960	5289.41	37.00(99.30)	79.34(98.50)	178.48(96.63)	367.32(93.06)
satellite	6435	2244.09	75.46(96.64)	109.28(95.13)	213.98(90.46)	387.48(82.73)
segmentation	2310	621.49	36.41(94.14)	74.38(88.03)	149.33(75.97)	298.94(51.90)
sick-euthyroid	3163	349.80	14.71(95.74)	26.58(92.40)	62.60(82.10)	122.88(64.87)
waveform	5000	2409.44	110.34(95.42)	227.54(90.56)	265.90(88.96)	535.43(77.78)

Table 9: The table shows the time taken by the Bagging algorithm for the different-sized training sub-samples chosen. The percentage reduction from the standard technique, is also shown in parenthesis, next to the time taken in minutes.

3.5.3 Analysis

It is clear from the graphs shown in Figures 4 to 14 that the sub-sample Bagging technique gives better (lower error rate) results as the number of component classifiers in the ensemble

Dataset	Standard time(mins)	Sub-sample size	Sub-sample time(mins)	Reduction in Time(%)
DNA	5701.60	1000	292.39	94.87
hypothyroid	354.89	1000	124.96	64.78
kr-vs-kp	612.17	500	100.45	83.59
led-creator-+17	685.26	200	34.82	94.91
letter-recognition	19594.26	500	560.52	97.14
nursery	5289.41	500	178.48	96.63
sick-euthyroid	349.80	1000	122.88	64.87

Table 10: The table shows the time taken by the Bagging algorithm for the different-sized training sub-samples chosen.

increases and as the training set increases. The graphs for almost all the datasets (except agaricus-lepiota and hypothyroid) show a drastic reduction in the error rates obtained for the lower number (approximately 0 to 10) of ensembles.

The graphs for the DNA, hypothyroid, kr-vs-kp, led-creator-+17, letter-recognition, nursery and sick-euthyroid datasets show a good performance especially with the availability of at least a 1000 training instances. The error rate is seen to rapidly decrease to the error rate obtained for the single classifier (in many cases the error rate of the Bagged classifier) as the ensemble size increases. The led-creator-+17, kr-vs-kp and the letter-recognition datasets show optimal performances for datasets of sizes 500 and 1000 with an ensemble size of 25. The led-creator-+17 dataset even shows good performance with a training set size of 200. Hence we see that for most domains it is in fact possible to obtain a good performance with a substantially smaller training set. Focusing on Table 9, we see that significant reduction in time for Bagging occurs because of the reduction in the training set size. For the DNA, hypothyroid, kr-vs-kp, led-creator-+17, letter-recognition, nursery and sick-euthyroid domains, the sub-sample error rates are comparable to that of the standard Bagging results and the simple classifier. This accuracy is achieved in a much shorter time. For example, the time reduction obtained for the above mentioned datasets is given in the Table 10. It is quite clear from the reduction in the time taken that the sub-sample technique does not compromise efficiency for time for most domains provided a sufficiently large training set along with a good number of ensembles is chosen.

3.6 Advanced Experiments - Boosting using Training Samples

In this section, I present results obtained from ensembles built from applying the Boosting technique to sub-samples of the training set. Details on how the Boosting technique works is covered in Section 2.6. The questions we are trying to answer in this section are:

Question 1: *Will the Sub-sample Boosting method yield accuracies that are comparable to the standard Boosting methods that use more resources and processing time?*

Question 2: *Can using the ensemble approach with Boosting overcome the disadvantages that come with using sub-sampling techniques?*

3.6.1 Methodology

Each classifier in the ensemble was built using a random sub-sample of the dataset. The training samples used were of a fixed size containing either a 100, 200, 500 or 1000 samples. The Boosting technique was used to build the ensemble of classifiers. Using 10-fold cross-validation, for every fold in the 10-fold cross validation, an ensemble of 25 classifiers was built (for a total of 250 classifiers for each 10-fold cross-validation). The results were averaged over five standard 10-fold cross-validations.

3.6.2 Results

The results obtained for Boosting using sub-samples are as shown in Figures 15 to 25, similar to the Bagging results. For every dataset, the graph with plots representing the simple classifier results, the standard Boosting results and the sub-sample Boosting results are shown in the graph. The times of the experiments are shown in Table 9 for all the datasets. The first and second columns indicate the dataset name and sizes. The third column shows the time taken by the standard Boosting algorithm to train an ensemble size of 25. The columns 3 to 7 show the time taken (in mins) for the different domains.

The graphs shown in Figures 15 to 25 plot the error rates for Boosting Sub-samples of fixed sizes (100, 200, 500 and 1000). The plots for the base classifier result is indicated by a horizontal line marking the error rate. The standard Boosting plot is also included in every case.

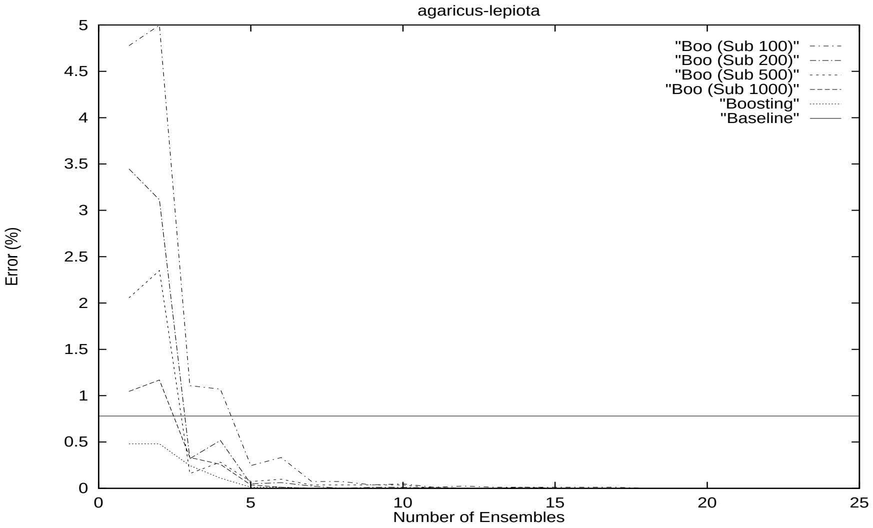

Figure 15: Graph for the agaricus-lepiota dataset showing the plots for Boosting Sub-samples using different fixed-sized datasets of 100, 200, 500 and 1000 instances. The standard Boosting plot is also shown. The plot for the single classifier results is shown as a plot independent of ensemble size.

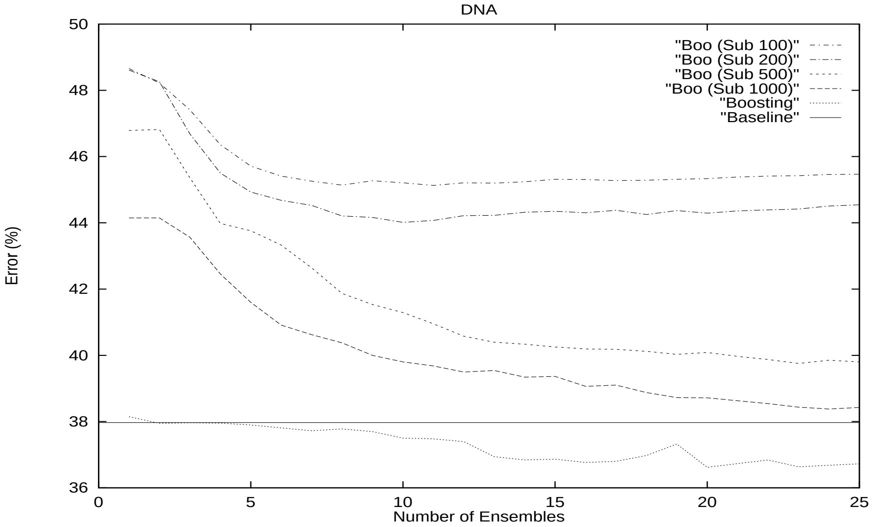

Figure 16: Graph for the DNA dataset showing the plots for Boosting Sub-samples similar to Figure 15.

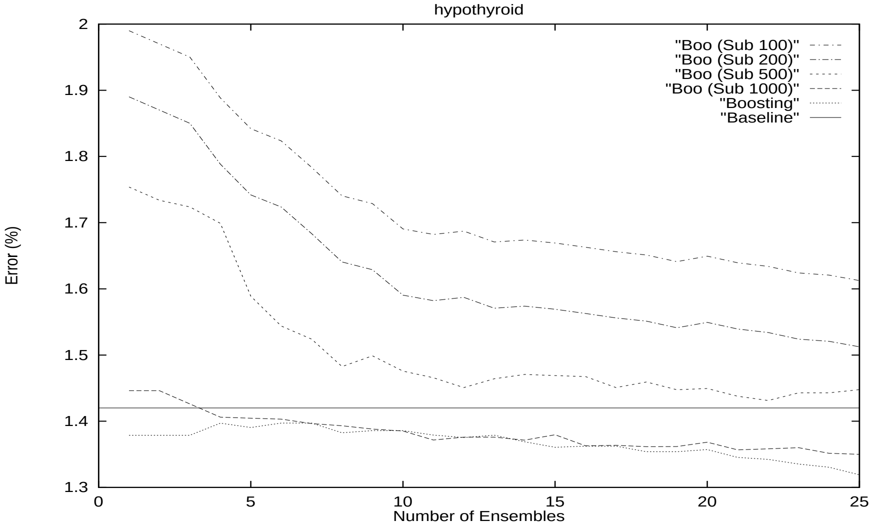

Figure 17: Graph for the hypothyroid dataset showing the plots for Boosting Sub-samples similar to Figure 15.

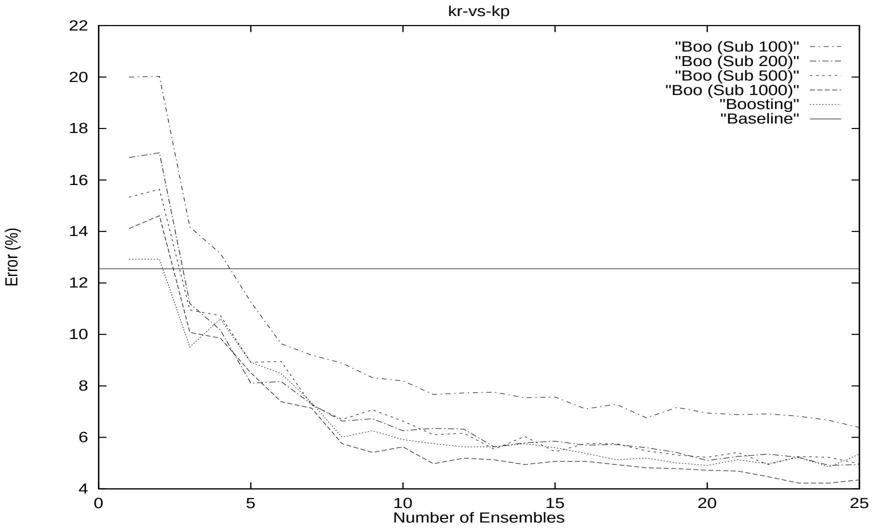

Figure 18: Graph for the kr-vs-kp dataset showing the plots for Boosting Sub-samples similar to Figure 15.

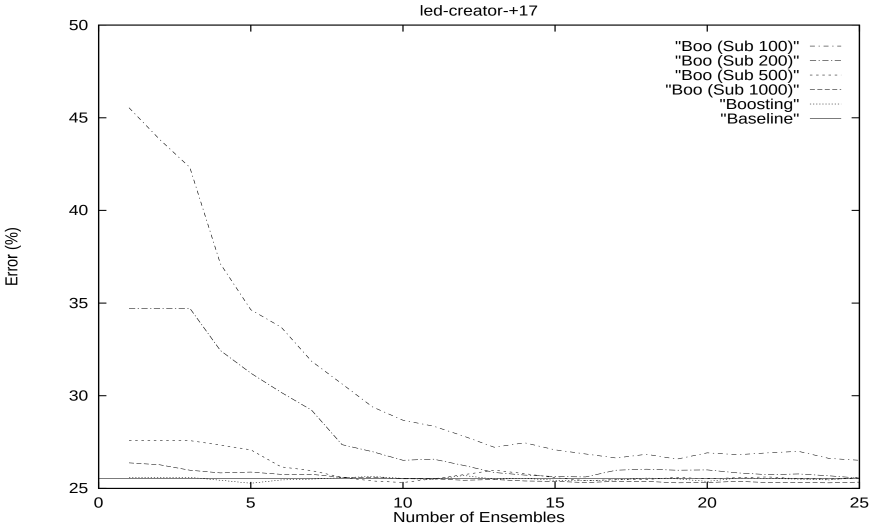

Figure 19: Graph for the led-creator-+17 dataset showing the plots for Boosting Sub-samples similar to Figure 15.

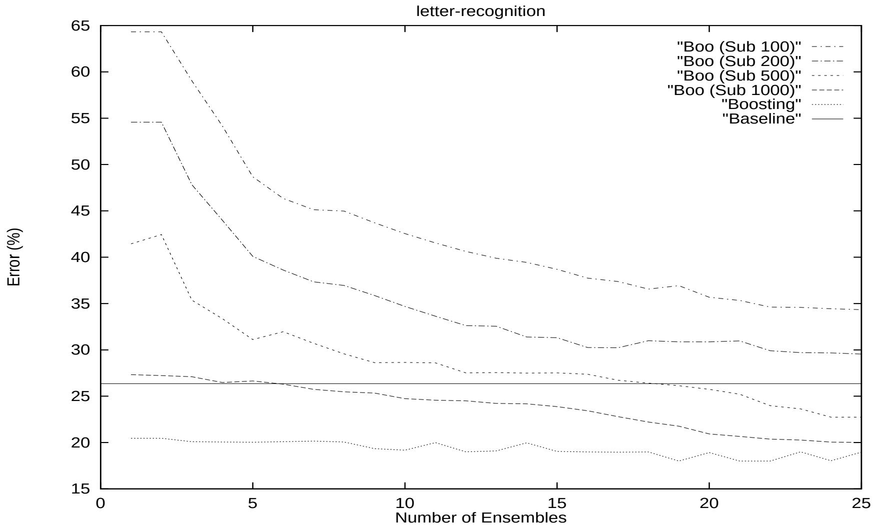

Figure 20: Graph for the letter-recognition dataset showing the plots for Boosting Sub-samples similar to Figure 15.

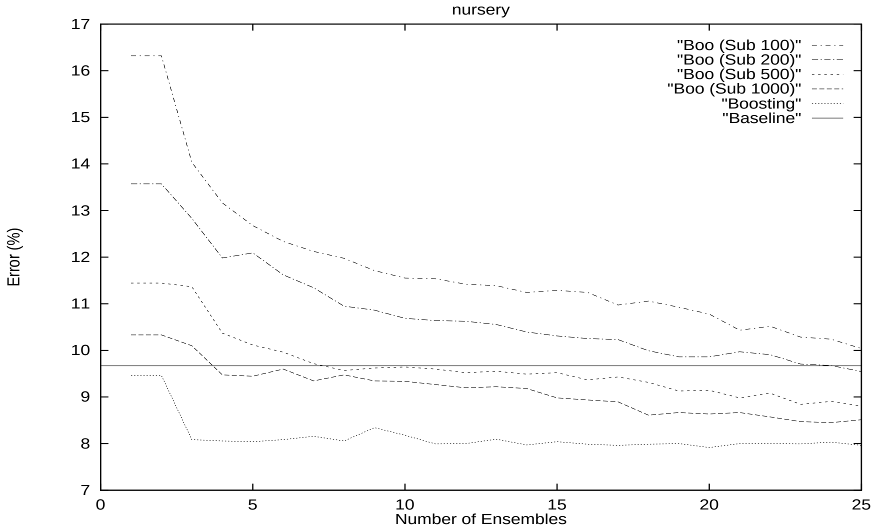

Figure 21: Graph for the nursery dataset showing the plots for Boosting Sub-samples similar to Figure 15.

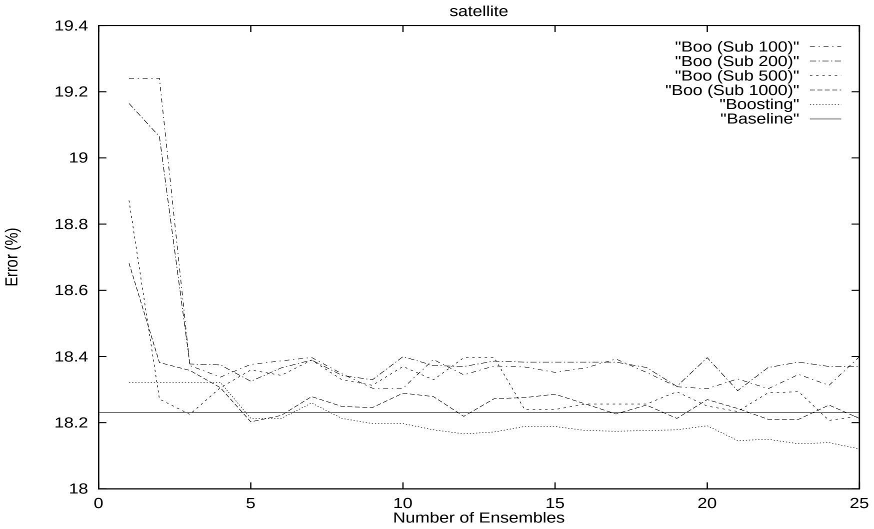

Figure 22: Graph for the satellite dataset showing the plots for Boosting Sub-samples similar to Figure 15.

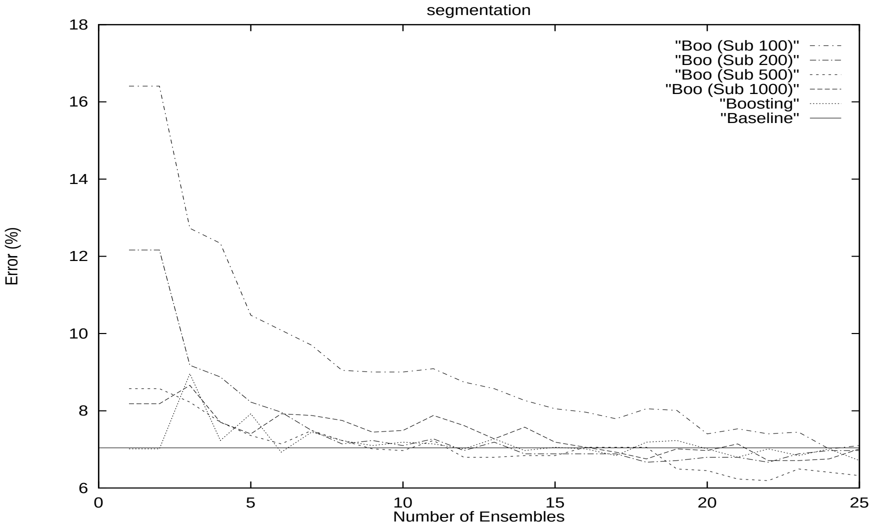

Figure 23: Graph for the segmentation dataset showing the plots for Boosting Sub-samples similar to Figure 15.

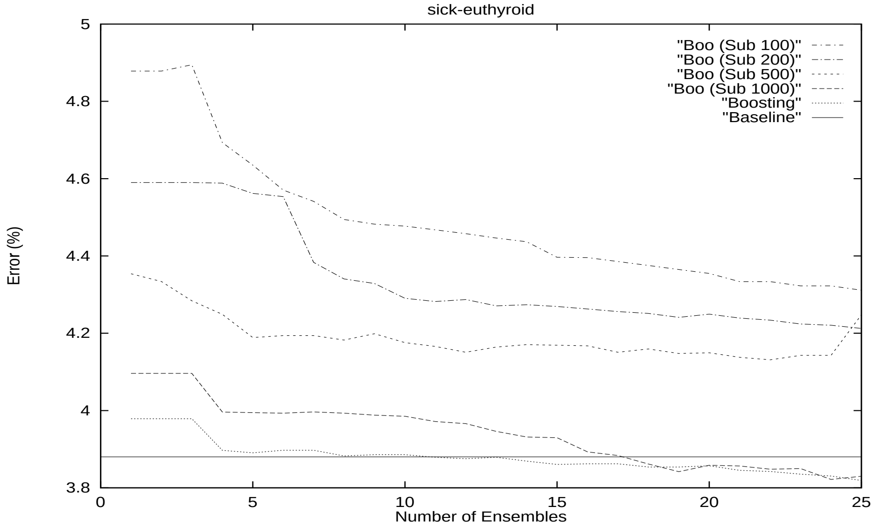

Figure 24: Graph for the sick-euthyroid dataset showing the plots for Boosting Sub-samples similar to Figure 15.

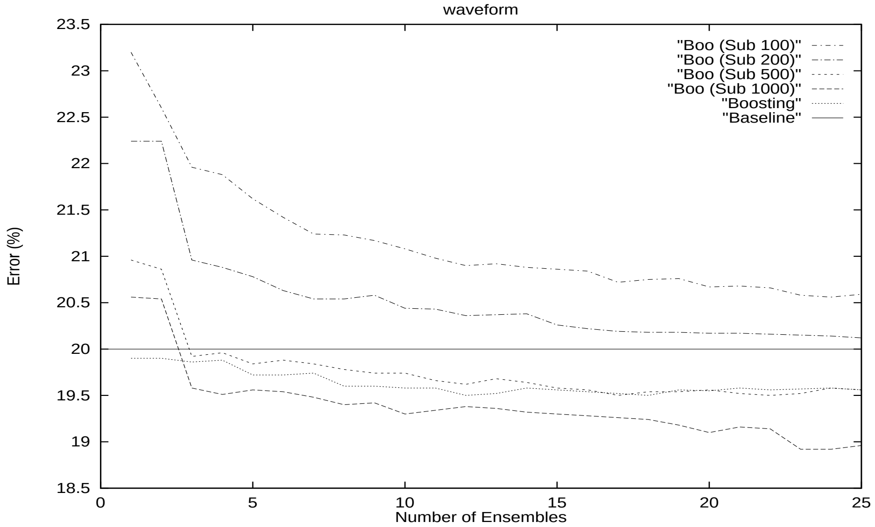

Figure 25: Graph for the waveform dataset showing the plots for Boosting Sub-samples similar to Figure 15.

Dataset	Dataset	**Time Taken(mins) with Time Reduction(%)**				
		Training Set Size				
	size	Standard	100	200	500	1000
agaricus-lepiota	8124	7538.25	106.90(98.51)	208.38(96.96)	516.88(93.62)	1031.92(86.31)
DNA	21623	36813.23	354.65(99.42)	586.72(98.81)	765.03(97.46)	1891.67(94.86)
hypothyroid	3163	1766.54	83.87(95.47)	172.42(92.08)	303.38(81.77)	620.57(64.87)
kr-vs-kp	3196	953.95	37.55(96.96)	70.87(93.92)	160.87(83.59)	331.65(65.23)
led-creator-+17	5000	4601.93	141.95(97.35)	207.80(94.92)	516.87(88.96)	1023.54(77.78)
letter-recognition	20000	84661.36	749.54(99.31)	1441.32(98.61)	2951.27(97.14)	5814.52(93.13)
nursery	12960	20510.15	212.87(99.30)	421.36(98.50)	1050.77(96.63)	2101.35(89.75)
satellite	6435	20974.32	479.73(96.64)	910.85(95.13)	2226.47(90.46)	4398.57(79.03)
segmentation	2310	1893.72	279.23(94.14)	331.73(88.03)	560.37(75.97)	910.88(51.90)
sick-euthyroid	3163	5116.74	184.06(95.79)	372.53(92.40)	905.25(82.10)	1797.43(64.87)
waveform	5000	15271.38	509.76(95.42)	1912.90(90.56)	1754.53(88.96)	3392.64(77.78)

Table 11: The table shows the time taken by the Boosting algorithm for the different-sized training sub-samples chosen. The percentage reduction from the standard technique is also shown in parenthesis next to the time taken in minutes.

3.6.3 Analysis

As is clear from the graphs in Figures 15 to 25, the experiments with Boosting prove to be less uniform and more interesting than Bagging in general. The error-rates overall are lower than those obtained for Sub-sample Bagging. For the agaricus-lepiota, hypothyroid, kr-vs-kp, led-creator-+17, letter-recognition, nursery, satellite, segmentation and the waveform datasets, the Sub-sample method seems to work well with a substantial number of ensembles. It is interesting to note that in the case of the agaricus-lepiota dataset, the error reaches an optimum with only a small number of ensembles (5 - 10) with and a training set of only 100 instances.

The segmentation and the led-creator-+17 datasets show that with only 15 ensembles and training sizes of 200 or more, the sub-sample technique gives results that are very similar to those obtained from standard Boosting techniques. The graph for the kr-vs-kp dataset shows error rates that are definitely lower than the single classifier with ensem-

Dataset	Standard time(mins)	Sub-sample size	Sub-sample time(mins)	Reduction in Time(%)
agaricus-lepiota	1423.87	100	21.16	98.51
kr-vs-kp	612.17	500	100.45	83.59
led-creator-+17	685.26	500	75.64	88.96
segmentation	621.49	200	74.38	88.03
sick-euthyroid	349.80	1000	122.88	64.87
waveform	2409.44	500	265.90	88.96

Table 12: The table shows the time taken by the Boosting algorithm for the different-sized training sub-samples chosen.

ble sizes approximately greater than 5, regardless of the training set size. The plots for the DNA, letter-recognition and the nursery datasets show the error rates obtained for the sub-sampling technique is comparable to but not as low as those obtained from standard techniques.

For the domains where Sub-sample Boosting works well, the error seems to increase considerably when the ensemble size is increased, unlike training size which does not seem to affect the error rate as much. This can be observed from the graphs for agaricus-lepiota, kr-vs-kp, led-creator-+17, satellite and segmentation datasets. Hence for most domains, where the Sub-sample Boosting technique gives good results, it would not make much of a difference if one were to increase the training set size. Table 12 shows the percentage in time reduction possible using Sub-sampling technique over the standard method for the domains in which the former performs better. From the graphs in Figures 15 to 25 and the time reduction estimates in Table 12 we can see that the Sub-sample Boosting technique would be a profitable method to use, in terms of time, for the following domains: agaricus-lepiota, kr-vs-kp, led-creator-+17, segmentation, sick-euthyroid and waveform. Comparing the Sub-sample Bagging results to the Sub-sample Boosting results we see that the Sub-sample Boosting technique produces results that are more unstable.

4 Conclusions

In this thesis, I implement two different ensemble building techniques and evaluate them with respect to learning from samples of training data. Experiments on these techniques indicate that the questions put forth earlier have been answered with respect to the different domains used.

In the preliminary tests, I first determined the baseline results with which to compare the results of the newer technique. The results showed that the Naive Bayes classifier produces results that are comparable to C4.5 in performance.

Also included in the preliminary tests are the Bagging and the Boosting experiments with the Naive Bayes classifier. The results show that Bagging is more consistent in giving lower error-rates than the Boosting algorithm when compared with the base classifier. I then tested these two ensemble techniques with sub-samples of the training data.

The Sub-sampling techniques were implemented using fixed training set sizes of a 100, 200, 500 and 1000 instances. The Bagging technique seemed to perform in a stable way for most domains, with the error rates gradually decreasing as the training set and the number of ensembles increased. It was possible to obtain a good performance with a 1000 training instances in many domains. It was seen that the new technique of sub-sampling did in fact work quite well, given sufficient training data and ensembles. The error seemed to decrease correspondingly when the training set size was increased. The same behavior was seen when the number of ensembles was increased. This shows that a larger training set is definitely useful, but at the same time it may be possible to obtain the same accuracy with a smaller training set using a larger number of ensembles. As the time statistics show, this would greatly reduce the learning time needed. Results show that it is possible to reduce the time taken to obtain the base results (from the earlier experiments) with the standard techniques.

The Boosting technique, on the other hand, gave non-uniform results. The experiments suggested that this technique could reduce the error rates considerably for some domains, implying the advantage of speed that the sub-sampling method promises in general. Boosting with sub-samples produced a drastic reduction in the error rates for some domains when compared with the results obtained for Bagging with sub-samples. The inherent extreme nature of Boosting was seen to exist even when sub-samples of training data were used. The Boosting technique, when it worked, did really well at reducing the error rates drastically even with a training set size of a 100 and a few ensembles (5 -10). But this behavior was not consistent for all domains. Furthermore, reduction in the error rates was very small for corresponding increases in the training set size and the number of ensembles for these domains. Another interesting observation was that increasing the training set size did not play a great part in lowering the error rates for a few datasets. Overall, the Boosting method with sub-samples produced correspondingly better results as the number of ensembles increased.

The main contribution of this thesis is to investigate the problem of overcoming the disadvantage of using sub-samples of training data by making use of ensemble techniques. The strategy proved to work quite well for most domains (7 out of a 11 domains) with the Bagging technique reducing the time required for the learning task. It proved effective for a good number of datasets (6 out of a 11 domains) with Boosting.

The questions from Chapter 1 that proved as a chief motivation behind this thesis are restated here along with the conclusions that could be drawn about answering them in the context of the experiments conducted:

Question 1: *Can a model that has been trained on samples taken from the collection of training data perform as well as the model that has been trained using the entire training data collection?*

From the results of the experiments we see that the new techniques developed indicate that a model trained from sub-samples of the training data could perform as well as the model that has been trained using the entire data collection depending on the domain and the learning technique being used.

Question 2: *Will this method yield accuracies that are comparable to the methods that use more resources and processing time?*

The accuracies obtained for most of the datasets indicate that the sub-sample methods are not only comparable to the standard techniques but are also competitive in terms of using lesser time and resources.

Question 3: *Can using the ensemble approach overcome the disadvantages that come with using sub-sampling techniques?*

The sub-sampling technique comes with the obvious disadvantage of reducing the accuracy of the model being built when compared to the standard model. As the results show, the ensemble method seems to perform well enough to give good results despite this fact. In most cases, provided a sufficiently large training set and ensembles are available this method proves competitive to the standard learning techniques.

This claim that induction from large datasets is now manageable or practicable is made from a theoretical point of view, by examining abstract requirements and trends. But a specific concrete application may present practical difficulties that should not be left understated. The dataset may be too large to move or copy from the machine where it resides. In extreme cases, so many training instances may be available that performing even the simplest computation on them all would be too expensive. In such cases, sub-sampling on a relatively small training set may become mandatory, which might cause reduction in the accuracy obtained.

The principle conclusions of this thesis are that induction from very large datasets is desirable and can be manageable with the techniques presented here. Gains in accuracy are available in some domains with the use of only sub-samples of the training data available. Learning can be accelerated using ensemble techniques. These techniques may reduce the learning time to a manageable level for large datasets.

In this work, I have addressed the problems that come with using sub-samples for learning. We see from the results that the disadvantage of using sub-samples, i.e., the problem of reduction in accuracy, is overcome by using ensemble techniques in most cases. The claim made above can be justified by studying the experiments and the results.

References

[Ali and Pazzani, 1996] Ali, K. M. and Pazzani, M. J. (1996). Error reduction through learning multiple descriptions. In *Machine Learning*, volume 24(3), pages 173–202.

[Alpaydin, 1993] Alpaydin, E. (1993). Mulitple networks for function learning. In *Proceedings of the IEEE International Conference on Neural Networks*, pages 27–32, San Fransisco, CA.

[Asker and Maclin, 1997] Asker, L. and Maclin, R. (1997). Ensembles as a sequence of classifiers. In *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence*, Nagoya, Japan.

[Becker et al., 1997a] Becker, B., Kohavi, R., and Sommerfield, D. (1997a). Improving simple bayes. In *The 9th European Conference on Machine Learning, Poster Paper*, pages 78–87.

[Becker et al., 1997b] Becker, B., Kohavi, R., and Sommerfield, D. (1997b). Visualizing the simple bayesian classifier. In *KDD Workshop on Issues in the Integration of Data Mining and Data Visualization.*

[Breiman, 1989] Breiman, L. (1989). Tech. Report. volume 367. (unpublished).

[Breiman, 1996] Breiman, L. (1996). Bagging Predictors. ML. volume 24(2), pages 123–140.

[Catlett, 1991] Catlett, J. (1991). *Megainduction: Machine Learning on large databases.* PhD thesis, University of Sydney, Sydney, Australia.

[Cestnik, 1990] Cestnik, B. (1990). Estimating probabilities: A crucial task in machine learning. In *Proceedings of the ninth European Confernce on Artificial Intelligence*, pages 147–149.

[Cestnik et al., 1986] Cestnik, B., Knonenko, I., and Bratko, I. (1986). A knowledge-elicitation tool for sphisticated users. In *Progress in Machine Learning - Proceedings of the Second European Working Session on Learning*, pages 31–45, Wilmslow, UK. Sigma Press.

[Clemen, 1989] Clemen, R. (1989). Combining forecasts: A review and annotated bibliography. In *International Journal of Forecasting*, volume 5, pages 559–583.

[Dietterich, 1998] Dietterich, T. G. (1998). An Experimental Comparison of Three Methods for Constucting Ensmebles of Decision Trees: Bagging, Boosting, and Randomization. In *Machine Learning*, pages 1–22.

[Domingos and Pazzani, 1997] Domingos, P. and Pazzani, M. (1997). Beyond independence: Conditons for the optimality of a simple Bayesian classifier. In *Proceedings of the Thirteenth International Confernece on Machine Learning*, pages 105–112, San Fransisco, CA. Morgan Kaufmann.

[Dougherty et al., 1995] Dougherty, J., Kohavi, R., and Sahami, M. (1995). Supervised and unsupervised discretization of continuous features. In *Machine Learning: Proceedings of the Twelfth International Conference*, pages 194–202. Morgan Kaufmann Publishers.

[Drucker et al., 1994] Drucker, H., Cortes, C., Jackel, L., LeCun, Y., and Vapnik, V. (1994). Boosting and other machine learning algorithms. In *Proceedings of the Eleventh International Conference on Machine Learning*, pages 53–61.

[Duda and Hart, 1973] Duda, R. O. and Hart, P. E. (1973). Pattern Classification and Scene Analysis. New York. John Wiley.

[Fayyad and Irani, 1993] Fayyad, U. M. and Irani, K. B. (1993). Multi-interval discretization of continuous valued attributes for classification learning. In *Proceedings of the 13th International Joint Conference on Artificial Intelligence*. Morgan Kaufmann Publishers.

[Freund and Schapire, 1996] Freund, Y. and Schapire, R. E. (1996). Experiments with a new boosting algorithm. In *Machine Learning: Proceedings of the Thirteenth International Conference*, pages 325–332, San Fransisco, CA. Morgan Kaufmann.

[Good, 1965] Good, I. J. (1965). The Estimation of Probability: An Essay on Modern Bayesian Methods. M.I.T Press.

[Granger, 1989] Granger, C. W. J. (1989). Journal of Forecasting. volume 8, page 176.

[Hansen and Salamon, 1990] Hansen, L. K. and Salamon, P. (1990). IEEE Transactions of Pattern Analysis and Machine Intelligence. volume 12, page 993.

[Kohavi and Sahami, 1996] Kohavi, R. and Sahami, M. (1996). Error-based and entropy-based discretization of continuous features. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, pages 114–119.

[Kohavi and Sommerfield, 1995] Kohavi, R. and Sommerfield, D. (1995). Feature subset selection using wrapper model: Overfitting and dynamic search space topology. In *The First International Conference on Knowledge Discovery and Data Mining*, pages 192–197.

[Krogh and Vedelsby, 1995] Krogh, A. and Vedelsby, J. (1995). Advances in Neural Information Processing Systems. pages 231–238, Cambridge, MA. MIT Press.

[Langely et al., 1992] Langely, P., Iba, W. F., and Thompson, K. (1992). An analysis of Bayesian classifiers. In *Proceedings of the Tenth National Conference on Artificial Intelligence*, pages 223–228. AAAI Press.

[Langely and Sage, 1997] Langely, P. and Sage, S. (1997). Scaling to domains with many irrelevant features. In *Computational Learning Theory and Natural Learning Systems*, Cambirdge, MA. M.I.T Press.

[Lincoln and Skrzypek, 1989] Lincoln, W. and Skrzypek, J. (1989). Synergy of clustering multiple back propogation networks. In Touretzky, D., editor, *Advances in Neural Information Processing Systems*, volume 2, pages 650–659, San Mateo, CA.

[Maclin and Opitz, 1997] Maclin, R. and Opitz, D. (1997). An empirical evaluations of bagging and boostings. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence*, pages 546–551, Cambridge, MA.

[Maclin and Shavlik, 1995] Maclin, R. and Shavlik, J. (1995). Combining the predictions of multiple classifiers: Using competitive learning to initialize neural networks. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, pages 524–530, Montreal, Canada.

[Opitz and Shavlik, 1996] Opitz, D. and Shavlik, J. (1996). Generating accurate and diverse members of a neural network ensemble. In *Advances in Neural Information Processing Systems*, volume 8, pages 534–541, Cambridge, MA. MIT Press.

[Perrone, 1993] Perrone, M. (1993). *Improving Pregression Estimation: Averaging Methods for Variance Reduction with Extension to General Convex Measure Optimization.* PhD thesis, Brown University, Providence, RI.

[Quinlan, 1996] Quinlan, J. R. (1996). Bagging, Boosting and C4.5. In *In Proccedings of the Thirteeth National Conference on AI.*

[Rogova, 1994] Rogova, G. (1994). Combining the results of several neural-network classifiers. In *Neural Networks*, volume 7, pages 777–781.

[Wolpert, 1992a] Wolpert, D. H. (1992a). Neural Networks. volume 5, page 241.

[Wolpert, 1992b] Wolpert, D. H. (1992b). Stacked Generalization, Neural Networks,. volume 5, pages 241–259.

[Zhang et al., 1992] Zhang, X., Mesirov, J., and Waltz, D. (1992). Hybrid system for protein secondary structure prediction. In *Journal of Molecular Biology*, volume 225, pages 1049–1063.