Department of Computer Science

The
University
of Waikato
Te Whare Wānanga
o Waikato

Hamilton, NewZealand

Correlation-based Feature Selection for Machine Learning

Mark A. Hall

This thesis is submitted in partial fulfilment of the requirements
for the degree of Doctor of Philosophy at The University of Waikato.

April 1999

Abstract

A central problem in machine learning is identifying a representative set of features from which to construct a classification model for a particular task. This thesis addresses the problem of feature selection for machine learning through a correlation based approach. The central hypothesis is that good feature sets contain features that are highly correlated with the class, yet uncorrelated with each other. A feature evaluation formula, based on ideas from test theory, provides an operational definition of this hypothesis. CFS (Correlation based Feature Selection) is an algorithm that couples this evaluation formula with an appropriate correlation measure and a heuristic search strategy.

CFS was evaluated by experiments on artificial and natural datasets. Three machine learning algorithms were used: C4.5 (a decision tree learner), IB1 (an instance based learner), and naive Bayes. Experiments on artificial datasets showed that CFS quickly identifies and screens irrelevant, redundant, and noisy features, and identifies relevant features as long as their relevance does not strongly depend on other features. On natural domains, CFS typically eliminated well over half the features. In most cases, classification accuracy using the reduced feature set equaled or bettered accuracy using the complete feature set. Feature selection degraded machine learning performance in cases where some features were eliminated which were highly predictive of very small areas of the instance space.

Further experiments compared CFS with a wrapper—a well known approach to feature selection that employs the target learning algorithm to evaluate feature sets. In many cases CFS gave comparable results to the wrapper, and in general, outperformed the wrapper on small datasets. CFS executes many times faster than the wrapper, which allows it to scale to larger datasets.

Two methods of extending CFS to handle feature interaction are presented and experimentally evaluated. The first considers pairs of features and the second incorporates feature weights calculated by the RELIEF algorithm. Experiments on artificial domains showed that both methods were able to identify interacting features. On natural domains, the pairwise method gave more reliable results than using weights provided by RELIEF.

Acknowledgements

First and foremost I would like to acknowledge the tireless and prompt help of my supervisor, Lloyd Smith. Lloyd has always allowed me complete freedom to define and explore my own directions in research. While this proved difficult and somewhat bewildering to begin with, I have come to appreciate the wisdom of his way—it encouraged me to think for myself, something that is unfortunately all to easy to avoid as an undergraduate.

Lloyd and the Department of Computer Science have provided me with much appreciated financial support during my degree. They have kindly provided teaching assistantship positions and travel funds to attend conferences.

I thank Geoff Holmes, Ian Witten and Bill Teahan for providing valuable feedback and reading parts of this thesis. Stuart Inglis (super-combo!), Len Trigg, and Eibe Frank deserve thanks for their technical assistance and helpful comments. Len convinced me (rather emphatically) *not* to use MS Word for writing a thesis. Thanks go to Richard Littin and David McWha for kindly providing the University of Waikato thesis style and assistance with LaTeX.

Special thanks must also go to my family and my partner Bernadette. They have provided unconditional support and encouragement through both the highs and lows of my time in graduate school.

Contents

Appendices

List of Figures

List of Tables

Chapter 1

Introduction

We live in the information-age—accumulating data is easy and storing it inexpensive. In 1991 it was alleged that the amount of stored information doubles every twenty months [PSF91]. Unfortunately, as the amount of machine readable information increases, the ability to understand and make use of it does not keep pace with its growth. Machine learning provides tools by which large quantities of data can be automatically analyzed. Fundamental to machine learning is *feature selection*. Feature selection, by identifying the most salient features for learning, focuses a learning algorithm on those aspects of the data most useful for analysis and future prediction. The hypothesis explored in this thesis is that feature selection for supervised classification tasks can be accomplished on the basis of correlation between features, and that such a feature selection process can be beneficial to a variety of common machine learning algorithms. A technique for correlation-based feature selection, based on ideas from test theory, is developed and evaluated using common machine learning algorithms on a variety of natural and artificial problems. The feature selector is simple and fast to execute. It eliminates irrelevant and redundant data and, in many cases, improves the performance of learning algorithms. The technique also produces results comparable with a state of the art feature selector from the literature, but requires much less computation.

1.1 Motivation

Machine learning is the study of algorithms that automatically improve their performance with experience. At the heart of performance is prediction. An algorithm that—when presented with data that exemplifies a task—improves its ability to predict key elements of the task can be said to have *learned.* Machine learning algorithms can be broadly characterized by the language used to represent learned knowledge. Research has shown that no single learning approach is clearly superior in all cases, and in fact, different learning algorithms often produce similar results [LS95]. One factor that can have an enormous impact on the success of a learning algorithm is the nature of the data used to characterize the task to be learned. If the data fails to exhibit the statistical regularity that machine learning algorithms exploit, then learning will fail. It is possible that new data may be constructed from the old in such a way as to exhibit statistical regularity and facilitate learning, but the complexity of this task is such that a fully automatic method is intractable [Tho92].

If, however, the data is suitable for machine learning, then the task of discovering regularities can be made easier and less time consuming by removing features of the data that are *irrelevant* or *redundant* with respect to the task to be learned. This process is called *feature selection.* Unlike the process of constructing new input data, feature selection is well defined and has the potential to be a fully automatic, computationally tractable process. The benefits of feature selection for learning can include a reduction in the amount of data needed to achieve learning, improved predictive accuracy, learned knowledge that is more compact and easily understood, and reduced execution time. The last two factors are of particular importance in the area of commercial and industrial *data mining.* Data mining is a term coined to describe the process of sifting through large databases for interesting patterns and relationships. With the declining cost of disk storage, the size of many corporate and industrial databases have grown to the point where analysis by anything but parallelized machine learning algorithms running on special parallel hardware is infeasible [JL96]. Two approaches that enable standard machine learning algorithms to be applied to large databases are feature selection and sampling. Both reduce the size of the database—feature selection by identifying the most salient *features* in the data; sampling by identifying representative *examples* [JL96]. This thesis focuses on feature selection—a process that can benefit learning algorithms regardless of the amount of data available to learn from.

Existing feature selection methods for machine learning typically fall into two broad categories—those which evaluate the worth of features using the learning algorithm that is to ultimately be applied to the data, and those which evaluate the worth of features by using heuristics based on general characteristics of the data. The former are referred to as *wrappers* and the latter *filters* [Koh95b, KJ96]. Within both categories, algorithms can be further differentiated by the exact nature of their evaluation function, and by how the space of feature subsets is explored.

Wrappers often give better results (in terms of the final predictive accuracy of a learning algorithm) than filters because feature selection is optimized for the particular learning algorithm used. However, since a learning algorithm is employed to evaluate each and every set of features considered, wrappers are prohibitively expensive to run, and can be intractable for large databases containing many features. Furthermore, since the feature selection process is tightly coupled with a learning algorithm, wrappers are less general than filters and must be re-run when switching from one learning algorithm to another.

In the author's opinion, the advantages of filter approaches to feature selection outweigh their disadvantages. In general, filters execute many times faster than wrappers, and therefore stand a much better chance of scaling to databases with a large number of features than wrappers do. Filters do not require re-execution for different learning algorithms. Filters can provide the same benefits for learning as wrappers do. If improved accuracy for a particular learning algorithm is required, a filter can provide an intelligent starting feature subset for a wrapper—a process that is likely to result in a shorter, and hence faster, search for the wrapper. In a related scenario, a wrapper might be applied to search the *filtered* feature space—that is, the reduced feature space provided by a filter. Both methods help scale the wrapper to larger datasets. For these reasons, a filter approach to feature selection for machine learning is explored in this thesis.

Filter algorithms previously described in the machine learning literature have exhibited a number of drawbacks. Some algorithms do not handle noise in data, and others require that the level of noise be roughly specified by the user a-priori. In some cases, a subset of features is not selected explicitly; instead, features are ranked with the final choice left to the user. In other cases, the user must specify how many features are required, or must manually set a threshold by which feature selection terminates. Some algorithms require data to be transformed in a way that actually increases the initial number of features. This last case can result in a dramatic increase in the size of the search space.

1.2 Thesis statement

This thesis claims that feature selection for supervised machine learning tasks can be accomplished on the basis of correlation between features. In particular, this thesis investigates the following hypothesis:

> *A good feature subset is one that contains features highly correlated with (predictive of) the class, yet uncorrelated with (not predictive of) each other.*

Evaluation of the above hypothesis is accomplished by creating a feature selection algorithm that evaluates the worth of feature sets. An implementation (Correlation based Feature Selection, or CFS) is described in Chapter 4. CFS measures correlation between nominal features, so numeric features are first discretized. However, the general concept of correlation-based feature selection does not depend on any particular data transformation—all that must be supplied is a means of measuring the correlation between any two variables. So, in principle, the technique may be applied to a variety of supervised classification problems, including those in which the class (the variable to be predicted) is numeric.

CFS is a fully automatic algorithm—it does not require the user to specify any thresholds or the number of features to be selected, although both are simple to incorporate if desired. CFS operates on the original (albeit discretized) feature space, meaning that any knowledge induced by a learning algorithm, using features selected by CFS, can be interpreted in terms of the original features, not in terms of a transformed space. Most importantly, CFS is a filter, and, as such, does not incur the high computational cost associated with repeatedly invoking a learning algorithm.

CFS assumes that features are conditionally independent given the class. Experiments in

Chapter 6 show that CFS can identify relevant features when moderate feature dependencies exist. However, when features depend strongly on others given the class, CFS can fail to select all the relevant features. Chapter 8 explores methods for detecting feature dependencies given the class.

1.3 Thesis Overview

Chapter 2 defines terms and provides an overview of concepts from supervised machine learning. It also reviews some common machine learning algorithms and techniques for discretization—the process of converting continuous attributes to nominal attributes. Many feature selectors (including the implementation of CFS presented here) and machine learning algorithms are best suited to, or cannot handle problems in which attributes are nominal.

Chapter 3 surveys feature selection techniques for machine learning. Two broad categories of algorithms are discussed—those that involve a machine learning scheme to estimate the worth of features, and those that do not. Advantages and disadvantages of both approaches are discussed.

Chapter 4 begins by presenting the rationale for correlation based feature selection, with ideas borrowed from test theory. Three methods of measuring association between nominal variables are reviewed and empirically examined in Section 4.3. The behaviour of these measures with respect to attributes with more values and the number of available training examples is discussed; emphasis is given to their suitability for use in a correlation based feature selector. Section 4.4 describes CFS, an implementation of a correlation based feature selector based on the rationale of Section 4.1 and incorporating the correlation measures discussed in Section 4.2. Operational requirements and assumptions of the algorithm are discussed, along with its computational expense and some simple optimizations that can be employed to decrease execution time.

Chapter 5 describes the datasets used in the experiments discussed in Chapters 6, 7, and 8. It also outlines the experimental method.

The first half of Chapter 6 empirically tests three variations of CFS (each employing one of the correlation measures examined in Chapter 4) on artificial problems. It is shown that CFS is effective at eliminating irrelevant and redundant features, and can identify relevant features as long as they do not strongly depend on other features. One of the three correlation measures is shown to be inferior to the other two when used with CFS. The second half of Chapter 6 evaluates CFS with machine learning algorithms applied to natural learning domains. Results are presented and analyzed in detail for one of the three variations of CFS. It is shown that, in many cases, CFS improves the performance and reduces the size of induced knowledge structures for machine learning algorithms. A shortcoming in CFS is revealed by results on several datasets. In some cases CFS will fail to select locally predictive features, especially if they are overshadowed by strong, globally predictive ones. A method of merging feature subsets is shown to partially mitigate the problem.

Chapter 7 compares CFS with a well known implementation of the wrapper approach to feature selection. Results show that, in many cases, CFS gives results comparable to the wrapper, and, in general, outperforms the wrapper on small datasets. Cases where CFS is inferior to the wrapper can be attributed to the shortcoming of the algorithm revealed in Chapter 6, and to the presence of strong class-conditional feature dependency. CFS is shown to execute significantly faster than the wrapper.

Chapter 8 presents two methods of extending CFS to detect class-conditional feature dependency. The first considers pairwise combinations of features; the second incorporates a feature weighting algorithm that is sensitive to higher order (including higher than pairwise) feature dependency. The two methods are compared and results show that both improve results on some datasets. The second method is shown to be less reliable than the first.

Chapter 9 presents conclusions and suggests future work.

Chapter 2

Supervised Machine Learning: Concepts and Definitions

The field of artificial intelligence embraces two approaches to artificial learning [Hut93]. The first is motivated by the study of mental processes and says that artificial learning is the study of algorithms embodied in the human mind. The aim is to discover how these algorithms can be translated into formal languages and computer programs. The second approach is motivated from a practical computing standpoint and has less grandiose aims. It involves developing programs that learn from past data, and, as such, is a branch of data processing. The sub-field of machine learning has come to epitomize the second approach to artificial learning and has grown rapidly since its birth in the mid-seventies. Machine learning is concerned (on the whole) with *concept learning* and *classification learning*. The latter is simply a generalization of the former [Tho92].

2.1 The Classification Task

Learning how to classify objects to one of a pre-specified set of categories or classes is a characteristic of intelligence that has been of keen interest to researchers in psychology and computer science. Identifying the common "core" characteristics of a set of objects that are representative of their class is of enormous use in focusing the attention of a person or computer program. For example, to determine whether an animal is a zebra, people know to look for stripes rather than examine its tail or ears. Thus, stripes figure strongly in our *concept* (generalization) of zebras. Of course stripes alone are not sufficient to form a class description for zebras as tigers have them also, but they are certainly one of the important characteristics. The ability to perform classification and to be able to *learn* to classify gives people and computer programs the power to make decisions. The efficacy of these decisions is affected by performance on the classification task.

In machine learning, the classification task described above is commonly referred to as *supervised learning*. In supervised learning there is a specified set of classes, and example objects are labeled with the appropriate class (using the example above, the program is told what is a zebra and what is not). The goal is to generalize (form class descriptions) from the training objects that will enable novel objects to be identified as belonging to one of the classes. In contrast to supervised learning is *unsupervised learning*. In this case the program is not told which objects are zebras. Often the goal in unsupervised learning is to decide which objects should be grouped together—in other words, the learner forms the classes itself. Of course, the success of classification learning is heavily dependent on the quality of the data provided for training—a learner has only the input to learn from. If the data is inadequate or irrelevant then the concept descriptions will reflect this and misclassification will result when they are applied to new data.

2.2 Data Representation

In a typical supervised machine learning task, data is represented as a table of *examples* or *instances*. Each instance is described by a fixed number of measurements, or *features*, along with a label that denotes its class. Features (sometimes called *attributes*) are typically one of two types: nominal (values are members of an unordered set), or numeric (values are real numbers). Table 2.1 [Qui86] shows fourteen instances of suitable and unsuitable days for which to play a game of golf. Each instance is a day described in terms of the (nominal) attributes Outlook, Humidity, Temperature and Wind, along with the class label which indicates whether the day is suitable for playing golf or not.

A typical application of a machine learning algorithms requires two sets of examples: *training examples* and *test examples*. The set of training examples are used to produce the

Instance #	Features				Class
	Outlook	Temperature	Humidity	Wind	
1	sunny	hot	high	false	Don't play
2	sunny	hot	high	true	Don't Play
3	overcast	hot	high	false	Play
4	rain	mild	high	false	Play
5	rain	cool	normal	false	Play
6	rain	cool	normal	true	Don't Play
7	overcast	cool	normal	true	Play
8	sunny	mild	high	false	Don't Play
9	sunny	cool	normal	false	Play
10	rain	mild	normal	false	Play
11	sunny	mild	normal	true	Play
12	overcast	mild	high	true	Play
13	overcast	hot	normal	false	Play
14	rain	mild	high	true	Don't Play

Table 2.1: The "Golf" dataset.

learned concept descriptions and a separate set of test examples are needed to evaluate the accuracy. When testing, the class labels are not presented to the algorithm. The algorithm takes, as input, a test example and produces, as output, a class label (the predicted class for that example).

2.3 Learning Algorithms

A *learning algorithm*, or an *induction algorithm*, forms concept descriptions from example data. Concept descriptions are often referred to as the *knowledge* or *model* that the learning algorithm has induced from the data. Knowledge may be represented differently from one algorithm to another. For example, C4.5 [Qui93] represents knowledge as a decision tree; naive Bayes [Mit97] represents knowledge in the form of probabilistic summaries.

Throughout this thesis, three machine learning algorithms are used as a basis for comparing the effects of feature selection with no feature selection. These are naive Bayes, C4.5, and IB1—each represents a different approach to learning. These algorithms are well known in the machine learning community and have proved popular in practice. C4.5 is the most sophisticated algorithm of the three and induces knowledge that is (arguably) easier to interpret than the other two. IB1 and naive Bayes have proved popular because they are simple to implement and have been shown to perform competitively with more complex algorithms such as C4.5 [CN89, CS93, LS94a] . The following three sections briefly review these algorithms and indicate under what conditions feature selection can be useful.

2.3.1 Naive Bayes

The naive Bayes algorithm employs a simplified version of Bayes formula to decide which class a novel instance belongs to. The posterior probability of each class is calculated, given the feature values present in the instance; the instance is assigned the class with the highest probability. Equation 2.1 shows the naive Bayes formula, which makes the assumption that feature values are statistically independent within each class.

$$p(C_i|v_1, v_2, \ldots, v_n) = \frac{p(C_i) \prod_{j=1}^{n} p(v_j|C_i)}{p(v_1, v_2, \ldots, v_n)} \tag{2.1}$$

The left side of Equation 2.1 is the posterior probability of class C_i given the feature values, $< v_1, v_2, \ldots, v_n >$, observed in the instance to be classified. The denominator of the right side of the equation is often omitted because it is a constant which is easily computed if one requires that the posterior probabilities of the classes sum to one. Learning with the naive Bayes classifier is straightforward and involves simply estimating the probabilities in the right side of Equation 2.1 from the training instances. The result is a probabilistic summary for each of the possible classes. If there are numeric features it is common practice to assume a normal distribution—again the necessary parameters are estimated from the training data.

Tables 2.2(a) through 2.2(d) are contingency tables showing frequency distributions for the relationships between the features and the class in the golf dataset (Table 2.1). From these tables is easy to calculate the probabilities necessary to apply Equation 2.1.

Imagine we woke one morning and wished to determine whether the day is suitable for a game of golf. Noting that the outlook is sunny, the temperature is hot, the humidity is nor-

	Play	Don't Play	
Sunny	2	3	5
Overcast	4	0	4
Rain	3	2	5
	9	5	14

(a) Outlook

	Play	Don't Play	
Hot	2	2	4
Mild	4	2	6
Cool	3	1	4
	9	5	14

(b) Temperature

	Play	Don't Play	
High	3	4	7
Norm	6	1	7
	9	5	14

(c) Humidity

	Play	Don't Play	
True	3	5	6
False	6	2	8
	9	5	14

(d) Wind

Table 2.2: Contingency tables compiled from the "Golf" data.

mal and there is no wind (wind=false), we apply Equation 2.1 and calculate the posterior probability for each class, using probabilities derived from Tables 2.2(a) through 2.2(d):

$$\begin{aligned} p(\text{Don't Play} \mid \text{sunny, hot, normal, false}) &= p(\text{Don't Play}) \times p(\text{sunny} \mid \text{Don't Play}) \times \\ &\quad p(\text{hot} \mid \text{Don't Play}) \times p(\text{normal} \mid \text{Don't Play}) \times \\ &\quad p(\text{false} \mid \text{Don't Play}) \\ &= 5/14 \times 3/5 \times 2/5 \times 1/5 \times 2/5 \\ &= 0.0069. \end{aligned}$$

$$\begin{aligned} p(\text{Play} \mid \text{sunny, hot, normal, false}) &= p(\text{Play}) \times p(\text{sunny} \mid \text{Play}) \times \\ &\quad p(\text{hot} \mid \text{Play}) \times p(\text{normal} \mid \text{Play}) \times \\ &\quad p(\text{false} \mid \text{Play}) \\ &= 9/14 \times 2/9 \times 2/9 \times 6/9 \times 6/9 \\ &= 0.0141. \end{aligned}$$

On this day we would play golf.

Due to the assumption that feature values are independent within the class, the naive Bayesian classifier's predictive performance can be adversely affected by the presence of redundant attributes in the training data. For example, if there is a feature X that is perfectly correlated with a second feature Y, then treating them as independent means that X (or Y) has twice as much affect on Equation 2.1 as it should have. Langley and Sage [LS94a] have found that naive Bayes performance improves when redundant features are removed. However, Domingos and Pazzani [DP96] have found that, while strong correlations between features will degrade performance, naive Bayes can still perform very well when moderate dependencies exist in the data. The explanation for this is that moderate dependencies will result in inaccurate probability estimation, but the probabilities are not so far "wrong" as to result in increased mis-classification.

The version of naive Bayes used for the experiments described in this thesis is that provided in the $\mathcal{MLC}$++ utilities [KJL$^+$94]. In this version, the probabilities for nominal features are estimated using frequency counts calculated from the training data. The probabilities for numeric features are assumed to come from a normal distribution; again, the necessary parameters are estimated from training data. Any zero frequencies are replaced by $0.5/m$ as the probability, where m is the number of training examples.

2.3.2 C4.5 Decision Tree Generator

C4.5 [Qui93], and its predecessor, ID3 [Qui86], are algorithms that summarise training data in the form of a decision tree. Along with systems that induce logical rules, decision tree algorithms have proved popular in practice. This is due in part to their robustness and execution speed, and to the fact that explicit concept descriptions are produced, which users can interpret. Figure 2.1 shows a decision tree that summarises the golf data. Nodes in the tree correspond to features, and branches to their associated values. The leaves of the tree correspond to classes. To classify a new instance, one simply examines the features tested at the nodes of the tree and follows the branches corresponding to their observed values in the instance. Upon reaching a leaf, the process terminates, and the class at the leaf is assigned to the instance.

Using the decision tree (Figure 2.1) to classify the example day (sunny, hot, normal, false) initially involves examining the feature at the root of the tree (Outlook). The value

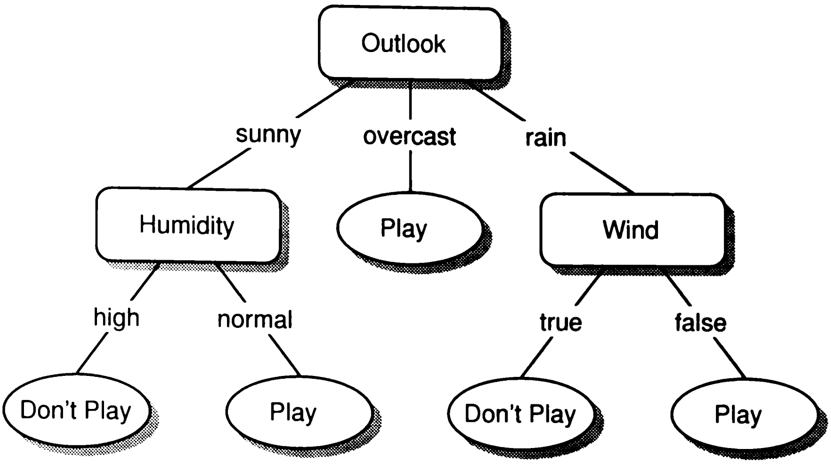


Figure 2.1: A decision tree for the "Golf" dataset. Branches correspond to the values of attributes; leaves indicate classifications.

for Outlook in the new instance is "sunny", so the left branch is followed. Next the value for Humidity is evaluated—in this case the new instance has the value "normal", so the right branch is followed. This brings us to a leaf node and the instance is assigned the class "Play".

To build a decision tree from training data, C4.5 and ID3 employ a greedy approach that uses an information theoretic measure as its guide. Choosing an attribute for the root of the tree divides the training instances into subsets corresponding to the values of the attribute. If the entropy of the class labels in these subsets is less than the entropy of the class labels in the full training set, then information has been gained (see Section 4.2.1 in Chapter 4) through splitting on the attribute. C4.5 uses the *gain ratio* criterion [Qui86] to select the attribute attribute to be at the root of the tree. The gain ratio criterion selects, from among those attributes with an average-or-better gain, the attribute that maximsises the ratio of its gain divided by its entropy. The algorithm is applied recursively to form sub-trees, terminating when a given subset contains instances of only one class.

The main difference between C4.5 and ID3 is that C4.5 prunes its decision trees. Pruning simplifies decision trees and reduces the probability of overfitting the training data [Qui87]. C4.5 prunes by using the upper bound of a confidence interval on the resubstitution error. A node is replaced by its best leaf when the estimated error of the leaf is within one standard deviation of the estimated error of the node.

C4.5 has proven to be a benchmark against which the performance of machine learning algorithms are measured. As an algorithm it is robust, accurate, fast, and, as an added bonus, it produces a comprehensible structure summarising the knowledge it induces. C4.5 deals remarkably well with irrelevant and redundant information, which is why feature selection has generally resulted in little if any improvement in its accuracy [JKP94]. However, removing irrelevant and redundant information can reduce the size of the trees induced by C4.5 [JKP94, KJ96]. Smaller trees are preferred because they are easier to understand.

The version of C4.5 used in experiments throughout this thesis is the original algorithm implemented by Quinlan [Qui93].

2.3.3 IB1-Instance Based Learner

Instance based learners represent knowledge in the form of specific cases or experiences. They rely on efficient matching methods to retrieve stored cases so they can be applied in novel situations. Like the Naive Bayes algorithm, instance based learners are usually computationally simple, and variations are often considered as models of human learning [CLW97]. Instance based learners are sometimes called *lazy* learners because learning is delayed until classification time, with most of the power residing in the matching scheme.

IB1 [AKA91] is an implementation of the simplest similarity based learner, known as nearest neighbour. IB1 simply finds the stored instance closest (according to a Euclidean distance metric) to the instance to be classified. The new instance is assigned to the retrieved instance's class. Equation 2.2 shows the distance metric employed by IB1.

$$D(x, y) = \sqrt{\sum_{j=1}^{n} f(x_j, y_j)} \tag{2.2}$$

Equation 2.2 gives the distance between two instances x and y; x_j and y_j refer to the jth feature value of instance x and y, respectively. For numeric valued attributes $f(x_j, y_j) =$

$(x_j - y_j)^2$; for symbolic valued attributes $f(x, y) = 0$, if the feature values x_j and y_j are the same, and 1 if they differ.

Table 2.3 shows the distance from the example day (sunny, hot, normal, false) to each of the instances in the golf data set by Equation 2.2. In this case there are three instances that are equally close to the example day, so an arbitrary choice would be made between them. An extension to the nearest neighbour algorithm, called k nearest neighbours, uses the most prevalent class from the k closest cases to the novel instance—where k is a parameter set by the user.

Instance #	Distance	Instance #	Distance
1	1	8	2
2	2	9	1
3	2	10	2
4	3	11	2
5	2	12	4
6	3	13	1
7	2	14	4

Table 2.3: Computed distance values for the "Golf" data.

The simple nearest neighbour algorithm is known to be adversely affected by the presence of irrelevant features in its training data. While nearest neighbour can learn in the presence of irrelevant information, it requires more training data to do so and, in fact, the amount of training data needed (sample complexity) to reach or maintain a given accuracy level has been shown to grow exponentially with the number of irrelevant attributes [AKA91, LS94c, LS94b]. Therefore, it is possible to improve the predictive performance of nearest neighbour, when training data is limited, by removing irrelevant attributes.

Furthermore, nearest neighbour is slow to execute due to the fact that each example to be classified must be compared to each of the stored training cases in turn. Feature selection can reduce the number of training cases because fewer features equates with fewer distinct instances (especially when features are nominal). Reducing the number of training cases needed (while maintaining an acceptable error rate) can dramatically increase the speed of the algorithm.

The version of IB1 used in experiments throughout this thesis is the version implemented by David Aha [AKA91]. Equation 2.2 is used to compute similarity between instances.

Attribute values are linearly normalized to ensure each attribute has the same affect on the similarity function.

2.4 Performance Evaluation

Evaluating the performance of learning algorithms is a fundamental aspect of machine learning. Not only is it important in order to compare competing algorithms, but in many cases is an integral part of the learning algorithm itself. An estimate of classification accuracy on new instances is the most common performance evaluation criterion, although others based on information theory have been suggested [KB91, CLW96].

In this thesis, classification accuracy is the primary evaluation criterion for experiments using feature selection with the machine learning algorithms. Feature selection is considered successful if the dimensionality of the data is reduced and the accuracy of a learning algorithm improves or remains the same. In the case of C4.5, the size (number of nodes) of the induced trees is also important—smaller trees are preferred because they are easier to interpret. Classification accuracy is defined as the percentage of test examples correctly classified by the algorithm. The error rate (a measure more commonly used in statistics) of an algorithm is one minus the accuracy. Measuring accuracy on a test set of examples is better than using the training set because examples in the test set have not been used to induce concept descriptions. Using the training set to measure accuracy will typically provide an optimistically biased estimate, especially if the learning algorithm *overfits* the training data.

Strictly speaking, the definition of accuracy given above is the *sample accuracy* of an algorithm. Sample accuracy is an estimate of the (unmeasurable) *true accuracy* of the algorithm, that is, the probability that the algorithm will correctly classify an instance drawn from the unknown distribution D of examples. When data is limited, it is common practice to *resample* the data, that is, partition the data into training and test sets in different ways. A learning algorithm is trained and tested for each partition and the accuracies averaged. Doing this provides a more reliable estimate of the true accuracy of an algorithm.

Random subsampling and k-fold *cross-validation* are two common methods of resampling [Gei75, Sch93]. In random subsampling, the data is randomly partitioned into disjoint training and test sets multiple times. Accuracies obtained from each partition are averaged. In k-fold cross-validation, the data is randomly split into k mutually exclusive subsets of approximately equal size. A learning algorithm is trained and tested k times; each time it is tested on one of the k folds and trained using the remaining $k - 1$ folds. The cross-validation estimate of accuracy is the overall number of correct classifications, divided by the number of examples in the data. The random subsampling method has the advantage that it can be repeated an indefinite number of times. However, it has the disadvantage that the test sets are not independently drawn with respect to the underlying distribution of examples D. Because of this, using a t-test for paired differences with random subsampling can lead to increased chance of Type I error—that is, identifying a significant difference when one does not actually exist [Die88]. Using a t-test on the accuracies produced on each fold of k fold cross-validation has lower chance of Type I error but may not give a stable estimate of accuracy. It is common practice to repeat k fold cross-validation n times in order to provide a stable estimate. However, this of course renders the test sets non-independent and increases the chance of Type I error. Unfortunately, there is no satisfactory solution to this problem. Alternative tests suggested by Dietterich [Die88] have low chance of Type I error but *high* chance of Type II error—that is, failing to identify a significant difference when one does actually exist.

Stratification is a process often applied during random subsampling and k-fold cross-validation. Stratification ensures that the class distribution from the whole dataset is preserved in the training and test sets. Stratification has been shown to help reduce the variance of the estimated accuracy—especially for datasets with many classes [Koh95b].

Stratified random subsampling with a paired t-test is used herein to evaluate accuracy. Appendix D reports results for the major experiments using the 5×2cv paired t test recommended by Dietterich [Die88]. As stated above, this test has decreased chance of type I error, but increased chance of type II error (see the appendix for details).

Plotting *learning curves* are another way that machine learning algorithms can be compared. A learning curve plots the classification accuracy of a learning algorithm as a function of the size of the training set—it shows how quickly an algorithm's accuracy improves as it is given access to more training examples. In situations where training data is limited, it is preferable to use a learning algorithm that achieves high accuracy with small training sets.

2.5 Attribute Discretization

Most classification tasks in machine learning involve learning to distinguish among nominal class values[1], but may involve features that are ordinal or continuous as well as nominal. While many machine learning algorithms have been developed to deal with mixed data of this sort, recent research [Tin95, DKS95] shows that common machine learning algorithms such as instance based learners and naive Bayes benefit from treating all features in a uniform fashion. One of the most common methods of accomplishing this is called *discretization*. Discretization is the process of transforming continuous valued attributes to nominal. In fact, the decision tree algorithm C4.5 [Qui93] accomplishes this internally by dividing continuous features into discrete ranges during the construction of a decision tree. Many of the feature selection algorithms described in the next chapter require continuous features to be discretized, or give superior results if discretization is performed at the outset [AD91, HNM95, KS96b, LS96]. Discretization is used as a preprocessing step for the correlation-based approach to feature selection presented in this thesis, which requires all features to be of the same type.

This section describes some discretization approaches from the machine learning literature.

[1] CART [BFOS84], M5'[WW97], and K* [CT95] are some machine learning algorithms capable of dealing with continuous class data.

2.5.1 Methods of Discretization

Dougherty, Kohavi, and Sahami [DKS95] define 3 axes along which discretization methods can be categorised:

1. Supervised versus. unsupervised;

2. Global versus. local;

3. Static versus. dynamic.

Supervised methods make use of the class label when discretizing features. The distinction between global and local methods is based on when discretization is performed. Global methods discretize features prior to induction, whereas local methods carry out discretization during the induction process. Local methods may produce different discretizations[2] for particular local regions of the instance space. Some discretization methods require a parameter, k, indicating the maximum number of intervals by which to divide a feature. Static methods perform one discretization pass on the data for each feature and determine the value of k for each feature independently of the others. On the other hand, dynamic methods search the space of possible k values for all features simultaneously. This allows inter-dependencies in feature discretization to be captured.

Global methods of discretization are most relevant to the feature selection algorithm presented in this thesis because feature selection is generally a global process (that is, a single feature subset is chosen for the entire instance space). Kohavi and Sahami [KS96a] have compared static discretization with dynamic methods using cross-validation to estimate the accuracy of different values of k. They report no significant improvement in employing dynamic discretization over static methods.

The next two sections discuss several methods for unsupervised and supervised global discretization of numeric features in common usage.

Unsupervised Methods The simplest discretization method is called *equal interval*

[2] For example, C4.5 may split the same continuous feature differently down different branches of a decision tree

width. This approach divides the range of observed values for a feature into k equal sized bins, where k is a parameter provided by the user. Dougherty *et al.* [DKS95] point out that this method of discretization is sensitive to outliers that may drastically skew the range. For example, given the observed feature values

$$0, 0, 0.5, 1, 1, 1.2, 2, 2, 3, 3, 3, 4, 4$$

and setting $k = 4$ gives a bin width of $(4 - 0) \div 4 = 1$, resulting in discrete ranges

$$[0 - 1], (1 - 2], (2 - 3], (3 - 4]$$

with a reasonably even distribution of examples across the bins. However, suppose there was an outlying value of 100. This would cause the ranges

$$[0 - 25], (25 - 50], (50 - 75], (75 - 100]$$

to be formed. In this case, all the examples except the example with the value 100 would fall into the first bin.

Another simple discretization method, *equal frequency intervals*, requires a feature's values to be sorted, and assigns $1/k$ of the values to each bin. Wong and Chiu [WC87] describe a variation on equal frequency intervals called *maximal marginal entropy* that iteratively adjusts the boundaries to minimise the entropy at each interval.

Because unsupervised methods do not make use of the class in setting interval boundaries, Dougherty *et al.* [DKS95] note that classification information can be lost as a result of placing values that are strongly associated with different classes in the same interval. The next section discusses methods for supervised discretization which overcome this problem.

Supervised Methods Holte [Hol93] presents a simple supervised discretization method that is incorporated in his one-level decision tree algorithm (1R). The method first sorts the values of a feature, and then attempts to find interval boundaries such that each interval has a strong majority of one particular class. The method is constrained to form intervals of some minimal size in order to avoid having intervals with very few instances.

Setiono and Liu [SL95] present a statistically justified heuristic method for supervised discretization called Chi2. A numeric feature is initially sorted by placing each observed value into its own interval. The next step uses a chi-square statistic χ^2 to determine whether the relative frequencies of the classes in adjacent intervals are similar enough to justify merging. The formula for computing the χ^2 value for two adjacent intervals is

$$\chi^2 = \sum_{i=1}^{2} \sum_{j=1}^{C} \frac{(A_{ij} - E_{ij})^2}{E_{ij}}, \tag{2.3}$$

where C is the number of classes, A_{ij} is the number of instances in the i-th interval with class j, R_i is the number of instances in the i-th interval, C_j is the number of instances of class j in the two intervals, N is the total number of instances in the two intervals, and E_{ij} is the expected frequency of $A_{ij} = R_i \times C_j/N$.

The extent of the merging process is controlled by an automatically set χ^2 threshold. The threshold is determined through attempting to maintain the fidelity of the original data.

Catlett [Cat91] and Fayyad and Irani [FI93] use a minimum entropy heuristic to discretize numeric features. The algorithm uses the class entropy of candidate partitions to select a cut point for discretization. The method can then be applied recursively to the two intervals of the previous split until some stopping conditions are satisfied, thus creating multiple intervals for the feature. For a set of instances S, a feature A, and a cut point T, the class information entropy of the partition induced by T is given by

$$E(A,T;S) = \frac{|S_1|}{S}\text{Ent}(S_1) + \frac{|S_2|}{S}\text{Ent}(S_2), \tag{2.4}$$

where S_1 and S_2 are two intervals of S bounded by cut point T, and $\text{Ent}(S)$ is the class entropy of a subset S given by

$$\mathrm{Ent}(S) = -\sum_{i=1}^{C} p(C_i, S)\log_2(p(C_i, S)). \tag{2.5}$$

For feature A, the cut point T which minimises Equation 2.5 is selected (conditionally) as a binary discretization boundary. Catlett [Cat91] employs ad hoc criteria for terminating the splitting procedure. These include: stopping if the number of instances in a partition is sufficiently small, stopping if some maximum number of partitions have been created, and stopping if the entropy induced by all possible cut points for a set is equal. Fayyad and Irani [FI93] employ a stopping criterion based on the minimum description length principle [Ris78]. The stopping criterion prescribes accepting a partition induced by T if and only if the cost of encoding the partition and the classes of the instances in the intervals induced by T is less than the cost of encoding the classes of the instances before splitting. The partition induced by cut point T is accepted *iff*

$$\mathrm{Gain}(A, T; S) > \frac{\log_2(N-1)}{N} + \frac{\Delta(A, T; S)}{N}, \tag{2.6}$$

where N is the number of instances in the set S,

$$\mathrm{Gain}(A, T; S) = \mathrm{Ent}(S) - E(A, T; S), \tag{2.7}$$

and

$$\Delta(A, T; S) = \log_2(3^c - 2) - [c\mathrm{Ent}(S) - c_1\mathrm{Ent}(S_1) - c_2\mathrm{Ent}(S_2)]. \tag{2.8}$$

In Equation 2.8, c, c_1, and c_2 are the number of distinct classes present in S, S_1, and S_2 respectively.

C4.5 [Qui86, Qui93] uses Equation 2.7 locally at the nodes of a decision tree to determine a binary split for a numeric feature. Kohavi and Sahami [KS96a] use C4.5 to perform global discretization on numeric features. C4.5 is applied to each numeric feature separately to build a tree which contains binary splits that only test a single feature. C4.5's internal pruning mechanism is applied to determine an appropriate number of nodes in the tree and hence the number of discretization intervals.

A number of studies [DKS95, KS96a] comparing the effects of using various discretization techniques (on common machine learning domains and algorithms) have found the method of Fayyad and Irani to be superior overall. For that reason, this method of discretization is used in the experiments described in chapters 6, 7 and 8.

Chapter 3

Feature Selection for Machine Learning

Many factors affect the success of machine learning on a given task. The representation and quality of the example data is first and foremost. Theoretically, having more features should result in more discriminating power. However, practical experience with machine learning algorithms has shown that this is not always the case. Many learning algorithms can be viewed as making a (biased) estimate of the probability of the class label given a set of features. This is a complex, high dimensional distribution. Unfortunately, induction is often performed on limited data. This makes estimating the many probabilistic parameters difficult. In order to avoid overfitting the training data, many algorithms employ the Occam's Razor [GL97] bias to build a simple model that still achieves some acceptable level of performance on the training data. This bias often leads an algorithm to prefer a small number of predictive attributes over a large number of features that, if used in the proper combination, are fully predictive of the class label. If there is too much irrelevant and redundant information present or the data is noisy and unreliable, then learning during the training phase is more difficult.

Feature subset selection is the process of identifying and removing as much irrelevant and redundant information as possible. This reduces the dimensionality of the data and may allow learning algorithms to operate faster and more effectively. In some cases, accuracy on future classification can be improved; in others, the result is a more compact, easily interpreted representation of the target concept.

Recent research has shown common machine learning algorithms to be adversely affected by irrelevant and redundant training information. The simple nearest neighbour algorithm is sensitive to irrelevant attributes—its sample complexity (number of training examples needed to reach a given accuracy level) grows exponentially with the number of irrelevant attributes [LS94b, LS94c, AKA91]. Sample complexity for decision tree algorithms can grow exponentially on some concepts (such as parity) as well. The naive Bayes classifier can be adversely affected by redundant attributes due to its assumption that attributes are independent given the class [LS94a]. Decision tree algorithms such as C4.5 [Qui86, Qui93] can sometimes overfit training data, resulting in large trees. In many cases, removing irrelevant and redundant information can result in C4.5 producing smaller trees [KJ96].

This chapter begins by highlighting some common links between feature selection in pattern recognition and statistics and feature selection in machine learning. Important aspects of feature selection algorithms are described in section 3.2. Section 3.3 outlines some common heuristic search techniques. Sections 3.4 through 3.6 review current approaches to feature selection from the machine learning literature.

3.1 Feature Selection in Statistics and Pattern Recognition

Feature subset selection has long been a research area within statistics and pattern recognition [DK82, Mil90]. It is not surprising that feature selection is as much of an issue for machine learning as it is for pattern recognition, as both fields share the common task of classification. In pattern recognition, feature selection can have an impact on the economics of data acquisition and on the accuracy and complexity of the classifier [DK82]. This is also true of machine learning, which has the added concern of distilling useful knowledge from data. Fortunately, feature selection has been shown to improve the comprehensibility of extracted knowledge [KJ96].

Machine learning has taken inspiration and borrowed from both pattern recognition and statistics. For example, the heuristic search technique sequential backward elimination (section 3.3) was first introduced by Marill and Green [MG63]; Kittler [Kit78] introduced different variants, including a forward method and a stepwise method. The use of cross-validation for estimating the accuracy of a feature subset—which has become the backbone of the wrapper method in machine learning—was suggested by Allen [All74] and applied to the problem of selecting predictors in linear regression.

Many statistical methods[1] for evaluating the worth of feature subsets based on characteristics of the training data are only applicable to numeric features. Furthermore, these measures are often monotonic (increasing the size of the feature subset can never decrease performance)—a condition that does not hold for practical machine learning algorithms[2]. Because of this, search algorithms such as dynamic programming and branch and bound [NF77], which rely on monotonicity in order to prune the search space, are not applicable to feature selection algorithms that use or attempt to match the general bias of machine learning algorithms.

3.2 Characteristics of Feature Selection Algorithms

Feature selection algorithms (with a few notable exceptions) perform a search through the space of feature subsets, and, as a consequence, must address four basic issues affecting the nature of the search [Lan94]:

1. Starting point. Selecting a point in the feature subset space from which to begin the search can affect the direction of the search. One option is to begin with no features and successively add attributes. In this case, the search is said to proceed forward through the search space. Conversely, the search can begin with all features and successively remove them. In this case, the search proceeds backward through the search space. Another alternative is to begin somewhere in the middle and move outwards from this point.

2. Search organisation. An exhaustive search of the feature subspace is prohibitive for all but a small initial number of features. With N initial features there exist 2^N possible subsets. Heuristic search strategies are more feasible than exhaustive

[1] Measures such as residual sum of squares (RSS), Mallows C_p, and separability measures such as F Ratio and its generalisations are described in Miller [Mil90] and Parsons [Par87] respectively.

[2] For example, decision tree algorithms (such as C4.5 [Qui93]) discover regularities in training data by partitioning the data on the basis of observed feature values. Maintaining statistical reliability and avoiding overfitting necessitates the use of a small number of strongly predictive attributes.

ones and can give good results, although they do not guarantee finding the optimal subset. Section 2.2.3 discusses some heuristic search strategies that have been used for feature selection.

3. Evaluation strategy. How feature subsets are evaluated is the single biggest differentiating factor among feature selection algorithms for machine learning. One paradigm, dubbed the *filter* [Koh95b, KJ96] operates independent of any learning algorithm—undesirable features are filtered out of the data before learning begins. These algorithms use heuristics based on general characteristics of the data to evaluate the merit of feature subsets. Another school of thought argues that the bias of a particular induction algorithm should be taken into account when selecting features. This method, called the *wrapper* [Koh95b, KJ96], uses an induction algorithm along with a statistical re-sampling technique such as cross-validation to estimate the final accuracy of feature subsets. Figure 3.1 illustrates the filter and wrapper approaches to feature selection.

4. Stopping criterion. A feature selector must decide when to stop searching through the space of feature subsets. Depending on the evaluation strategy, a feature selector might stop adding or removing features when none of the alternatives improves upon the merit of a current feature subset. Alternatively, the algorithm might continue to revise the feature subset as long as the merit does not degrade. A further option could be to continue generating feature subsets until reaching the opposite end of the search space and then select the best.

3.3 Heuristic Search

Searching the space of feature subsets within reasonable time constraints is necessary if a feature selection algorithm is to operate on data with a large number of features. One simple search strategy, called greedy hill climbing, considers local changes to the current feature subset. Often, a local change is simply the addition or deletion of a single feature from the subset. When the algorithm considers only additions to the feature subset it is

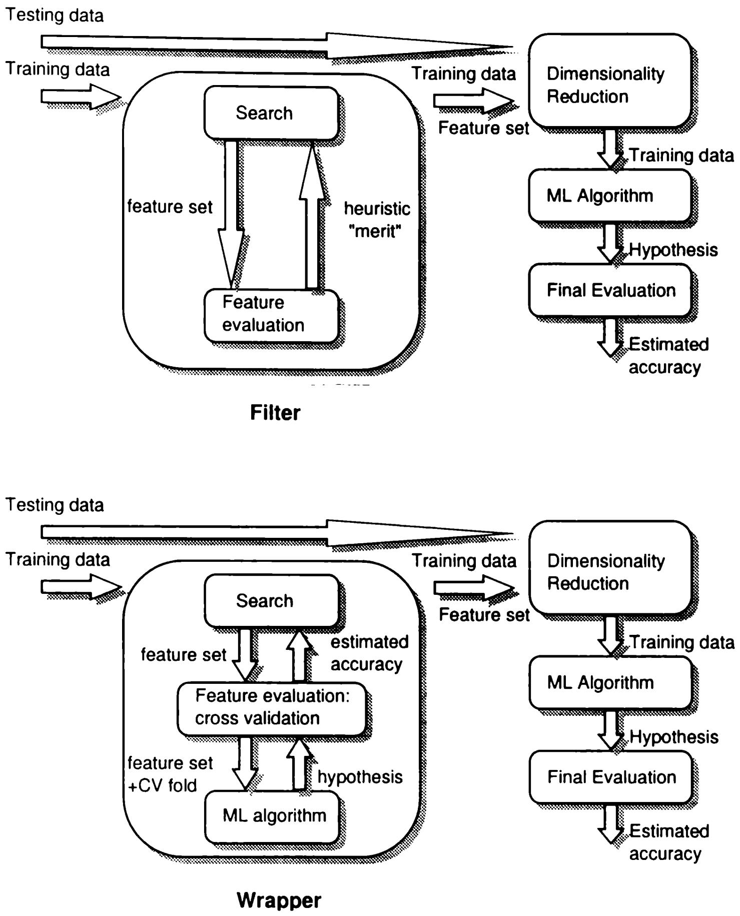


Figure 3.1: Filter and wrapper feature selectors.

known as *forward selection*; considering only deletions is known as *backward elimination* [Kit78, Mil90]. An alternative approach, called stepwise bi-directional search, uses both addition and deletion. Within each of these variations, the search algorithm may consider all possible local changes to the current subset and then select the best, or may simply choose the first change that improves the merit of the current feature subset. In either case, once a change is accepted, it is never reconsidered. Figure 3.2 shows the feature subset space for the golf data. If scanned from top to bottom, the diagram shows all local additions to each node; if scanned from bottom to top, the diagram shows all possible local deletions from each node. Table 3.1 shows the algorithm for greedy hill climbing search.

Best first search [RK91] is an AI search strategy that allows backtracking along the search path. Like greedy hill climbing, best first moves through the search space by making local

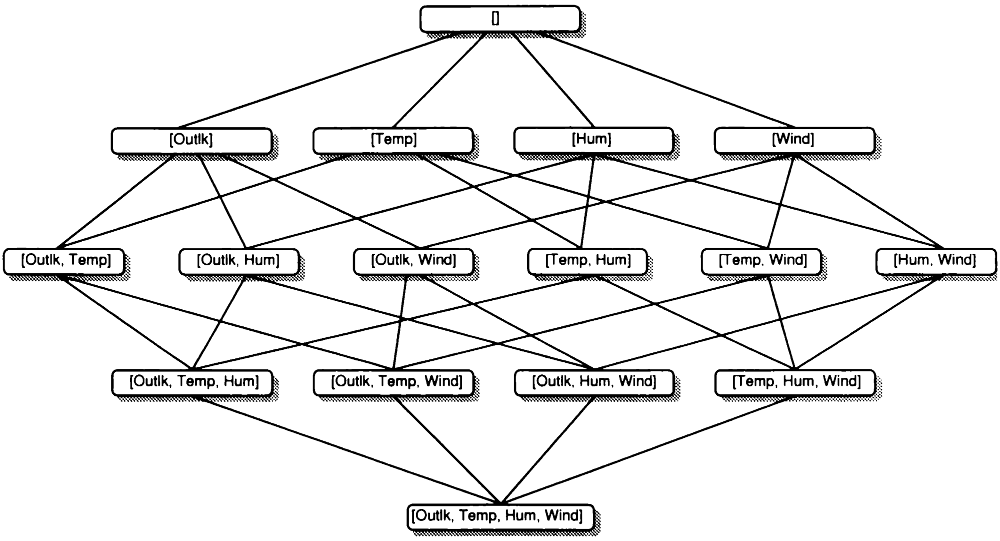


Figure 3.2: Feature subset space for the "golf" dataset.

```
1. Let $s \leftarrow$start state.
2. Expand $s$ by making each possible local change.
3. Evaluate each child $t$ of $s$.
4. Let $s' \leftarrow$child $t$ with highest evaluation $e(t)$.
5. If $e(s') \geq e(s)$ then $s \leftarrow s'$, goto 2.
6. Return $s$.
```

Table 3.1: Greedy hill climbing search algorithm

changes to the current feature subset. However, unlike hill climbing, if the path being explored begins to look less promising, the best first search can back-track to a more promising previous subset and continue the search from there. Given enough time, a best first search will explore the entire search space, so it is common to use a stopping criterion. Normally this involves limiting the number of fully expanded[3] subsets that result in no improvement. Table 3.2 shows the best first search algorithm.

```
1. Begin with the OPEN list containing the start state, the CLOSED list empty,
   and BEST$\leftarrow$start state.
2. Let $s = \arg\max e(x)$ (get the state from OPEN with the highest evaluation).
3. Remove $s$ from OPEN and add to CLOSED.
4. If $e(s) \geq e(\text{BEST})$, then BEST $\leftarrow s$.
5. For each child $t$ of $s$ that is not in the OPEN or CLOSED list, evaluate and add to OPEN.
6. If BEST changed in the last set of expansions, goto 2.
7. Return BEST.
```

Table 3.2: Best first search algorithm

Genetic algorithms are adaptive search techniques based on the principles of natural selection in biology [Hol75]. They employ a population of competing solutions—evolved over time—to converge to an optimal solution. Effectively, the solution space is searched in parallel, which helps in avoiding local optima. For feature selection, a solution is typically a fixed length binary string representing a feature subset—the value of each position in the string represents the presence or absence of a particular feature. The algorithm is an iterative process where each successive generation is produced by applying genetic operators such as *crossover* and *mutation* to the members of the current generation. Mutation changes some of the values (thus adding or deleting features) in a subset randomly. Crossover combines different features from a pair of subsets into a new subset. The application of genetic operators to population members is determined by their fitness (how good a feature subset is with respect to an evaluation strategy). Better feature subsets have a greater chance of being selected to form a new subset through crossover or mutation. In this manner, good subsets are "evolved" over time. Table 3.3 shows a simple genetic search strategy.

[3] A fully expanded subset is one in which all possible local changes have been considered.

```
1.  Begin by randomly generating an initial population $P$.
2.  Calculate $e(x)$ for each member $x \in P$.
3.  Define a probability distribution $p$ over the members of $P$ where $p(x) \propto e(x)$.
4.  Select two population members $x$ and $y$ with respect to $p$.
5.  Apply crossover to $x$ and $y$ to produce new population members $x'$ and $y'$.
6.  Apply mutation to $x'$ and $y'$.
7.  Insert $x'$ and $y'$ into $P'$ (the next generation).
8.  If $|P'| < |P|$, goto 4.
9.  Let $P \leftarrow P'$.
10. If there are more generations to process, goto 2.
11. Return $x \in P$ for which $e(x)$ is highest.
```

Table 3.3: Simple genetic search strategy.

3.4 Feature Filters

The earliest approaches to feature selection within machine learning were filter methods. All filter methods use heuristics based on general characteristics of the data rather than a learning algorithm to evaluate the merit of feature subsets. As a consequence, filter methods are generally much faster than wrapper methods, and, as such, are more practical for use on data of high dimensionality.

3.4.1 Consistency Driven Filters

Almuallim and Dieterich [AD91] describe an algorithm originally designed for boolean domains called FOCUS. FOCUS exhaustively searches the space of feature subsets until it finds the minimum combination of features that divides the training data into pure classes (that is, where every combination of feature values is associated with a single class). This is referred to as the "min-features bias". Following feature selection, the final feature subset is passed to ID3 [Qui86], which constructs a decision tree. There are two main difficulties with FOCUS, as pointed out by Caruanna and Freitag [CF94]. Firstly, since FOCUS is driven to attain consistency on the training data, an exhaustive search may be intractable if many features are needed to attain consistency. Secondly, a strong bias towards consistency can be statistically unwarranted and may lead to overfitting the training data—the algorithm will continue to add features to repair a single inconsistency.

The authors address the first of these problems in their 1992 paper [AD92]. Three algorithms—each consisting of forward selection search coupled with a heuristic to approximate the min-features bias—are presented as methods to make FOCUS computationally feasible on domains with many features.

The first algorithm evaluates features using the following information theoretic formula:

$$Entropy(Q) = -\sum_{i=0}^{2^{|Q|}-1} \frac{p_i + n_i}{|Sample|}\left[\frac{p_i}{p_i + N_i}\log_2\frac{p_i}{p_i + N_i} + \frac{n_i}{p_i + n_i}\log_2\frac{n_i}{p_i + n_i}\right]. \quad (3.1)$$

For a given feature subset Q, there are $2^{|Q|}$ possible truth value assignments to the features. A given feature set Q divides the training data into groups of instances with the same truth value assignments to the features in Q. Equation 3.1 measures the overall entropy of the class values in these groups—p_i and n_i denote the number of positive and negative examples in the i-th group respectively. At each stage, the feature which minimises Equation 3.1 is added to the current feature subset.

The second algorithm chooses the most discriminating feature to add to the current subset at each stage of the search. For a given pair of positive and negative examples, a feature is discriminating if its value differs between the two. At each stage, the feature is chosen which discriminates the greatest number of positive-negative pairs of examples—that have not yet been discriminated by any existing feature in the subset.

The third algorithm is like the second except that each positive-negative example pair contributes a weighted increment to the score of each feature that discriminates it. The increment depends on the total number of features that discriminate the pair.

Liu and Setiono [LS96] describe an algorithm similar to FOCUS called LVF. Like FOCUS, LVF is consistency driven and, unlike FOCUS, can handle noisy domains if the approximate noise level is known a-priori. LVF generates a random subset S from the feature subset space during each round of execution. If S contains fewer features than the current best subset, the *inconsistency* rate of the dimensionally reduced data described by S is compared with the *inconsistency* rate of the best subset. If S is at least as consistent as the best subset, S replaces the best subset. The inconsistency rate of the training data prescribed by a given feature subset is defined over all groups of matching instances. Within a group of matching instances the inconsistency count is the number of instances in the group minus the number of instances in the group with the most frequent class value. The overall inconsistency rate is the sum of the inconsistency counts of all groups of matching instances divided by the total number of instances.

Liu and Setiono report good results for LVF when applied to some artificial domains and mixed results when applied to commonly used natural domains. They also applied LVF to two "large" data sets—the first having $65,000$ instances described by 59 attributes; the second having $5,909$ instances described by 81 attributes. They report that LVF was able to reduce the number of attributes on both data sets by more than half. They also note that due to the random nature of LVF, the longer it is allowed to execute, the better the results (as measured by the inconsistency criterion).

Feature selection based on rough sets theory [Mod93, Paw91] uses notions of consistency similar to those described above. In rough sets theory an *information system* is a 4-tuple $S = (U, Q, V, f)$, where

U is the finite universe of instances.

Q is the finite set of features.

V is the set of possible feature values.

f is the information function. Given an instance and a feature, f maps it to a value $v \in V$.

For any subset of features $P \subseteq Q$, an *indiscernibility relation* IND(P) is defined as:

$$\begin{aligned} \text{IND}(P) &= (x, y) \in U \times U : f(x, a) = f(y, a), \\ & \text{for every feature } a. \in P \end{aligned} \tag{3.2}$$

The indiscernibility relation is an equivalence relation over U. Hence, it partitions the instances into equivalence classes—sets of instances indiscernible with respect to the features in P. Such a partition (classification) is denoted by $U/\text{IND}(P)$. In supervised machine learning, the sets of instances indiscernible with respect to the class attribute contain (obviously) the instances of each class.

For any subset of instances $X \subseteq U$ and subset of features $P \subseteq Q$, the lower $\underline{P}$, and the upper, $\overline{P}$ approximations of $X$are defined as follows:

$$\underline{P}(X) = \cup\{Y \in U/\text{IND}(P) : Y \subseteq X\} \tag{3.3}$$

$$\overline{P}(X) = \cup\{Y \in U/\text{IND}(P) : Y \cap X \neq \emptyset\} \tag{3.4}$$

If $\underline{P}(X) = \overline{P}(X)$ then X is an exact set (*definable* using feature subset P), otherwise X is a *rough* set with respect to P.

The instances in U that can be classified to the equivalence classes of $U/\text{IND}(P)$ by using feature set R is called the *positive region* of P with respect to R, and is defined as follows:

$$\text{POS}_R(P) = \bigcup_{X \in U/\text{IND}(P)} \underline{R}(X). \tag{3.5}$$

The degree of consistency afforded by feature subset R with respect to the equivalence classes of $U/\text{IND}(P)$ is given by:

$$\gamma_R(P) = \frac{|\text{POS}_R(P)|}{|U|}. \tag{3.6}$$

IF $\gamma_R(P) = 1$ then P is totally consistent with respect to R.

Feature selection in rough sets theory is achieved by identifying a *reduct* of a given set of features. A set $R \subseteq P$ is a reduct of P if it is *independent* and $\text{IND}(R) = \text{IND}(P)$. R is independent if there does not exist a strict subset R' of R such that $\text{IND}(R') = \text{IND}(R)$. Each reduct has the property that a feature cannot be removed from it without changing the indiscernibility relation.

Both rough sets and the LVF algorithm are likely to assign higher consistency to attributes that have many values. An extreme example is an attribute that has as many values as there are instances. An attribute such as this has little power to generalize beyond the training data. If R is such an attribute, and C is the class attribute, then it is easy to show that

$\text{POS}_R(C)$ contains all the instances[4] and $\gamma_R(P) = 1$. Similarly, for LVF, the feature R guarantees that there is no inconsistency in the data.

3.4.2 Feature Selection Through Discretization

Setiono and Liu [SL95] note that discretization has the potential to perform feature selection among numeric features. If a numeric feature can justifiably be discretized to a single value, then it can safely be removed from the data. The combined discretization and feature selection algorithm Chi2 (discussed in section 2.5.1), uses a chi-square statistic χ^2 to perform discretization. Numeric attributes are initially sorted by placing each observed value into its own interval. Each numeric attribute is then repeatedly discretized by using the χ^2 test to determine when adjacent intervals should be merged. The extent of the merging process is controlled by the use of an automatically set χ^2 threshold. The threshold is determined by attempting to maintain the original fidelity of the data—inconsistency (measured the same way as in the LVF algorithm described above) controls the process.

The authors report results on three natural domains containing a mixture of numeric and nominal features, using C4.5 [Qui86, Qui93] before and after discretization. They conclude that Chi2 is effective at improving C4.5's performance and eliminating some features. However, it is not clear whether C4.5's improvement is due entirely to some features having been removed or whether discretization plays a role as well.

3.4.3 Using One Learning Algorithm as a Filter for Another

Several researchers have explored the possibility of using a particular learning algorithm as a pre-processor to discover useful feature subsets for a primary learning algorithm.

Cardie [Car95] describes the application of decision tree algorithms to the task of selecting feature subsets for use by instance based learners. C4.5 was applied to three natural language data sets; only the features that appeared in the final decision trees were used

[4] Each element in $U/\text{IND}(R)$ is a set containing exactly one unique instance from U. Therefore, each element of $U/\text{IND}(R)$ is a subset of one of the equivalence classes in $U/\text{IND}(C)$.

with a k nearest neighbour classifier. The use of this hybrid system resulted in significantly better performance than either C4.5 or the k nearest neighbour algorithm when used alone.

In a similar approach, Singh and Provan [SP96] use a greedy *oblivious* decision tree algorithm to select features from which to construct a Bayesian network. Oblivious decision trees differ from those constructed by algorithms such as C4.5 in that all nodes at the same level of an oblivious decision tree test the same attribute. Feature subsets selected by three oblivious decision tree algorithms—each employing a different information theoretic splitting criterion—were evaluated with a Bayesian network classifier on several machine learning datasets. Results showed that Bayesian networks using features selected by the oblivious decision tree algorithms outperformed Bayesian networks without feature selection and Bayesian networks with features selected by a wrapper.

Holmes and Nevill-Manning [HNM95] use Holte's 1R system [Hol93] to estimate the predictive accuracy of individual features. 1R builds rules based on a single features (called 1-rules[5]). If the data is split into training and test sets, it is possible to calculate a classification accuracy for each rule and hence each feature. From classification scores, a ranked list of features is obtained. Experiments with choosing a select number of the highest ranked features and using them with common machine learning algorithms showed that, on average, the top three or more features are as accurate as using the original set. This approach is unusual due to the fact that no search is conducted. Instead, it relies on the user to decide how many features to include from the ranked list in the final subset.

Pfahringer [Pfa95] uses a program for inducing decision table majority classifiers to select features. DTM (Decision Table Majority) classifiers are a simple type of nearest neighbour classifier where the similarity function is restricted to returning stored instances that are exact matches with the instance to be classified. If no instances are returned, the most prevalent class in the training data is used as the predicted class; otherwise, the majority class of all matching instances is used. DTMs work best when all features are nominal. Induction of a DTM is achieved by greedily searching the space of possible decision tables. Since a decision table is defined by the features it includes, induction is simply

[5] 1-rules can be thought of as single level decision trees.

feature selection. In Pfahringer's approach, the minimum description length (MDL) principle [Ris78] guides the search by estimating the cost of encoding a decision table and the training examples it misclassifies with respect to a given feature subset. The features appearing in the final decision table are then used with other learning algorithms. Experiments on a small selection of machine learning datasets showed that feature selection by DTM induction can improve the accuracy of C4.5 in some cases. DTM classifiers induced using MDL were also compared with those induced using cross-validation (a wrapper approach) to estimate the accuracy of tables (and hence feature sets). The MDL approach was shown to be more efficient than, and perform as well as, as cross-validation.

3.4.4 An Information Theoretic Feature Filter

Koller and Sahami [KS96b] recently introduced a feature selection algorithm based on ideas from information theory and probabilistic reasoning [Pea88]. The rationale behind their approach is that, since the goal of an induction algorithm is to estimate the probability distributions over the class values, given the original feature set, feature subset selection should attempt to remain as close to these original distributions as possible. More formally, let C be a set of classes, V a set of features, X a subset of V, v an assignment of values $(v_1, \ldots, v_n)$ to the features in V, and v_x the projection of the values in v onto the variables in X. The goal of the feature selector is to choose X so that $\Pr(C|X = v_x)$ is as close as possible to $\Pr(C|V = v)$. To achieve this goal, the algorithm begins with all the original features and employs a backward elimination search to remove, at each stage, the feature that causes the least change between the two distributions. Because it is not reliable to estimate high order probability distributions from limited data, an approximate algorithm is given that uses pair-wise combinations of features. Cross entropy is used to measure the difference between two distributions and the user must specify how many features are to be removed by the algorithm. The cross entropy of the class distribution given a pair of features is:

$$\begin{aligned} &D\Big(\Pr(C|V_i = v_i, V_j = v_j), \Pr(C|V_j = v_j)\Big) = \\ &\sum_{c \in C} p(c|V_i = v_i, V_j = v_j) \log_2 \frac{p(c|V_i = v_i, V_j = v_j)}{p(c|V_j = v_j)}. \end{aligned} \tag{3.7}$$

For each feature i, the algorithm finds a set M_i, containing K attributes from those that remain, that is likely to subsume[6] the information feature i has about the class values. M_i contains K features out of the remaining features for which the value of Equation 3.7 is smallest. The expected cross entropy between the distribution of the class values, given M_i, V_i, and the distribution of class values given just M_i, is calculated for each feature i. The feature for which this quantity is minimal is removed from the set. This process iterates until the user-specified number of features are removed from the original set.

Experiments on four natural domains and two artificial domains using C4.5 and naive Bayes as the final induction algorithm, showed that the feature selector gives the best results when the size K of the conditioning set M is set to 2. In two domains containing over 1000 features the algorithm is able to reduce the number of features by more than half, while improving accuracy by one or two percent.

One problem with the algorithm is that it requires features with more than two values to be encoded as binary in order to avoid the bias that entropic measures have toward features with many values. This can greatly increase the number of features in the original data, as well as introducing further dependencies. Furthermore, the meaning of the original attributes is obscured, making the output of algorithms such as C4.5 hard to interpret.

3.4.5 An Instance Based Approach to Feature Selection

Kira and Rendell [KR92] describe an algorithm called RELIEF that uses instance based learning to assign a relevance weight to each feature. Each feature's weight reflects its ability to distinguish among the class values. Features are ranked by weight and those that exceed a user-specified threshold are selected to form the final subset. The algorithm works by randomly sampling instances from the training data. For each instance sampled, the nearest instance of the same class (nearest hit) and opposite class (nearest miss) is found. An attribute's weight is updated according to how well its values distinguish the sampled instance from its nearest hit and nearest miss. An attribute will receive a high weight if it differentiates between instances from different classes and has the same value for instances of the same class. Equation 3.8 shows the weight updating formula used by

[6] M_i is an approximation of a *markov blanket*[Pea88] for feature i.

RELIEF:

$$W_X = W_X - \frac{\text{diff}(X, R, H)^2}{m} + \frac{\text{diff}(X, R, M)^2}{m}, \tag{3.8}$$

where W_X is the weight for attribute X, R is a randomly sampled instance, H is the nearest hit, M is the nearest miss, and m is the number of randomly sampled instances. The function diff calculates the difference between two instances for a given attribute. For nominal attributes it is defined as either 1 (the values are different) or 0 (the values are the same), while for continuous attributes the difference is the actual difference normalised to the interval $[0, 1]$. Dividing by m guarantees that all weights are in the interval $[-1, 1]$.

RELIEF operates on two-class domains. Kononenko [Kon94] describes enhancements to RELIEF that enable it to cope with multi-class, noisy and incomplete domains. Kira and Rendell provide experimental evidence that shows RELIEF to be effective at identifying relevant features even when they interact[7] (for example, in parity problems). However, RELIEF does not handle redundant features. The authors state:

> "If most of the given features are relevant to the concept, it (RELIEF) would select most of the given features even though only a small number of them are necessary for concept description."

Scherf and Brauer [SB97] describe a similar instance based approach (EUBAFES) to assigning feature weights developed independently of RELIEF. Like RELIEF, EUBAFES strives to reinforce similarities between instances of the same class while simultaneously decrease similarities between instances of different classes. A gradient descent approach is employed to optimize feature weights with respect to this goal.

3.5 Feature Wrappers

Wrapper strategies for feature selection use an induction algorithm to estimate the merit of feature subsets. The rationale for wrapper approaches is that the induction method that

[7] *Interacting* features are those whose values are dependent on the values of other features and the class, and as such, provide further information about the class. On the other hand, *redundant* features, are those whose values are dependent on the values of other features irrespective of the class, and as such, provide no further information about the class.

will ultimately use the feature subset should provide a better estimate of accuracy than a separate measure that has an entirely different inductive bias [Lan94]. Feature wrappers often achieve better results than filters due to the fact that they are tuned to the specific interaction between an induction algorithm and its training data. However, they tend to be much slower than feature filters because they must repeatedly call the induction algorithm and must be re-run when a different induction algorithm is used. Since the wrapper is a well defined process, most of the variation in its application are due to the method used to estimate the off-sample accuracy of a target induction algorithm, the target induction algorithm itself, and the organisation of the search. This section reviews work that has focused on the wrapper approach and methods to reduce its computational expense.

3.5.1 Wrappers for Decision Tree Learners

John, Kohavi, and Pfleger [JKP94] were the first to advocate the wrapper [All74] as a general framework for feature selection in machine learning. They present formal definitions for two degrees of feature relevance, and claim that the wrapper is able to discover relevant features. A feature X_i is said to be strongly relevant to the target concept(s) if the probability distribution of the class values, given the full feature set, changes when X_i is removed. A feature X_i is said to be weakly relevant if it is not strongly relevant and the probability distribution of the class values, given some subset S (containing X_i) of the full feature set, changes when X_i is removed. All features that are not strongly or weakly relevant are irrelevant. Experiments were conducted on three artificial and three natural domains using ID3 and C4.5 [Qui86, Qui93] as the induction algorithms. Accuracy was estimated by using 25-fold cross validation on the training data; a disjoint test set was used for reporting final accuracies. Both forward selection and backward elimination search were used. With the exception of one artificial domain, results showed that feature selection did not significantly change ID3 or C4.5's generalisation performance. The main effect of feature selection was to reduce the size of the trees.

Like John et al., Caruanna and Freitag [CF94] test a number of greedy search methods with ID3 on two calendar scheduling domains. As well as backward elimination and forward selection they also test two variants of stepwise bi-directional search—one starting with all features, the other with none. Results showed that although the bi-directional searches slightly outperformed the forward and backward searches, on the whole there was very little difference between the various search strategies except with respect to computation time. Feature selection was able to improve the performance of ID3 on both calendar scheduling domains.

Vafaie and De Jong [VJ95] and Cherkauer and Shavlik [CS96] have both applied genetic search strategies in a wrapper framework for improving the performance of decision tree learners. Vafaie and De Jong [VJ95] describe a system that has two genetic algorithm driven modules—the first performs feature selection, and the second performs constructive induction[8] [Mic83]. Both modules were able to significantly improve the performance of ID3 on a texture classification problem. Cherkauer and Shavlik [CS96] present an algorithm called SET-Gen which strives to improve the comprehesibility of decision trees as well as their accuracy. To achive this, SET-Gen's genetic search uses a fitness function that is a linear combination of an accuracy term and a simplicity term:

$$\text{Fitness}(X) = \frac{3}{4}A + \frac{1}{4}\left(1 - \frac{S + F}{2}\right), \tag{3.9}$$

where X is a feature subset, A is the average cross-validation accuracy of C4.5, S is the average size of the trees produced by C4.5 (normalized by the number of training examples), and F is is the number of features is the subset X (normalized by the total number of available features).

Equation 3.9 ensures that the fittest population members are those feature subsets that lead C4.5 to induce small but accurate decision trees.

3.5.2 Wrappers for Instance Based Learning

The wrapper approach was proposed at approximately the same time and independently

[8] Constructive induction is the process of creating new attributes by applying logical and mathematical operators to the original features.

of John et al. by Langley and Sage [LS94c, LS94b] during their investigation of the simple nearest neighbour algorithm's sensitivity to irrelevant attributes. Scaling experiments showed that the nearest neighbour's sample complexity (the number of training examples needed to reach a given accuracy) increases exponentially with the number of irrelevant attributes present in the data [AKA91, LS94c, LS94b]. An algorithm called OBLIVION is presented which performs backward elimination of features using an oblivious decision tree[9] as the induction algorithm. Experiments with OBLIVION using k-fold cross validation on several artificial domains showed that it was able to remove redundant features and learn faster than C4.5 on domains where features interact.

Moore and Lee [ML94] take a similar approach to augmenting nearest neighbour algorithms, but their system uses leave-one-out instead of k-fold cross-validation and concentrates on improving the prediction of numeric rather than discrete classes. Aha and Blankert [AB94] also use leave-one-out cross validation, but pair it with a beam search[10] instead of hill climbing. Their results show that feature selection can improve the performance of IB1 (a nearest neighbour classifier) on a sparse (very few instances) cloud pattern domain with many features. Moore, Hill, and Johnson [MHJ92] encompass not only feature selection in the wrapper process, but also the number of nearest neighbours used in prediction and the space of combination functions. Using leave-one-out cross validation, they achieve significant improvement on several control problems involving the prediction of continuous classes. In a similar vein, Skalak [Ska94] combines feature selection and prototype selection into a single wrapper process using random mutation hill climbing as the search strategy. Experimental results showed significant improvement in accuracy for nearest neighbour on two natural domains and a drastic reduction in the algorithm's storage requirement (number of instances retained during training).

Domingos [Dom97] describes a *context sensitive* wrapper approach to feature selection for instance based learners. The motivation for the approach is that there may be features that are either relevant in only a restricted area of the instance space and irrelevant

[9] When all the original features are included in the tree and given a number of assumptions at classification time, Langley and Sage note that the structure is functionally equivalent to the simple nearest neighbour; in fact, this is how it is implemented in OBLIVION.

[10] Beam search is a limited version of best first search that only remembers a portion of the search path for use in backtracking

elsewhere, or relevant given only certain values (weakly interacting) of other features and otherwise irrelevant. In either case, when features are estimated *globally* (over the entire instance space), the irrelevant aspects of these sorts of features may overwhelm their useful aspects for instance based learners. This is true even when using backward search strategies with the wrapper[11]. Domingos presents an algorithm called RC which can detect and make use of context sensitive features. RC works by selecting a (potentially) different set of features for each instance in the training set. It does this by using a backward search strategy and cross validation to estimate accuracy. For each instance in the training set, RC finds its nearest neighbour of the same class and removes those features in which the two differ. The accuracy of the entire training dataset is then estimated by cross validation. If the accuracy has not degraded, the modified instance in question is accepted; otherwise the instance is restored to its original state and *deactivated* (no further feature selection is attempted for it). The feature selection process continues until all instances are inactive.

Experiments on a selection of machine learning datasets showed that RC outperformed standard wrapper feature selectors using forward and backward search strategies with instance based learners. The effectiveness of the context sensitive approach was also shown on artificial domains engineered to exhibit restricted feature dependency. When features are globally relevant or irrelevant, RC has no advantage over standard wrapper feature selection. Furthermore, when few examples are available, or the data is noisy, standard wrapper approaches can detect globally irrelevant features more easily than RC. Domingos also noted that wrappers that employ instance based learners (including RC) are unsuitable for use on databases containing many instances because they are quadratic in N (the number of instances).

Kohavi [KF94, Koh95a] uses wrapper feature selection to explore the potential of decision table majority (DTM) classifiers. Appropriate data structures allow the use of fast incremental cross-validation with DTM classifiers. Experiments showed that DTM classifiers using appropriate feature subsets compared very favourably with sophisticated algorithms

[11] In the wrapper approach, backward search strategies are generally more effective than forward search strategies in domains with feature interactions. Because backward search typically begins with all the features, the removal of a strongly interacting feature is usually detected by decreased accuracy during cross validation.

such as C4.5.

3.5.3 Wrappers for Bayes Classifiers

Due to the naive Bayes classifier's assumption that, within each class, probability distributions for attributes are independent of each other, Langley and Sage [LS94a] note that its performance on domains with redundant features can be improved by removing such features. A forward search strategy is employed to select features for use with naive Bayes, as opposed to the backward strategies that are used most often with decision tree algorithms and instance based learners. The rationale for a forward search is that it should immediately detect dependencies when harmful redundant attributes are added. Experiments showed overall improvement and increased learning rate on three out of six natural domains, with no change on the remaining three.

Pazzani [Paz95] combines feature selection and simple constructive induction in a wrapper framework for improving the performance of naive Bayes. Forward and backward hill climbing search strategies are compared. In the former case, the algorithm considers not only the addition of single features to the current subset, but also creating a new attribute by joining one of the as yet unselected features with each of the selected features in the subset. In the latter case, the algorithm considers both deleting individual features and replacing pairs of features with a joined feature. Results on a selection of machine learning datasets show that both approaches improve the performance of naive Bayes. The forward strategy does a better job at removing redundant attributes than the backward strategy. Because it starts with the full set of features, and considers all possible pairwise joined features, the backward strategy is more effective at identifying attribute interactions than the forward strategy.

Improvement for naive Bayes using wrapper-based feature selection is also reported by Kohavi and Sommerfield [KS95] and Kohavi and John [KJ96].

Provan and Singh [PS96] have applied the wrapper to select features from which to construct Bayesian networks. Their results showed that while feature selection did not improve accuracy over networks constructed from the full set of features, the networks constructed after feature selection were considerably smaller and faster to learn.

3.5.4 Methods of Improving the Wrapper

Most criticism of the wrapper approach to feature selection is concerned with its computational cost. For each feature subset examined, an induction algorithm is invoked k times in an k-fold cross validation. This can make the wrapper prohibitively slow for use on large data sets with many features. This drawback has led some researchers to investigate ways of mitigating the cost of the evaluation process.

Caruanna and Freitag [CF94] devised a scheme that caches decision trees. This can substantially reduce the number of trees grown during feature selection and allow larger spaces to be searched.

Moore and Lee [ML94] present a method to "race" competing models or feature subsets. If at some point during leave-one-out cross-validation, a subset is deemed to be unlikely to have the lowest estimated error, its evaluation is terminated. This has the effect of reducing the percentage of training examples used during evaluation and reduces the computational cost of fully evaluating each subset. The algorithm also "blocks" all near identical feature subsets—except one—in the race. This prevents having to run feature subsets with nearly identical predictions right to the end. Both racing and blocking use Bayesian statistics to maintain a probability distribution on the estimate of the mean leave-one-out cross validation error for each competing subset. The algorithm uses forward selection, but instead of sequentially trying all local changes to the best subset, these changes are raced. The race finishes when only one competing subset remains or the cross validation ends.

Kohavi and John [KS95] introduce the notion of "compound" search space operators in an attempt to make backward and best first search strategies computationally feasible. When all local changes (additions or deletions of single features) to a given feature subset have been evaluated, the first compound operator is created, combining the two best local changes. This operator is then applied to the feature subset, creating a new subset further away in the search space. If the first compound operator leads to a subset with an improved estimate, a second compound operator is constructed that combines the best three local changes, and so forth. The use of compound operators propels the search more quickly toward the strongly relevant features. Experiments using compound operators with a forward best first search showed no significant change in the accuracy for ID3 and naive Bayes. When compound operators were combined with a backward best first search, accuracy degraded slightly for ID3 but improved for C4.5. The poor results with ID3 suggest that the best first search can still get stuck in some local maxima. The improvement with C4.5 is due to C4.5's pruning (again a form of feature selection), which allows the best first search to overcome the local maxima.

Moore and Lee [ML94] describe another search variant called schemata search that takes interacting features into account and speeds up the search process. Rather than starting with an empty or full set of features, the search begins with all features marked as "unknown". In each iteration, a feature is chosen and raced between being in the subset or excluded from it. All combinations of unknown features are used with equal probability. Due to the probabilistic nature of the search, a feature that should be in the subset will win the race, even if it is dependent on another feature. Experiments on artificial domains showed schemata search to be effective at identifying relevant features (more so than raced versions of forward and backward selection) and much faster than raced backward selection.

3.6 Feature Weighting Algorithms

Feature weighting can be viewed as a generalisation of feature selection. In feature selection, feature weights are restricted to 0 or 1 (a feature is used or it is not). Feature weighting allows finer differentiation between features by assigning each a continuous valued weight. Algorithms such as nearest neighbour (that normally treat each feature equally) can be easily modified to include feature weighting when calculating similarity between cases. One thing to note is that, in general, feature weighting algorithms do not reduce the dimensionality of the data. Unless features with very low weight are removed from the data initially, it is assumed that each feature is useful for induction; its degree of usefulness is reflected in the magnitude of its weight. Using continuous weights for features involves searching a much larger space and involves a greater chance of overfitting [KLY97].

Salzberg [Sal91] incorporates incremental feature weighting in an instance based learner called EACH. For each correct classification made, the weight for each matching feature is incremented by Δ_f (the global feature adjustment rate). Mismatching features have their weights decremented by this same amount. For incorrect classifications, the opposite occurs—mismatching features are incremented while the weights of matching features are decremented. Salzberg reported that the value of Δ_f needs to be tuned for different data sets to give best results.

Wettschereck and Aha [WA95] note that EACH's weighting scheme is insensitive to skewed concept descriptions. IB4 [Aha92] is an extension of the k nearest neighbour algorithm that addresses this problem by calculating a separate set of feature weights for each concept. The weight for feature i is computed using

$$w_i = \max\left(\frac{\text{CumulativeWeight}_i}{\text{WeightNormaliser}_i} - 0.5, 0\right). \tag{3.10}$$

CumulativeWeight is expected to approach one half of WeightNormaliser for apparently irrelevant attributes. Both CumulativeWeight and WeightNormaliser are incrementally updated during learning. Let Λ be the higher of the observed frequencies among the classes of two instances X (the instance to be classified) and Y (its most similar neighbour in the concept description). CumulativeWeight$_i$ is incremented by

$$\begin{aligned} &1 - \text{diff}(x_i, y_i) \times (1 - \Lambda) \text{ if } X \text{ and } Y \text{ have the same class,} \\ &\text{diff}(x_i, y_i) \times (1 - \Lambda) \text{ otherwise.} \end{aligned} \tag{3.11}$$

WeightNormaliser is always incremented by $(1 - \Lambda)$. Experiments with IB4 showed it to be more tolerant of irrelevant features than the k nearest neighbour algorithm.

RELIEF[12] [KR92] is an algorithm that uses an instance based approach to assign weights to features. Wettschereck and Aha [WA95] use RELIEF to calculate weights for a k

[12] RELIEF was originally used for feature selection and is described in section 2.5.5

nearest neighbour algorithm—they report significant improvement over standard k nearest neighbour in seven out of ten domains.

Kohavi, Langley, and Yun [KLY97] describe an approach to feature weighting that considers a small set of discrete weights rather than continuous weights. Their approach uses the wrapper coupled with simple nearest neighbour to estimate the accuracy of feature weights and a best first search to explore the weight space. In experiments that vary the number of discrete weights considered by the algorithm, results showed that there is no advantage to increasing the number of non-zero discrete weights above two; in fact, with the exception of some carefully crafted artificial domains, using one non-zero weight (equivalent to feature selection) was difficult to outperform.

The above methods for feature weighting all use feedback from a nearest neighbour algorithm (either incrementally during learning or in a special stage prior to induction) to adjust weights. Some non-feedback methods for setting weights include: the *per category feature importance* [CMSW92] which sets the weight for a feature to the conditional probability of the class given the feature, the *cross-category feature importance* [WA95], which is like the per category feature importance but averages across classes, and the *mutual information*[13] [SW48] between the feature and the class. All of these approaches require numeric features to be discretized.

3.7 Chapter Summary

Practical machine learning algorithms often make assumptions or apply heuristics that trade some accuracy of the resulting model for speed of execution, and comprehensibility of the result. While these assumptions and heuristics are reasonable and often yield good results, the presence of *irrelevant* and *redundant* information can often fool them, resulting in reduced accuracy and less understandable results. Feature subset selection can help focus the learning algorithm on the important features for a particular problem. It can also reduce the dimensionality of the data, allowing learning algorithms to operate faster and more effectively.

[13] This is also known as the information *gain* between the feature and the class. See Chapter 3 for details.

There are two main approaches to feature subset selection described in the literature. The *wrapper*—which is tuned to the specific interaction between an induction algorithm and its training data—has been shown to give good results, but in practise may be too slow to be of practical use on large real-world domains containing many features. *Filter* methods are much faster as they do not involve repeatedly invoking a learning algorithm. Existing filter solutions exhibit a number of drawbacks. Some algorithms are unable to handle noise (or rely on the user to specify the level of noise for a particular problem). In some cases, a subset of features is not selected explicitly; instead, features are ranked with the final choice left to the user. Some algorithms do not handle both redundant and irrelevant features. Other algorithms require features to be transformed in such a way that actually increases the initial number of features and hence the search space. This last case can result in a loss of meaning from the original representation, which in turn can have an impact on the interpretation of induced models.

Feature weights are easily incorporated into learning algorithms such as nearest neighbour, but the advantage of feature weighting over feature selection is minimal at best, due to the increased chance of overfitting the data. In general, feature weighting does not reduce the dimensionality of the original data.

Chapter 4

Correlation-based Feature Selection

This thesis claims that feature selection for classification tasks in machine learning can be accomplished on the basis of correlation[1] between features, and that such a feature selection procedure can be beneficial to common machine learning algorithms. This chapter presents a correlation based feature selector (CFS) based on this claim; subsequent chapters examine the behaviour of CFS under various conditions and show that CFS can identify useful features for machine learning.

Section 4.1 outlines the rationale and motivation for a correlation-based approach to feature selection, with ideas borrowed from psychological measurement theory. Various machine learning approaches to measuring correlation between nominal variables are discussed in Section 4.2; their respective biases and implications for use with CFS are discussed in Section 4.3. Section 4.4 presents the CFS algorithm and the variations used for experimental purposes.

4.1 Rationale

Genari *et al.* [GLF89] state that

> "Features are relevant if their values vary systematically with category membership."

[1] The term correlation is used in its general sense in this thesis. It is not intended to refer specifically to classical linear correlation; rather it is used to refer to a degree of dependence or predictability of one variable with another.

In other words, a feature is useful if it is correlated with or predictive of the class; otherwise it is irrelevant. Kohavi and John [KJ96] formalize this definition as

Definition 1: A feature V_i is said to be relevant *iff* there exists some v_i and c for which $p(V_i = v_i) > 0$ such that

$$p(C = c|V_i = v_i) \neq p(C = c). \tag{4.1}$$

Empirical evidence from the feature selection literature shows that, along with irrelevant features, redundant information should be eliminated as well [LS94a, KJ96, KS95]. A feature is said to be redundant if one or more of the other features are highly correlated with it. The above definitions for relevance and redundancy lead to the following hypothesis, on which the feature selection method presented in this thesis is based:

> *A good feature subset is one that contains features highly correlated with (predictive of) the class, yet uncorrelated with (not predictive of) each other.*

In test theory [Ghi64], the same principle is used to design a composite test for predicting an external variable of interest. In this situation, the "features" are individual tests which measure traits related to the variable of interest (class). For example, a more accurate prediction of a person's success in a mechanics training course can be had from a composite of a number of tests measuring a wide variety of traits (ability to learn, ability to comprehend written material, manual dexterity and so forth), rather than from any one individual test which measures a restricted scope of traits. Ghiselli states:

> "When we develop a composite which we intend to use as a basis for predicting an outside variable, it is likely that the components we select to form the composite will have relatively low inter-correlations. When we seek to predict some variable from several other variables, we try to select predictor variables which measure different aspects of the outside variable."

If the correlation between each of the components in a test and the outside variable is known, and the inter-correlation between each pair of components is given, then the correlation between a composite test consisting of the summed components and the outside variable can be predicted from

$$r_{zc} = \frac{k\overline{r_{zi}}}{\sqrt{k + k(k-1)\overline{r_{ii}}}}, \tag{4.2}$$

where r_{zc} is the correlation between the summed components and the outside variable, k is the number of components, $\overline{r_{zi}}$ is the average of the correlations between the components and the outside variable, and $\overline{r_{ii}}$ is the average inter-correlation between components [Ghi64, Hog77, Zaj62].

Equation 4.2 is, in fact, Pearson's correlation coefficient, where all variables have been standardized. It shows that the correlation between a composite and an outside variable is a function of the number of component variables in the composite and the magnitude of the inter-correlations among them, together with the magnitude of the correlations between the components and the outside variable. Entering two illustrative values for $\overline{r_{zi}}$ in Equation 4.2, and allowing the values of k and $\overline{r_{ii}}$ to vary, the formula is solved for $\overline{r_{zc}}$ and the values are plotted in Figure 4.1. From this figure the following conclusions can be drawn:

- The higher the correlations between the components and the outside variable, the higher the correlation between composite and the outside variable.
- The lower the inter-correlations among the components, the higher the correlation between the composite and the outside variable.
- As the number of components in the composite increases (assuming the additional components are the same as the original components in terms of their average inter-correlation with the other components and with the outside variable), the correlation between the composite and the outside variable increases.

From Figure 4.1, it can be seen that increasing the number of components substantially increases the correlation between the composite and the outside variable. However, it is unlikely that a group of components that are all highly correlated with the outside

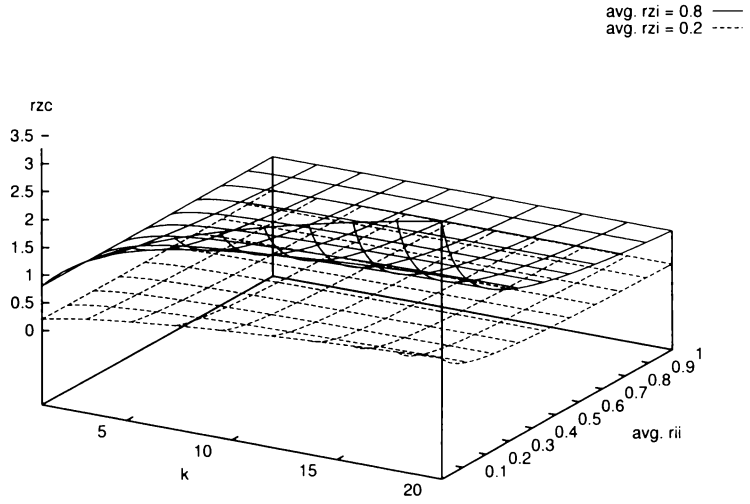


Figure 4.1: The effects on the correlation between an outside variable and a composite variable (r_{zc}) of the number of components (k), the inter-correlations among the components $(\overline{r_{ii}})$, and the correlations between the components and the outside variable $(\overline{r_{zi}})$.

variable will at the same time bear low correlations with each other [Ghi64]. Furthermore, Hogarth [Hog77] notes that, when addition of an additional component is considered, low inter-correlation with the already selected components may well predominate over high correlation with the outside variable.

Equation 4.2 is used in this thesis as a heuristic measure of the "merit" of feature subsets in supervised classification tasks. In this situation, z (the external variable) becomes C (the class); the problem remaining is to develop suitable ways of measuring the feature-class correlation and feature-feature inter-correlation. Supervised learning tasks often involve different data features, any of which may be continuous, ordinal, nominal, or binary. In order to have a common basis for computing the correlations in Equation 4.2, it is desirable to have a uniform way of treating different types of features. Discretization using the method of Fayyad and Irani [FI93] is applied as a pre-processing step to convert continuous features to nominal.

For prediction it is clear that redundant attributes should be eliminated—if a given feature's predictive ability is covered by another then it can safely be removed. Indeed, some learning algorithms (such as naive Bayes) require this in order to maximise predictive performance [LS94a]. However, for data mining applications where comprehensible results are of paramount importance, it is not always clear that redundancy should be eliminated. For example, a rule may make more "sense" to a user if an attribute is replaced with one highly correlated with it. CFS (described in section 4.4) accommodates this situation by providing a report generation facility. For any given attribute in the final subset, CFS can list its close substitutes, either in terms of the overall merit of the final subset if the attribute in question was to be replaced by one of the substitutes, or simply correlation with the attribute in question.

4.2 Correlating Nominal Features

Once all features and the class are treated in a uniform manner, the feature-class correlation and feature-feature inter-correlations in Equation 4.2 may be calculated. Research on decision tree induction has provided a number of methods for estimating the quality of an attribute—that is, how predictive one attribute is of another. Measures of attribute quality characterize the variability present in the collections of instances corresponding to the values of a particular attribute. For this reason they are sometimes known as impurity functions [Bre96b, CB97]. A collection of instances is considered *pure* if each instance is the same with respect to the value of a second attribute; the collection of instances is *impure* (to some degree) if instances differ with respect to the value of the second attribute. Decision tree induction typically only involves measuring how predictive attributes are of the class. This corresponds to the feature-class correlations in Equation 4.2. To calculate the merit of a feature subset using Equation 4.2, feature-feature inter-correlations—the ability of one feature to predict another (and vice versa)—must be measured as well.

Because decision tree learners perform a greedy simple-to-complex hill climbing search through the space of possible trees, their general inductive bias is to favour smaller trees over larger ones [Mit97]. One factor that can impact on both the size of the tree and how it well it generalizes to new instances is the bias inherent in the attribute quality measure used to select among attributes to test at the nodes of the tree. Some quality measures are known to unfairly favour attributes with more values over those with fewer values [Qui86, WL94, Kon95]. This can result in the construction of larger trees that may overfit the training data and generalize poorly. Similarly, if such measures are used as the correlations in Equation 4.2, feature subsets containing features with more values may be preferred—a situation that could lead to inferior performance by a decision tree learner if it is restricted to using such a subset.

Kononenko [Kon95] examines the biases of eleven measures for estimating the quality of attributes. Two of these, *relief* and MDL, with the most acceptable biases with respect to attribute level (number of values), are described in this section. For the inter-correlation between two features, a measure is needed that characterizes the predictive ability of one attribute for another and vice versa. Simple symmetric versions of *relief* and MDL are presented for this purpose. A third measure (not tested by Kononenko), symmetrical uncertainty [PFTV88], with bias similar to *relief* and MDL, is also presented.

Section 4.3 reconstructs experiments done by Kononenko to analyze the bias of attribute quality measures. The behaviour of symmetrical uncertainty, MDL, and *relief* with respect to attribute level and how this may affect feature selection is discussed. The experimental scenario is extended to examine the behaviour of the measures with respect to the number of available training examples; again implications for feature selection are discussed.

Versions of the CFS feature selector using *relief*, MDL, and symmetric uncertainty are empirically compared in Chapter 6.

4.2.1 Symmetrical Uncertainty

A probabilistic model of a nominal valued feature Y can be formed by estimating the individual probabilities of the values $y \in Y$ from the training data. If this model is used to estimate the value of Y for a novel sample (drawn from the same distribution as the training data), then the entropy of the model (and hence of the attribute) is the number of bits it would take, on average, to correct the output of the model. Entropy is a measure of the *uncertainty* or unpredictability in a system. The entropy of Y is given by

$$H(Y) = -\sum_{y \in Y} p(y) \log_2(p(y)). \tag{4.3}$$

If the observed values of Y in the training data are partitioned according to the values of a second feature X, and the entropy of Y with respect to the partitions induced by X is less than the entropy of Y prior to partitioning, then there is a relationship between features Y and X. Equation 4.4 gives the entropy of Y after observing X.

$$H(Y|X) = -\sum_{x \in X} p(x) \sum_{y \in Y} p(y|x) \log_2(p(y|x)). \tag{4.4}$$

The amount by which the entropy of Y decreases reflects additional information about Y provided by X and is called the *information gain* [Qui86], or, alternatively, *mutual information* [SW48]. Information gain is given by

$$\begin{aligned} \text{gain} &= H(Y) - H(Y|X) \qquad (4.5) \\ &= H(X) - H(X|Y) \\ &= H(Y) + H(X) - H(X,Y). \end{aligned}$$

Information gain is a symmetrical measure—that is, the amount of information gained about Y after observing X is equal to the amount of information gained about X after observing Y. Symmetry is a desirable property for a measure of feature-feature intercorrelation to have. Unfortunately, information gain is biased in favour of features with more values. Furthermore, the correlations in Equation 4.2 should be normalized to ensure they are comparable and have the same affect. Symmetrical uncertainty [PFTV88] compensates for information gain's bias toward attributes with more values and normalizes its value to the range $[0, 1]$:

$$\text{symmetrical uncertainty coefficient} = 2.0 \times \left[\frac{\text{gain}}{H(Y) + H(X)}\right]. \tag{4.6}$$

4.2.2 *Relief*

RELIEF [KR92] is a feature weighting algorithm that is sensitive to feature interactions (see Chapters 3 and 8 for details). Kononenko [Kon95] notes that RELIEF attempts to approximate the following difference of probabilities for the weight of a feature X:

$$\begin{aligned} W_X &= P(\text{different value of } X | \text{ nearest instance of different class}) \\ &- P(\text{different value of } X | \text{ nearest instance of same class}). \end{aligned} \tag{4.7}$$

By removing the context sensitivity provided by the "nearest instance" condition, attributes are treated as independent of one another; Equation 4.8 then becomes [Kon94, Kon95]

$$\begin{aligned} \mathit{Relief}_X &= P(\text{different value of } X | \text{ different class}) \\ &- P(\text{different value of } X | \text{ same class}), \end{aligned} \tag{4.8}$$

which can be reformulated as

$$\mathit{Relief}_X = \frac{Gini' \times \sum_{x \in X} p(x)^2}{(1 - \sum_{c \in C} p(c)^2) \sum_{c \in C} p(c)^2}, \tag{4.9}$$

where C is the class variable and

$$Gini' = \left[\sum_{c \in C} p(c)(1 - p(c))\right] - \sum_{x \in X} \left(\frac{p(x)^2}{\sum_{x \in X} p(x)^2} \sum_{c \in C} p(c|x)(1 - p(c|x)) \right). \tag{4.10}$$

$Gini'$ is a modification of another attribute quality measure called the Gini-index[2] [Bre96b]. Both $Gini'$ and the Gini-index are similar to information gain in that they are biased in favour of attributes with more values.

To use *relief* symmetrically for two features, the measure can be calculated twice (each feature is treated in turn as the "class"), and the results averaged. Whenever *relief* is mentioned in subsequent chapters, it is the symmetrical version that is referred to.

[2] The only difference to Equation 4.10 is that the Gini-index uses $p(x)$ in place of $p(x)^2 / \sum p(x)^2$.

4.2.3 MDL

Roughly speaking, the minimum description length (MDL) principle [Ris78] states that the "best" theory to infer from training data is the one that minimizes the length (complexity) of the theory and the length of the data encoded with respect to the theory. The MDL principle can be taken as an operational definition of Occam's Razor[3]. More formally, if T is a theory inferred from data D, then the total description length is given by

$$\mathrm{DL}(T, D) = \mathrm{DL}(T) + \mathrm{DL}(D|T). \tag{4.11}$$

In Equation 4.11, all description lengths are measured in bits. If the data (D) is the observed values of a feature Y, and these values are partitioned according to the values of a second feature X, then the description length of the data given the theory (the second term in Equation 4.11) can be approximated by multiplying the average entropy of Y given X by the number of observed instances.

One problem with just using entropy to measure the quality of a model (and hence an attribute) is that it is possible to construct a model that predicts the data perfectly, and as such has zero entropy. Such a model is not necessarily as good as it seems. For example, consider an attribute X that has as many distinct values as there are instances in the data. If the data is partitioned according to the values of X, then there will be exactly one value of Y (with probability 1) in each of these partitions, and the entropy of Y with respect to X will be zero. However, a model such as this is unlikely to generalize well to new data; it has *overfitted* the training data—that is, it is overly sensitive to statistical idiosyncrasies of the training data. The first term in Equation 4.11 deals with just this sort of problem. A model such as the one just described is very complex and would take many bits to describe. So although the model has reduced the description length of the data to zero, the value of Equation 4.11 would still be large due to the high cost of describing the model. The best models (according to the MDL principle) are those which are predictive of the data and, at the same time, have captured the underlying structure

[3] The Occam's Razor principle, commonly attributed to William of Occam (early 14th century), states: "Entities should not be multiplied beyond necessity." This principle is generally interpreted as: "Given the choice between theories that are equally consistent with the observed phenomena, prefer the simplest".

in a compact fashion. Quinlan [Qui89] discusses the use of the MDL principle in coding decision trees; Kononenko [Kon95] defines an MDL measure of attribute quality:

$$\text{MDL} = \frac{(\text{Prior\_MDL} - \text{Post\_MDL})}{n} \tag{4.12}$$

$$\text{Prior\_MDL} = \log_2\binom{n}{n_1, \ldots, n_C} + \log_2\binom{n + C - 1}{C - 1} \tag{4.13}$$

$$\text{Post\_MDL} = \sum_j \log_2\binom{n_{.j}}{n_{1j}, \ldots, n_{Cj}} + \sum_j \log_2\binom{n_{.j} + C - 1}{C - 1}, \tag{4.14}$$

where n is the number of training instances, C is the number of class values, $n_{i.}$ is the number of training instances from class C_i, $n_{.j}$ is the number of training instances with the j-th value of the given attribute, and n_{ij} is the number of training instances of class C_i having the j-th value for the given attribute.

Equation 4.12 gives the average compression (per instance) of the class afforded by an attribute. Prior_MDL is the description length of the class labels prior to partitioning on the values of an attribute. Post_MDL performs the same calculation as Prior_MDL for each of the partitions induced by an attribute and sums the result. The first term of Equation 4.13 and Equation 4.14 encodes the class labels with respect to the model encoded in the respective second term. The model for Prior_MDL is simply a probability distribution over the class labels (that is, how many instances of each class are present); the model for Post_MDL is the probability distribution of the class labels in each of the partitions induced by the given attribute.

To obtain a measure that lies between 0 and 1, Equation 4.12 can be normalized by dividing by Prior_MDL$/n$. This gives the fraction by which the average description length of the class labels is reduced through partitioning on the values of an attribute. Equation 4.12 is a non-symmetric measure; exchanging the roles of the attribute and the class does not give the same result. To use the measure symmetrically for two features, it can be calculated twice (treating each feature in turn as the "class") and the results averaged. Whenever the MDL measure is mentioned in subsequent chapters, it is the normalized symmetrical version that is referred to.

4.3 Bias in Correlation Measures between Nominal Features

This section examines bias in the methods, discussed above, for measuring correlation between nominal features. Measures such as information gain tend to overestimate the worth of multi-valued attributes. This problem is well known in the decision tree community. Quinlan [Qui86] shows that the gain of an attribute A (measured with respect to the class or another feature) is less than or equal to the gain of an attribute A' formed by randomly partitioning A into a larger number of values. This means that, in general, the derived attribute (and by analogy, attributes with more values) will appear to be more predictive of or correlated with the class than the original one.

For example, suppose there is an attribute A with values a, b and there are two possible classes p, n (as shown in Table 4.1(a)). Given the eight instances shown in Table 4.1(a), the entropy of the class is 1.0 bit, the entropy of the class given attribute A is 1.0 bit, and the gain (calculated from Equation 4.6) is 0.0 (the attribute provides no further information about the class). If a second attribute A' is formed by converting 'b' values of attribute A into the value 'c' with probability 0.5, then the examples shown in Table 4.1(b) may occur. In this case, the entropy of the class with respect to attribute A' is 0.84 bits and the gain is 0.16 bits. However, since the the additional partitioning of A' was produced randomly, A' cannot be reasonably considered more correlated with the class than A.

One approach to eliminating this bias in decision tree induction is to construct only binary decision trees. This entails dividing the values of an attribute into two mutually exclusive subsets. Bias is now eliminated by virtue of all features having only two values. However, Quinlan [Qui86] notes that this process results in large increase in computation—for a given feature A with a values, at a given node in the tree, there are 2^a possible ways of subsetting the values of A, each of which must be evaluated in order to select the best. Some feature selection algorithms, such as the one described by Koller and Sa-

A	Class
a	p
a	p
a	n
a	n
b	p
b	p
b	n
b	n

(a)

A'	Class
a	p
a	p
a	n
a	n
b	p
b	p
b	n
c	n

(b)

Table 4.1: A two-valued non informative attribute A (a) and a three valued attribute A' derived by randomly partitioning A into a larger number of values (b). Attribute A' appears more predictive of the class than attribute A according to the information gain measure.

hami [KS96b], avoid bias in favour of multi-valued features by using a boolean encoding. Each value of an attribute A is represented by binary indicator attribute. For a given value of attribute A in a dataset, the appropriate indicator attribute is set to 1, and the indicator attributes corresponding to the other possible values of A are set to 0. This can greatly increase the number of features (and hence the size of the search space) and also introduce more dependencies into the data than were originally present. Furthermore, both subsetting in decision trees and boolean encoding for feature selection can result in less intelligible decision trees and a loss of meaning from the original attributes.

In the following sections the bias of symmetrical uncertainty, *relief*, and MDL is examined. The purpose of exploring bias in the measures is to obtain an indication as to how each measure will affect the heuristic "merit" calculated from Equation 4.2, and to get a feel for which measures exhibit bias similar to that employed by common learning algorithms. Each measure's behaviour with respect to irrelevant attributes is of particular interest for the feature selection algorithm presented in this thesis.

4.3.1 Experimental Measurement of Bias

To test bias in the various measures, the Monte Carlo simulation technique of White and Liu is adopted [WL94]. The technique approximates the distributions of the various measures under differing conditions (such as the number of attribute values and/or class values). Estimated parameters derived from these distributions can then be compared to see what effects (if any) these conditions have on the measures. White and Liu examined effects of attribute and class level on various measures using random (irrelevant) attributes generated independently of the class. Kononenko [Kon95] extended this scenario by including attributes predictive of the class. Section 4.3.2 examines the bias of symmetrical uncertainty, normalized symmetrical MDL, and symmetrical *relief* using the experimental methodology of Kononenko [Kon95]. Section 4.3.3 explores the effect of varying the sample size on the behaviour of the measures.

Method The experiments in this section use artificial data generated with the following properties:

- two, five or ten equiprobable classes;
- two, five, ten, twenty or forty attribute values;
- attributes are either irrelevant (have values drawn from the uniform distribution independently of the class), or are made informative using Kononenko's method [Kon95];
- 1000 training instances are used for the experiments in Section 4.3.2; the number of training instances is allowed to vary for the experiments in Section 4.3.3.

Multi-valued attributes are made informative by joining the values of the attribute into two subsets. If an attribute has a values, then subsets $\{1, \dots, (a \text{ div } 2)\}$ and $\{(a \text{ div } 2 + 1), \dots, a\}$ are formed. The probability that the attribute's value is from one of the subsets depends on the class; the selection of one particular value inside the subset is random from the uniform distribution. The probability that the attribute's value is in one of the subsets is given by

$$p\left(j \in \left\{1, \dots, \left(\frac{a}{2}\right)\right\} \middle| i\right) = \begin{cases} 1/(i + kC) & i \bmod 2 = 0 \\ 1 - 1/(i + kC) & i \bmod 2 \neq 0 \end{cases} \tag{4.15}$$

where C is the number of class values, i is an integer indexing the possible class values $\{c_1, \dots, c_i\}$, and k is a parameter controlling the level of association between the attribute and the class—higher values of k make the attribute more informative. From Equation 4.15 it can be seen that attributes are also more informative for higher numbers of classes. All experiments presented in this section use $k = 1$.

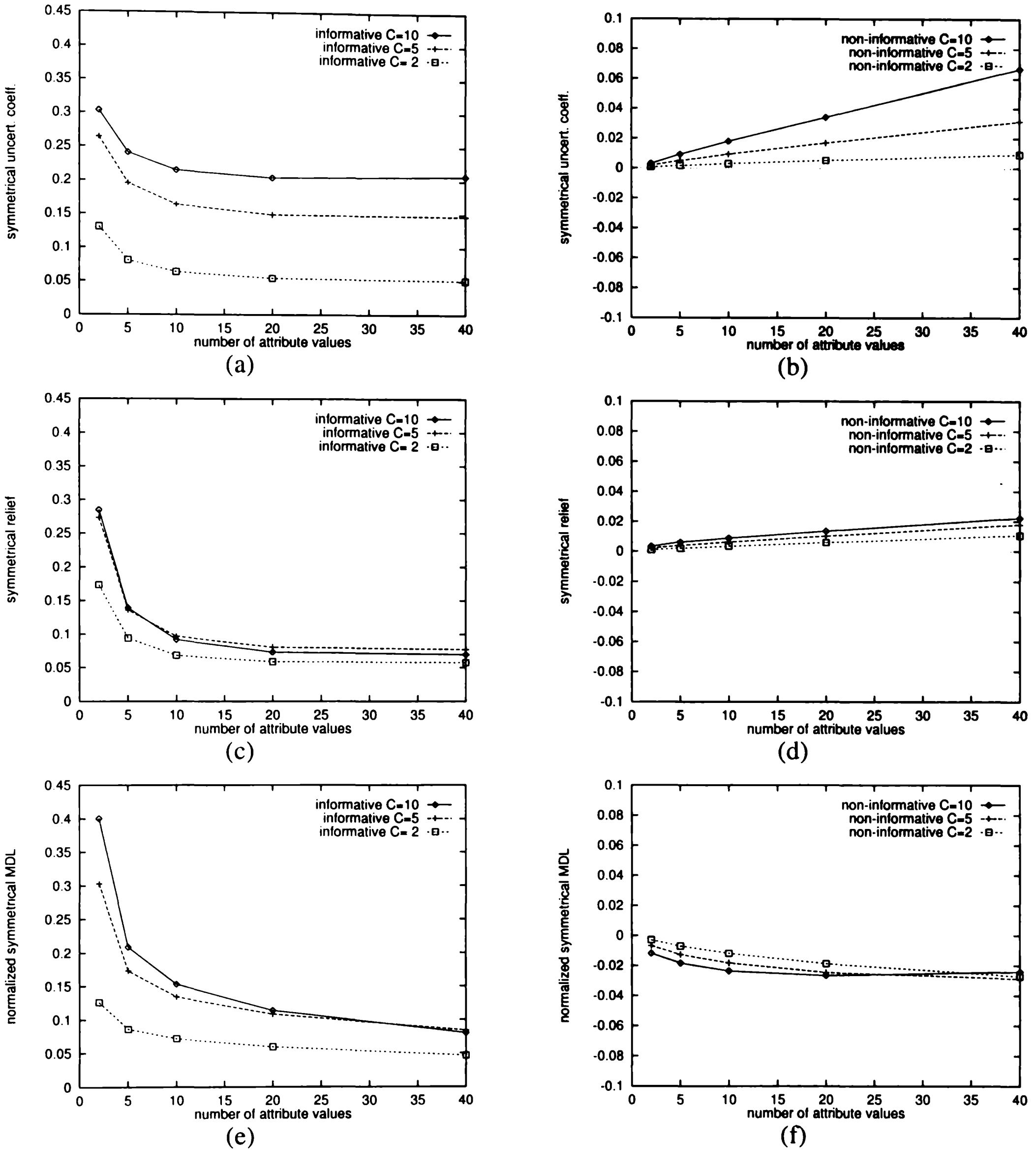
The merit of all features is calculated using symmetrical uncertainty, MDL, and *relief*. The results of each measure are averaged over 1000 trials.

4.3.2 Varying the Level of Attributes

This section explores the effects of varying the number of attribute values on the bias of the measures, using a fixed number of training instances. Figure 4.2 show the results for informative and non-informative attributes when there are 2, 5, and 10 classes.

The estimates of informative attributes by all three measures decrease exponentially with the number of values. The effect is less extreme for symmetrical uncertainty compared with the other two. This behaviour is comparable with Occam's Razor, which states that, all things being equal, the simplest explanation is usually the best. In practical terms, feature selection using these measures will prefer features with fewer values to those with more values; furthermore, since probability estimation is likely to be more reliable for attributes with fewer values (especially if data is limited), there is less risk of overfitting the training data and generalizing poorly to novel cases.

For non-informative attributes, the MDL measure is the best of the three. Its estimates are always less than zero—clearly distinguishing them from the informative attributes. Symmetrical uncertainty and *relief* both exhibit a linear bias in favour of non-informative attributes with more values. *Relief's* estimates are lower (relative to the informative attributes) than symmetrical uncertainty. When the curves corresponding to the number of classes are compared between informative and non-informative attributes for symmetrical uncertainty and *relief*, it can be seen from the scale of the graphs that there is a clear separation between the estimates for informative and non-informative attributes—even when the informative attributes are least informative, that is, when the number of classes is 2 (see Equation 4.15). However, there is a possibility that a non-informative attribute with



(a) (b) (c) (d) (e) (f)

Figure 4.2: The effects of varying the attribute and class level on symmetrical uncertainty (a & b), symmetrical relief (c & d), and normalized symmetrical MDL (e & f) when attributes are informative (graphs on the left) and non-informative (graphs on the right). Curves are shown for 2, 5, and 10 classes.

many values could be estimated as more useful than a slightly informative attribute by symmetrical uncertainty and *relief*. The next section—which examines the behaviour of the measures when the sample size is varied—shows that the danger of this occurring is greater when there are fewer training examples.

4.3.3 Varying the Sample Size

Experiments described in this section vary the number of training examples and examine the effect this has on the behaviour of the correlation measures. Training data sets containing between 50 and $20,000$ instances were generated. The results of each measure were averaged over 1000 trials for each training set size. In the graphs presented below, the number of classes is set to 10 and curves are generated for 2, 10, and 20 attribute values. Curves for 2 classes show similar (but less extreme) tendencies as those shown below and can be found in appendix A.

Figure 4.3 shows the results for the three correlation measures. The behaviour of all three measures is stable for large training set sizes. The estimates of both symmetrical uncertainty and symmetrical *relief* show a tendency to increase exponentially with fewer training examples. The effect is more marked for attributes (both informative and non-informative) with more values.

Since the number of training examples for a given problem is typically fixed, an increase in the value of the measure for a smaller training set does not pose a problem for informative attributes, given that the increase is the same for attributes of differing levels. However, as can be seen from the graphs, the increase is not constant with respect to attribute level and applies to non-informative attributes as well. Symmetrical uncertainty and *relief* show greater increase for both informative and non-informative attributes with greater numbers of values.

In the graph for the symmetrical uncertainty coefficient (Figure 4.3a and Figure 4.3b), the worth of an informative attribute with 20 values becomes greater than that of an informative attribute with 10 values for training sets of less than 500 examples. Both informative attributes with 10 and 20 values "overtake" the informative attribute with 2 values at 100 and 200 training examples respectively. Furthermore, the non-informative attribute with 20 values appears to be more useful than the informative attribute with 2 values for 100 or fewer training examples. *Relief* is slightly better behaved than the symmetrical uncertainty coefficient—while the estimate of an informative attribute with 20 values does overtake that of an informative attribute with 10 values, the estimates of irrelevant attributes do not exceed those of informative attributes.

For informative attributes, the behaviour of the MDL measure (Figure 4.3e and Figure 4.3f) is the exact opposite to that of symmetrical uncertainty and *relief*. Whilst stable for large numbers of training examples, the MDL measure exhibits a tendency to decrease exponentially with fewer training examples. A similar tendency can be observed for non-informative attributes. However, when there are fewer than 1000 training examples, the trend reverses and the measure begins to increase. Again, the effect is more prominent for attributes with greater numbers of values. At less than 200 training examples, non-informative attributes with 10 and 20 values appear slightly informative (> 0). In general, the MDL measure is more pessimistic than the other two—it requires more data in order to ascertain the quality of a feature.

4.3.4 Discussion

The preceding section empirically examined the bias of three attribute quality measures with respect to attribute level and training sample size.

For informative attributes, all three measures exhibit behaviour in the spirit of Occam's Razor by preferring attributes with fewer values when given a choice between equally informative attributes of varying level. When calculating the heuristic merit of feature subsets using Equation 4.2, this bias will result in a preference for subsets containing predictive features with fewer values—a situation that should be conducive to the induction of smaller models by machine learning schemes that prefer simple hypotheses.

With respect to irrelevant attributes, the MDL measure is the best of the three. Except when there are very few training examples, the MDL measure clearly identifies an irrelevant attribute by returning a negative value. Symmetrical uncertainty and *relief* are

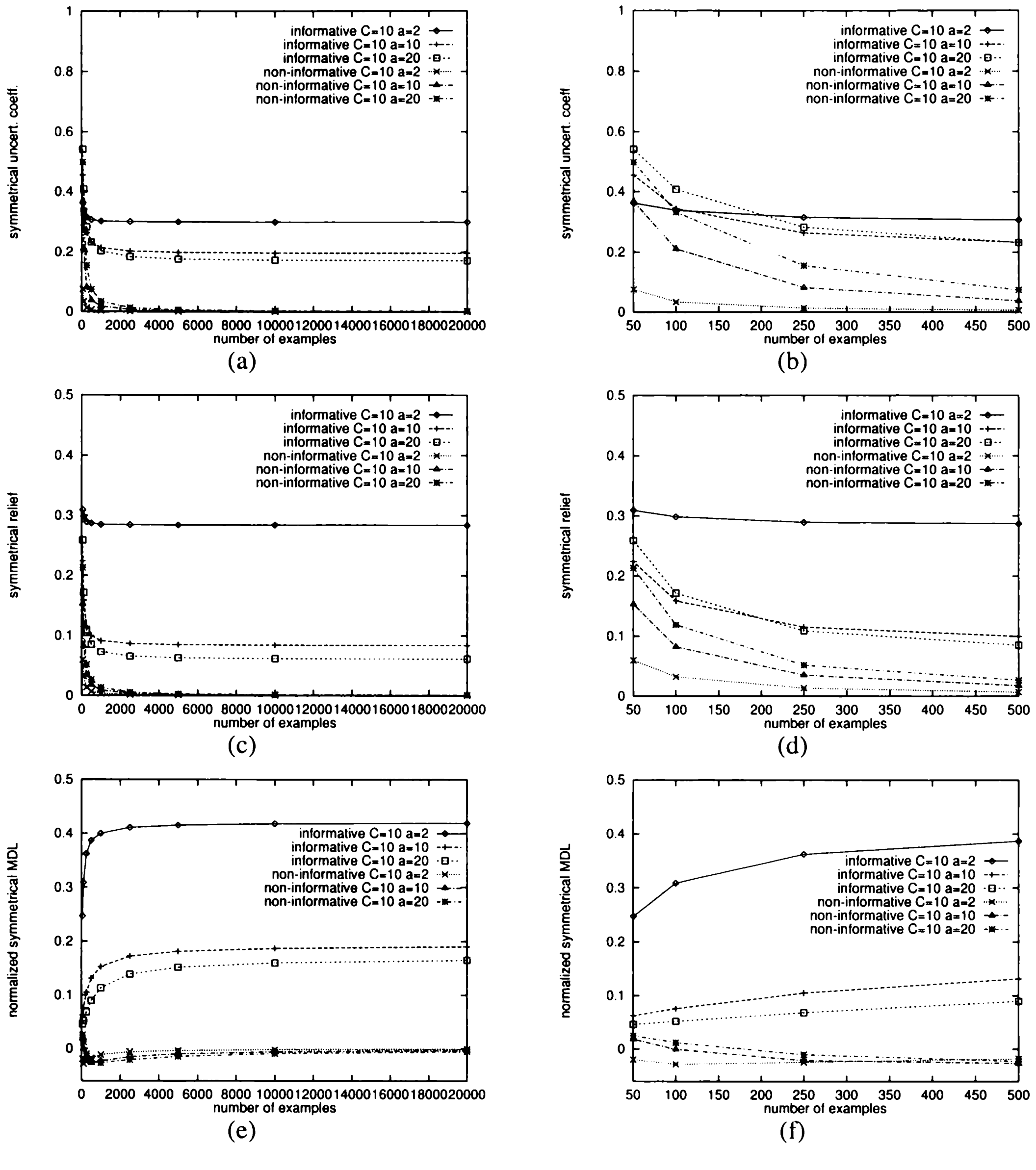


Figure 4.3: The effect of varying the training set size on symmetrical uncertainty (a & b), symmetrical *relief* (c & d), and normalized symmetrical MDL (e & f) when attributes are informative and non-informative. The number of classes is 2; curves are shown for 2, 10, and 20 valued attributes.

linearly biased in favour of irrelevant attributes with greater numbers of values. This is undesirable when measuring an attribute's ability to predict the class because an irrelevant attribute with many values may appear more useful than an informative attribute with few values. The experiments of Section 4.3.3 show that the danger of this occurring is greater for small training set sizes. However, this bias towards multi-valued irrelevant attributes can be advantageous with respect to the feature-feature inter-correlations used in the denominator of Equation 4.2. In Equation 4.2, a feature is more acceptable if it has low correlation with the other features—a multi-valued irrelevant feature will appear more correlated with the others and is less likely to be included in the subset.

Symmetrical uncertainty and *relief* are optimistic measures when there is little training data; the MDL measure is pessimistic. When training sets are small, using the MDL measure in Equation 4.2 may result in a preference for smaller feature subsets containing only very strong correlations with the class.

The next section introduces CFS, a correlation based feature selection algorithm that uses the attribute quality measures described above in its heuristic evaluation function.

4.4 A Correlation-based Feature Selector

CFS is a simple filter algorithm that ranks feature subsets according to a correlation based heuristic evaluation function. The bias of the evaluation function is toward subsets that contain features that are highly correlated with the class and uncorrelated with each other. Irrelevant features should be ignored because they will have low correlation with the class. Redundant features should be screened out as they will be highly correlated with one or more of the remaining features. The acceptance of a feature will depend on the extent to which it predicts classes in areas of the instance space not already predicted by other features. CFS's feature subset evaluation function (Equation 4.2) is repeated here (with slightly modified notation) for ease of reference:

$$M_S = \frac{k\overline{r_{cf}}}{\sqrt{k + k(k-1)\overline{r_{ff}}}} \tag{4.16}$$

where M_S is the heuristic "merit" of a feature subset S containing k features, $\overline{r_{cf}}$ is the mean feature-class correlation ($f \in S$), and $\overline{r_{ff}}$ is the average feature-feature intercorrelation. The numerator of Equation 4.16 can be thought of as providing an indication of how predictive of the class a set of features are; the denominator of how much redundancy there is among the features.

Equation 4.16 forms the core of CFS and imposes a ranking on feature subsets in the search space of all possible feature subsets. This addresses issue 3 (evaluation strategy) in Langley's [Lan94] characterization of search based feature selection algorithms (Chapter 3, Section 2). Since exhaustive enumeration of all possible feature subsets is prohibitive in most cases, issues 1, 2, and 4, concerning the organization of the search, start point, and stopping criterion must be addressed also. The implementation of CFS used in the experiments described in this thesis allows the user to choose from three heuristic search strategies: forward selection, backward elimination, and best first. Forward selection begins with no features and greedily adds one feature at a time until no possible single feature addition results in a higher evaluation. Backward elimination begins with the full feature set and greedily removes one feature at a time as long as the evaluation does not degrade. Best first can start with either no features or all features. In the former, the search progresses forward through the search space adding single features; in the latter the search moves backward through the search space deleting single features. To prevent the best first search from exploring the entire feature subset search space, a stopping criterion is imposed. The search will terminate if five consecutive fully expanded subsets show no improvement over the current best subset.

Figure 4.4 shows the stages of the CFS algorithm and how it is used in conjunction with a machine learning scheme. A copy of the training data is first discretized using the method of Fayyad and Irani [FI93], then passed to CFS. CFS calculates feature-class and feature-feature correlations using one of the measures described in section refsec:cnf and then searches the feature subset space. The subset with the highest merit (as measured by Equation 4.16) found during the search is used to reduce the dimensionality of both the original training data and the testing data. Both reduced datasets may then be passed to a machine learning scheme for training and testing. It is important to note that the general concept of correlation-based feature selection does not depend on any one module (such as discretizaton). A more sophisticated method of measuring correlation may make discretization unnecessary. Similarly, any conceivable search strategy may be used with CFS.

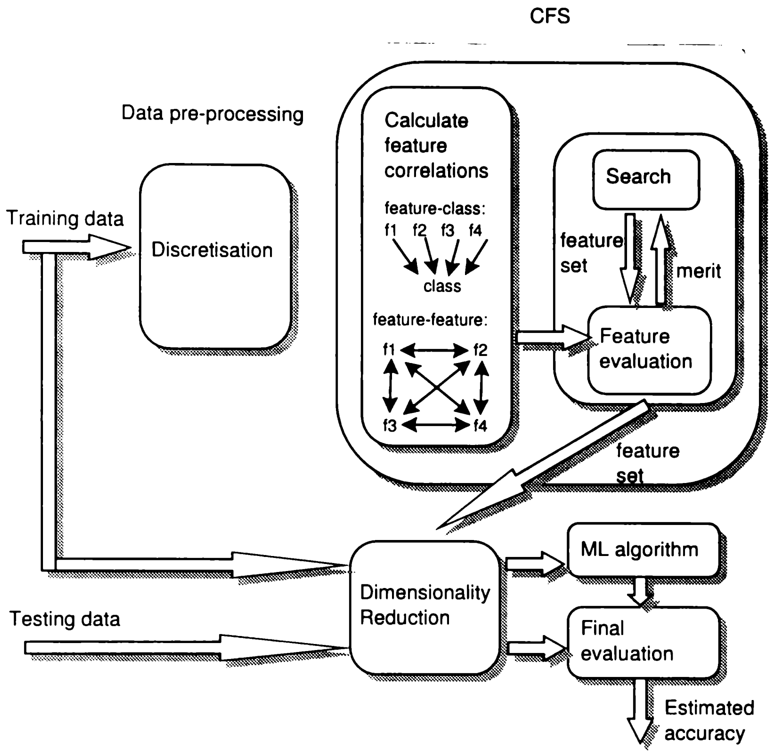


Figure 4.4: The components of CFS. Training and testing data is reduced to contain only the features selected by CFS. The dimensionally reduced data can then be passed to a machine learning algorithm for induction and prediction.

Three variations of CFS—each employing one of the attribute quality measures described in the previous section to estimate correlations in Equation 4.16—are evaluated in the experiments described in Chapter 5. CFS-UC uses symmetrical uncertainty to measure correlations, CFS-MDL uses normalized symmetrical MDL to measure correlations, and CFS-Relief uses symmetrical *relief* to measure correlations. Unknown (missing) data values are treated as a separate value when calculating correlations. The best way to deal with unknowns depends on their meaning in the domain. If the unknown has a special meaning (for example, a blank entry for a particular symptom of a disease may mean that the patient does not exhibit the symptom), treating it as a separate value is the best approach. However, if the unknowns represent truly missing information, then a more sophisticated scheme such as distributing the counts associated with missing entries across the values of an attribute (in proportion to their relative frequencies) may be more appropriate.

Table 4.2 and Table 4.3 give an example of CFS applied to the "Golf" data set (Table 2.1). Table 4.2 shows the feature correlation matrix for the data set—*relief* has been used to calculate the correlations. Table 4.3 illustrates a forward selection search through the feature subset space along with the merit of each subset, calculated using Equation 4.16. The search begins with the empty set of features, which has zero merit. Each single feature addition to the empty set is evaluated; Humidity is added to the subset because it has the highest score. The next step involves trying each of the remaining features with Humidity and choosing the best (Outlook). Similarly, in the next stage Wind is added to the subset. The last step tries the single remaining feature (Temperature) with the current subset; this does not improve its merit and the search terminates. The best subset found (Outlook, Humidity, Wind) is returned.

	Outlook	Temperature	Humidity	Wind	Class
Outlook	1.000	0.116	0.022	0.007	0.130
Temperature		1.000	0.248	0.028	0.025
Humidity			1.000	0.000	0.185
Wind				1.000	0.081

Table 4.2: Feature correlations calculated from the "Golf" dataset. *Relief* is used to calculate correlations.

Computational Expense The time complexity of CFS is quite low. It requires $m((n^2 - n)/2)$ operations for computing the pairwise feature correlation matrix, where m is the number of instances and n is the initial number of features. The feature selection search requires $(n^2 - n)/2$ operations (worst case) for a forward selection or backward elimination search. Best first search in its pure form is exhaustive, but the use of a stopping criterion makes the probability of exploring the entire search space small. In evaluating Equation 4.16 for a feature subset S containing k features, k additions are required in the numerator (feature-class correlations) and $(k^2 - k)/2$ additions are required in the denominator (feature-feature inter-correlations). Since the search algorithm imposes a partial ordering on the search space, the numerator and denominator of Equation 4.16 can be calculated incrementally; this requires only one addition (or subtraction if using a

Feature set	k	$\overline{r_{cf}}$	$\overline{r_{ff}}$	Merit
[]	0	N/A	N/A	0.0
[Outlook]	1	0.130	1.000	$\frac{1\times0.130}{\sqrt{1+1(1-1)1.0}} = 0.130$
[Temperature]	1	0.025	1.000	$\frac{1\times0.025}{\sqrt{1+1(1-1)1.0}} = 0.025$
[**Humidity**]	1	0.185	1.000	$\frac{1\times0.185}{\sqrt{1+1(1-1)1.0}} = 0.185$
[Wind]	1	0.081	1.000	$\frac{1\times0.081}{\sqrt{1+1(1-1)1.0}} = 0.081$
[**Outlook Humidity**]	2	0.158	0.022	$\frac{2\times0.158}{\sqrt{2+2(2-1)0.022}} = 0.220$
[Temperature Humidity]	2	0.105	0.258	$\frac{2\times0.105}{\sqrt{2+2(2-1)0.258}} = 0.133$
[Humidity Wind]	2	0.133	0.0	$\frac{2\times0.133}{\sqrt{2+2(2-1)0.0}} = 0.188$
[Outlook Temperature Humidity]	3	$0.11\bar{3}$	0.132	$\frac{3\times0.11\bar{3}}{\sqrt{3+3(3-1)0.132}} = 0.175$
[**Outlook Humidity Wind**]	3	0.132	0.0096	$\frac{3\times0.132}{\sqrt{3+3(3-1)0.0096}} = 0.226$
[Outlook Temperature Humidity Wind]	4	0.105	0.0718	$\frac{4\times0.105}{\sqrt{4+4(4-1)0.0718}} = 0.191$

Table 4.3: A forward selection search using the correlations in Table 4.2. The search starts with the empty set of features [] which has merit 0.0. Subsets in bold show where a local change to the previous best subset has resulted in improvement with respect to the evaluation function.

backward search) in the numerator and k additions/subtractions in the denominator.

If a forward search is used, it is not necessary to pre-compute the entire feature correlation matrix; feature correlations can be calculated as they are needed during the search. Unfortunately, this can not be applied to a backward search as a backward search begins with all features.

Independence Assumption Like the naive Bayesian classifier, CFS assumes that features are conditionally independent given the class. Experiments with naive Bayes on real data sets has shown that it can perform well even when this assumption is moderately violated [DP96, DP97]; it is expected that CFS can identify relevant features when moderate feature dependencies exist. However, when strong feature interactions occur, CFS may fail to select all the relevant features. An extreme example of this is a parity problem—no feature in isolation will appear any better than any other feature (relevant or not). Chapter 8 explores two additional methods for detecting feature dependencies given the class.

4.5 Chapter Summary

This chapter presents a new feature selection technique for discrete-class supervised learning. The technique, dubbed CFS (Correlation-based Feature Selection) assumes that useful feature subsets contain features that are predictive of the class but uncorrelated with one another. CFS computes a heuristic measure of the "merit" of a feature subset from pair-wise feature correlations and a formula adapted from test theory. Heuristic search is used to traverse the space of feature subsets in reasonable time; the subset with the highest merit found during the search is reported.

CFS treats features uniformly by discretizing all continuous features in the training data at the outset. The supervised discretization method of Fayyad and Irani [FI93] is employed because this method has been found to perform the best when used as a pre-processing step for machine learning algorithms [DKS95, KS96a].

Three measures of association between nominal variables are reviewed for the task of quantifying the feature-class and feature-feature correlations necessary to calculate the merit of a feature subset from Equation 4.16. Symmetrical uncertainty, MDL, and *relief* all prefer predictive features with fewer values over those with more values. The bias of these measures is likely to promote the choice of feature subsets that will give good results with machine learning algorithms (especially those that prefer compact predictive theories) than the bias of measures (such as information gain) that favour multi-valued attributes. All three measures may report irrelevant attributes with many values as being predictive to some degree. The chance of an irrelevant attribute being preferred to a predictive one is greatest when there are few training examples.

Chapter 5

Datasets Used in Experiments

In order to evaluate the effectiveness of CFS for selecting features for machine learning, this thesis takes an empirical approach by applying CFS as a pre-processing step for several common machine learning algorithms.

This chapter reviews the datasets and the general methodology used in the experiments presented in Chapters 6, 7, and 8.

5.1 Domains

Twelve natural domains and six artificial domains were used for evaluating CFS with machine learning algorithms. The natural domains and the three artificial Monk's domains [TBB$^+$91] are drawn from the UCI repository of machine learning databases [MM98]. These domains were chosen because of (a) their predominance in the literature, and (b) the prevalence of nominal features, thus reducing the need to discretize feature values.

In addition, three artificial domains of increasing difficulty were borrowed from Langley and Sage [LS94c], where they were used to test a wrapper feature selector for nearest neighbour classification. Artificial domains are useful because they allow complete control over parameters such as attribute level, predictive ability of attributes, number of irrelevant/redundant attributes, and noise. Varying parameters allows conjectures to be tested and the behaviour of algorithms under extreme conditions to be examined.

Table 5.1 summarises the characteristics of these domains. The default accuracy is the accuracy of always predicting the majority class on the whole dataset. For the three artificial domains (A1, A2, and A3), the default accuracy is the default accuracy in the limit, that is, given that each example is equally likely and an infinite sample is available. Variations of the three artificial domains with added irrelevant and redundant features are used to test CFS's ability to screen out these types of features.

Domain	Instances	Features	% Missing	Average # Feature Vals	Max/Min # Feature Vals	Class Vals	Default Accuracy
mu	8124	22	1.3	5.3	12/1	2	51.8
vo	435	16	5.3	2.0	2/2	2	61.4
v1	435	15	5.3	2.0	2/2	2	61.4
cr	690	15	0.6	4.4	14/2	2	55.5
ly	148	18	0.0	2.9	8/2	4	54.7
pt	339	17	3.7	2.2	3/2	23	24.8
bc	286	9	0.3	4.6	11/2	2	70.3
dna	106	55	0.0	4.0	4/4	2	50.0
au	226	69	2.0	2.2	6/2	24	25.2
sb	683	35	9.5	2.8	7/2	19	13.5
hc	368	27	18.7	23.8	346/2	2	62.7
kr	3196	36	0.0	2.0	3/2	2	52.2
A1	1000	3	0.0	2.0	2/2	2	87.5
A2	1000	3	0.0	2.0	2/2	2	50.0
A3	1000	3	0.0	2.0	2/2	2	75.0
M1	432	6	0.0	2.8	4/2	2	50.0
M2	432	6	0.0	2.8	4/2	2	67.1
M3	432	6	0.0	2.8	4/2	2	52.8

Table 5.1: Domain characteristics. Datasets above the horizontal line are natural domains; those below are artificial. The % Missing column shows what percentage of the data set's entries (number of features × number of instances) have missing values. Average # Feature Vals and Max/Min # Feature Vals are calculated from the nominal features present in the data sets.

The following is a brief description of the datasets.

Mushroom (mu) This dataset contains records drawn from The Audubon Society Field Guide to North American Mushrooms [Lin81]. The task is to distinguish edible from poisonous mushrooms on the basis of 22 nominal attributes describing characteristics of the mushrooms such as the shape of the cap, odour, and gill spacing. This is a large dataset containing 8124 instances. C4.5 and IB1 can achieve over 99% accuracy on this dataset, but naive Bayes does not do as well, suggesting that many of the attributes may be redundant.

Vote (vo, v1) In this dataset the party affiliation of U.S House of Representatives Congressmen is characterised by how they voted on 16 key issues such as education spending and immigration. There are 435 (267 democrats, 168 republicans) instances and all features are binary. In the original data, there were nine different types of vote a congressman could make. Some of these have been collapsed into related voting categories. The v1 version of this dataset has the single most predictive attribute (physician-fee-freeze) removed.

Australian credit screening (cr) This dataset contains 690 instances from an Australian credit company. The task is to distinguish credit-worthy from non credit-worthy customers. There are 15 attributes whose names and values have been converted to meaningless symbols to ensure confidentiality of the data. There are six continuous features and nine nominal. The nominal features range from 2 to 14 values.

Lymphography (ly) This is a small medical dataset containing 148 instances. The task is to distinguish healthy patients from those with metastases or malignant lymphoma. All 18 features are nominal. This is the one of three medical domains (the others being Primary Tumour and Breast Cancer) provided by the University Medical Centre, Institute of Oncology, Ljubljana, Yugoslavia.

Primary Tumour (pt) This dataset involves predicting the location of a tumour in the body of a patient on the basis of 17 nominal features. There are 22 classes corresponding to body locations such as lung, pancreas, liver, and so forth. 365 instances are provided.

Breast Cancer (bc) The task is to predict whether cancer will recur in patients. There are 9 nominal attributes describing characteristics such as tumour size and location. There are 286 instances.

Dna-promoter (dna) A small dataset containing 53 positive examples of E. coli promoter gene sequences and 53 negative examples. There are 55 nominal attributes representing the gene sequence. Each attribute is a DNA nucleotide ("base-pair") having four possible values (A, G, T, C).

Audiology (au) The task is to diagnose ear dysfunctions. There are 226 instances described by 69 nominal features. There are 24 classes. This dataset is provided by Professor Jergen at the Baylor College of Medicine.

Soybean-large (sb) The task is to diagnose diseases in soybean plants. There are 683 examples described by 35 nominal features. Features measure properties of leaves and various plant abnormalities. There are 19 classes (diseases).

Horse colic (hc) There are 368 instances in this dataset, provided by Mary McLeish and Matt Cecile from the University of Guelph. There are 27 attributes, of which 7 are continuous. Features include whether a horse is young or old, whether it had surgery, pulse, respiratory rate, level of abdominal distension, etc. There are a number of attributes that could serve as the class—the most commonly used is whether a lesion is surgical.

Chess end-game (kr) This dataset contains 3196 chess end-game board descriptions. Each end game is a King + Rook versus King + Pawn on a7 (one square away from queening) and it is the King + Rook's side (white) to move. The task is to predict if white can win on the basis of 36 features that describe the board. There is one feature with three values; the others are binary.

A1, A2, A3 These three boolean domains are borrowed from Langley and Sage [LS94c]. They exhibit an increasing level of feature interaction. Irrelevant and redundant attributes are added to these domains to test CFS's ability to deal with these sorts of features.

A1 is a simple conjunction of three features and exhibits the least amount of feature interaction. The concept is:

$$A \wedge B \wedge C$$

The class is 1 when A, B, and C all have the value 1, otherwise the class is 0.

A2 is a disjunct of conjuncts (sometimes known as an m-of-n concept). In this case it is a 2-of-3 concept—that is, the class is 1 if 2 or more of bits A, B, and C are set to 1:

$$(A \wedge B) \vee (A \wedge C) \vee (B \wedge C)$$

This problem is more difficult than A1 due to the higher degree of interaction among the

3 features.

A3 exhibits the highest degree of feature interaction, similar to a parity problem in that no single feature in isolation will appear useful. The concept is:

$$(A \wedge B \wedge C) \vee (\bar{A} \wedge \bar{B} \wedge \bar{C})$$

This last domain is included as an example of a situation when CFS will fail to select the relevant features due to the fact that its assumption of attribute independence given the class is completely incorrect.

Monk's problems The Monk's problems are three artificial domains, each using the same representation, that have been used to compare machine learning algorithms [TBB$^+$91]. Monk's domains contain instances of robots described by six nominal features:

$$\begin{aligned}
\text{Head-shape} &\in \{\text{round, square, octagon}\}\\
\text{Body-shape} &\in \{\text{round, square, octagon}\}\\
\text{Is-smiling} &\in \{\text{yes, no}\}\\
\text{Holding} &\in \{\text{sword, balloon, flag}\}\\
\text{Jacket-colour} &\in \{\text{red, yellow, green, blue}\}\\
\text{Has-tie} &\in \{\text{yes, no}\}
\end{aligned}$$

There are three Monk's problems, each with 432 instances in total. For each problem there is a standard training and test set.

Monk1 (M1) The concept is:

$$(\text{head-shape} = \text{body-shape}) \text{ or } (\text{jacket-colour} = \text{red})$$

This problem is difficult due to the interaction between the first two features. Note that only one value of the jacket-colour feature is useful.

Monk2 (M2) The concept is:

Exactly two of the features have their first value.

This is a hard problem due to the pairwise feature interactions and the fact that only one value of each feature is useful. Note that all six features are relevant to the concept. C4.5 does no better than predicting the default class on this problem.

Monk3 (M3) The concept is:

$$\begin{gathered}(\text{jacket-colour} = \text{green and holding} = \text{sword}) \text{ or} \\ (\text{jacket-colour} \neq \text{blue and body-shape} \neq \text{octagon})\end{gathered}$$

The standard training set for this problem has 5% class noise added—that is, 5% of the training examples have had their label reversed. This is the only Monk's problem that is not noise free. It is possible to achieve approximately 97% accuracy using only the (jacket-colour $\neq$ blue and body-shape $\neq$ octagon) disjunct.

5.2 Experimental Methodology

The experiments described in this thesis compare runs of machine learning algorithms with and without feature selection on the datasets described in the previous section. Accuracy of algorithms is measured using random subsampling, which performs multiple random splits of a given dataset into disjoint train and test sets. In each trial, an algorithm is trained on the training dataset and the induced theory is evaluated on the test set. When algorithms are compared, each is applied to the same training and test sets. The testing accuracy of an algorithm is the percentage of test examples it classifies correctly. Table 5.2 shows the train and test set sizes used with the natural domains and the Monk's problems. On the natural domains, a two-thirds training and one-third testing split was used in all but four cases. A fifty/fifty split was used on the vote datasets, a one-third/two-thirds split on credit and one-eighth of the instances were used for training on mushroom (the largest dataset). In the case of the Monk's problems, testing is performed on the full dataset (as was done originally by Thrun et al. [TBB+91]). Various different train and test set sizes are used with the artificial domains A1–3 (see Chapter 6 for details).

Domain	Train size	Test size
mu	1000	7124
vo	218	217
v1	218	217
cr	228	462
ly	98	50
pt	226	113
bc	191	95
dna	69	37
au	149	77
sb	450	223
hc	242	126
kr	2110	1086
M1	124	432
M2	169	432
M3	122	432

Table 5.2: Training and test set sizes of the natural domains and the Monk's problems.

With the exception of learning curve experiments (described below), accuracy is averaged over 50 train-test trials on a given dataset. Furthermore, each train and test set split is *stratified*. Stratification ensures the class distribution from the whole dataset is preserved in the training and test sets. Stratification has been shown to help reduce the variance of the estimated accuracy—especially for datasets with many classes [Koh95b].

Since CFS requires datasets to be discretized, the discretization method of Fayyad and Irani [FI93] is applied to a copy of each training dataset before it is passed to CFS. Because only the effects of feature selection are of interest, all induction is performed using the original, undiscretized training datasets. For a test using CFS with dataset X, for example, the dataset is discretized, features are selected, then the machine learning algorithm is run, using the selected features in their original, nondiscrete form.

Two-tailed paired t-tests are used to determine whether the difference between two algorithms is significant or not. Difference in accuracy is considered significant when the p-value is less than 0.05 (confidence level is greater than 95%). When two or more algorithms are compared, a table of accuracies is given summarising the results. The symbols "+" (or "−") are used to denote that one algorithm is (statistically) significantly better (or worse) than the other. In discussion of results, if one algorithm is stated to be better or worse than another then it is significantly better or worse at the 0.05 level. Often, a bar graph showing the absolute accuracy difference between two algorithms is given. For example, Figure 5.1 shows the absolute difference in accuracy between naive Bayes using CFS for feature selection and naive Bayes without feature selection. A bar above the zero line indicates that CFS has managed to improve naive Baye's performance; a bar below the zero line indicates that CFS has degraded naive Bayes performance. Stars on the bar graph show where differences are statisically different. For C4.5, tables and graphs summarising induced tree sizes are reported.

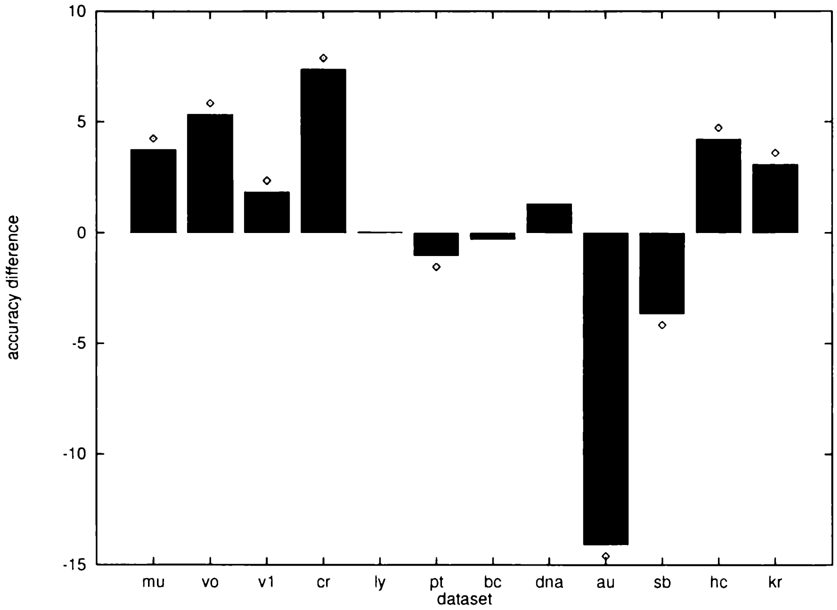


Figure 5.1: Effect of CFS feature selection on accuracy of naive Bayes classification. Dots show results that are statistically significant

The experiments described in Section 1 of the next chapter examine CFS's ability to deal with irrelevant and redundant attributes. It is interesting to examine learning curves for algorithms (such as naive Bayes and IB1) that are adversely affected by these kinds of attributes. A learning curve shows how quickly an algorithm's performance improves as it is given access to more training examples. Random subsampling, as described above, is used to generate training datasets of a given size (20 instances are added at every successive iteration). Testing is performed on the remaining instances. Accuracy for each training set size is averaged over ten trials. For example, Figure 5.2 shows a learning curve for IB1 on the A2 artificial domain with 17 added irrelevant attributes. The error bars represent a 90% confidence interval.

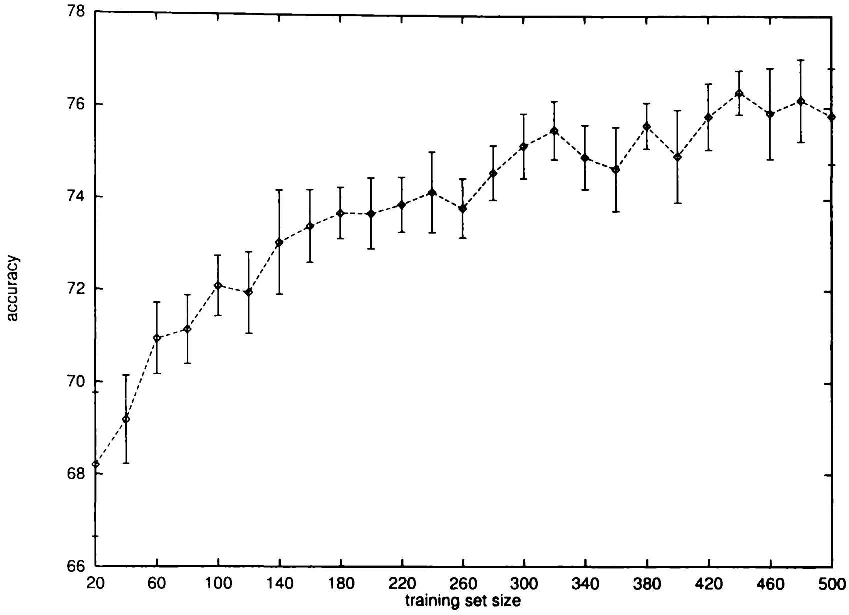


Figure 5.2: The learning curve for IB1 on the dataset A2 with 17 added irrelevant attributes.

Three variations of CFS, using the attribute correlation measures from the previous chapter, are evaluated in the experiments:

- CFS-UC uses symmetrical uncertainty to measure attribute correlations.
- CFS-MDL uses normalized symmetrical MDL to measure attribute correlations.
- CFS-Relief uses symmetrical *relief* to measure attribute correlations.

A forward best first search is used with all three variations of CFS; initial experiments showed this search strategy performed slightly better than forward selection, and the same as backward elimination. The forward best first search evaluated fewer subsets than backward elimination. The best first search stops when 5 consecutive fully expanded nodes showing no improvement (according to the heuristic merit function) have been evaluated. This stopping criterion is the default used in the $\mathcal{MLC}$++ [KJL$^+$94] implementation of the wrapper feature selector. Chapter 7 compares CFS with the wrapper.

When execution times for algorithms are mentioned, they are reported in CPU units on a Sun Sparc server 1000.

Chapter 6

Evaluating CFS with 3 ML Algorithms

This chapter describes an evaluation of CFS using artificial and natural machine learning datasets. With artificial domains, the relevant features are known in advance and CFS performance can be directly ascertained. With natural domains, the relevant features are often not known in advance, so the performance of CFS must be measured indirectly. One way to do this is to compare a learning algorithm's performance with and without feature selection by CFS.

Section 6.1 examines CFS's ability to deal with irrelevant and redundant features in artificial domains. In Section 6.2, CFS is used to select features on natural domains and the performance of machine learning algorithms, before and after feature selection, is compared.

6.1 Artificial Domains

The purpose of the experiments described in this section is to empirically test the claim that CFS's feature evaluation heuristic (Equation 4.16) can filter irrelevant and redundant features.

Boolean domains Sections 6.1.1 and 6.1.2 discuss the performance when irrelevant and redundant features are added to three artificial boolean domains borrowed from Langley and Sage [LS94c]. The three concepts are:

A1 $$A \wedge B \wedge C$$

A2 $(A \wedge B) \vee (A \wedge C) \vee (B \wedge C)$

A3 $(A \wedge B \wedge C) \vee (\bar{A} \wedge \bar{B} \wedge \bar{C})$

The concepts exhibit an increasing level of feature interaction and therefore provide the opportunity to test CFS's behaviour when its assumption of feature independence given the class is violated.

For each domain, a dataset is randomly generated containing 1000 examples with irrelevant attributes added, and a second dataset is randomly generated containing 1000 examples with redundant attributes added. In the experiments presented below, the number of relevant and irrelevant/redundant attributes selected by CFS is plotted as a function of the number of training examples shown to CFS. This allows the behaviour of the different attribute correlation measures used in the 3 variations of CFS (CFS-UC, CFS-MDL, and CFS-Relief) to be compared. Training sets increasing in size by 20 examples per iteration are selected by the random subsampling method described in the previous chapter. The number of relevant and irrelevant/redundant attributes is averaged over 10 trials for each training set size. Because IB1 is sensitive to irrelevant attributes, learning curves illustrating the impact of feature selection on IB1's accuracy are shown for the datasets with added irrelevant attributes. Similarly, as naive Bayes can be affected by redundant attributes, learning curves are shown for naive Bayes on the datasets with added redundant attributes.

Monk's problems The Monk's problems [TBB$^+$91] are challenging artificial domains that have been used to compare the performance of machine learning algorithms. These domains involve irrelevant features, noise, and high degrees of feature interaction. Section 6.1.3 tests CFS on these concepts.

6.1.1 Irrelevant Attributes

These versions of the boolean domains A1, A2, and A3 each have fifteen uniformly random boolean attributes, one uniformly random 5-valued attribute, and one uniformly random 20-valued attribute added for a total of twenty attributes, seventeen of which are irrelevant.

Concept A1 Figure 6.1 shows the number of irrelevant attributes selected by the three variations of CFS on concept A1. The results show that the average number of irrelevant attributes included by all three versions of CFS decreases rapidly as more training examples are seen. CFS-MDL decreases faster than either CFS-UC or CFS-Relief. This agrees with the results of Chapter 4 which showed the MDL measure to be the most effective at identifying irrelevant attributes.

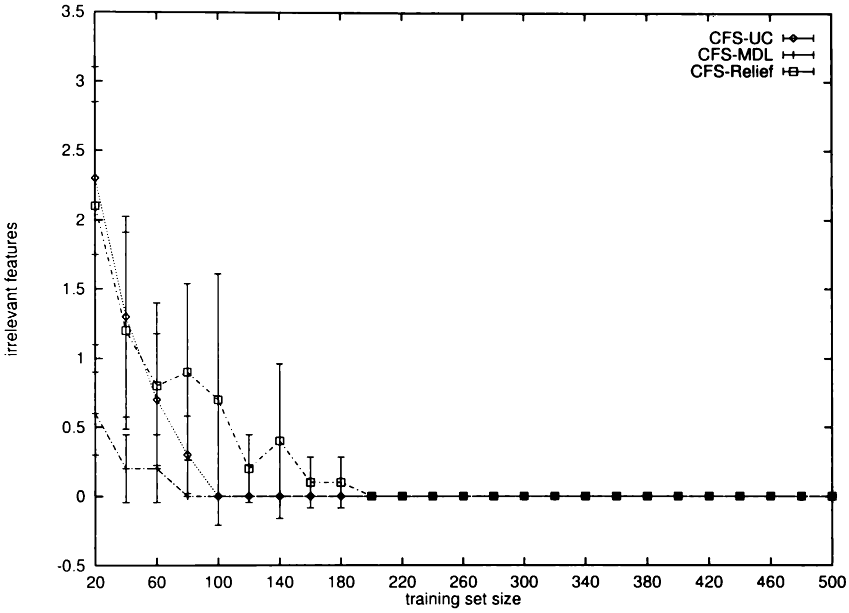


Figure 6.1: Number of irrelevant attributes selected on concept A1 (with added irrelevant features) by CFS-UC, CFS-MDL, and CFS-Relief as a function of training set size.

CFS-Relief selects fewer irrelevant attributes than CFS-UC at training sets of 20 examples, but requires more training examples than either of the other two methods before the number of irrelevant attributes drops to zero. However, given 60 training examples, all three variations of CFS select, on average, less than 1 irrelevant attribute of the 17.

Figure 6.2 shows the number of relevant attributes selected by the three variations of CFS on concept A1. The best results are exhibited by CFS-UC, which always selects all 3 relevant features once 40 training examples have been seen. The average number of relevant features selected by CFS-MDL starts low, but increases rapidly as more training examples are seen. This is consistent with the results of Chapter 4, which show that the

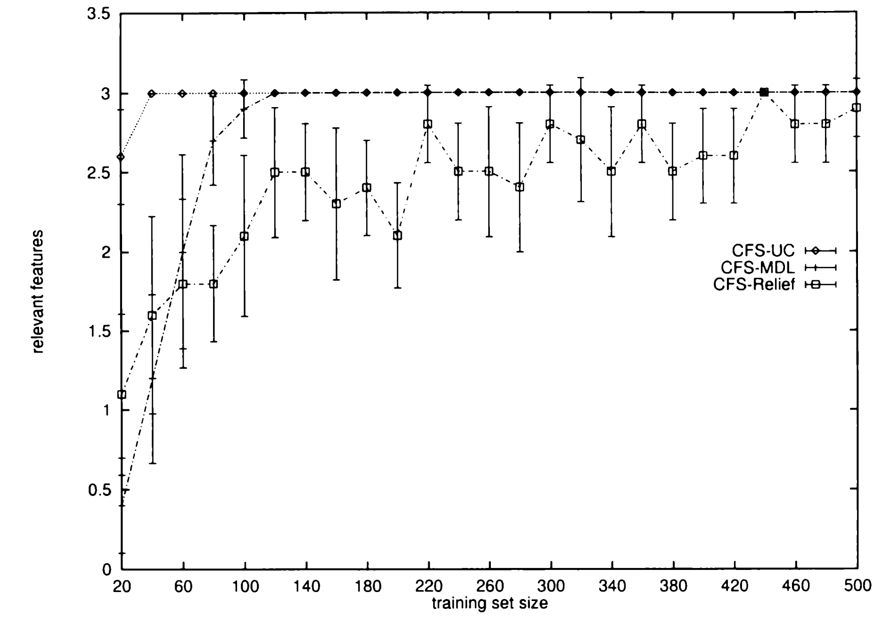


Figure 6.2: Number of relevant attributes selected on concept A1 (with added irrelevant features) by CFS-UC, CFS-MDL, and CFS-Relief as a function of training set size.

MDL measure is pessimistic (compared to either gain ratio or *relief*) in its estimates of relevant attributes when there are few training examples.

The results for CFS-Relief are worse than those for the other two methods. The average number of relevant features selected by CFS-Relief increases as more training instances are seen, but not as rapidly as the other two methods, and there is more variation (the curve is less stable). CFS-Relief does not reliably include all three relevant features by the time 500 training examples have been seen[1]. This is a curious result that bears further investigation.

Understanding the poor performance of CFS-Relief, relative to the other methods, begins with consideration of the feature-class correlations assigned by the three methods. Table 6.1 shows the feature-class correlations assigned to the three relevant features by symmetrical uncertainty, MDL, and *relief* using the full 1000 instances in the dataset for concept A1. The values assigned to the three features by symmetrical uncertainty and MDL are very close (the estimate for C is slightly lower than A or B), which is what one would expect. In fact, if examples of the concept are drawn from a uniformly random

[1] The experiment was continued up to 900 training examples. After 800 training examples, CFS-Relief was selecting all three relevant features consistently.

distribution, in the limit (as the sample size approaches infinity) the correlation values for all three features will be equal. However, *relief* estimates attribute C much lower (relative to A and B) than symmetrical uncertainty or MDL does.

Attribute	symmetrical uncertainty	MDL	*relief*
A	0.1855	0.1906	0.1074
B	0.1866	0.1918	0.1104
C	0.1710	0.1749	0.0682

Table 6.1: Feature-class correlation assigned to features A, B, and C by symmetrical uncertainty, MDL, and *relief* on concept A1.

Since A1 is a conjunctive concept, splitting the instances on the basis of any of the three relevant attributes will produce a pure subset (all instances are of class 0) corresponding to the value 0 of the attribute. Using 1R [Hol93] to produce rules for the three relevant attributes on the full set of 1000 instances gives:

Rule for A:

class(0) :- $A(1)$. (covers 364 out of 490 examples)
class(0) :- $A(0)$. (covers 510 out of 510 examples)

Rule for B:

class(0) :- $B(1)$. (covers 362 out of 488 examples)
class(0) :- $B(0)$. (covers 512 out of 512 examples)

Rule for C:

class(0) :- $C(1)$. (covers 391 out of 517 examples)
class(0) :- $C(0)$. (covers 483 out of 483 examples)

The proportion of class 0 examples covered when an attribute has the value 1 is about 74% for all three attributes, which indicates that they are being differentiated on the basis of the size of their respective pure nodes. When the value is 0, B covers 512 examples, A covers 510 examples, and C covers 483 examples—which is exactly the ranking assigned by the 3 measures in Table 6.1. *Relief* is a modification of the gini impurity measure[2][Bre96b]. Breiman notes that the gini criterion prefers splits that put the largest class into one pure node. *Relief* appears to be more sensitive to the size of the pure node than either symmetrical uncertainty or MDL.

Figure 6.3 shows the learning curves for IB1 with and without feature selection. The

[2] See Chapter 4 Section 4.2.2.

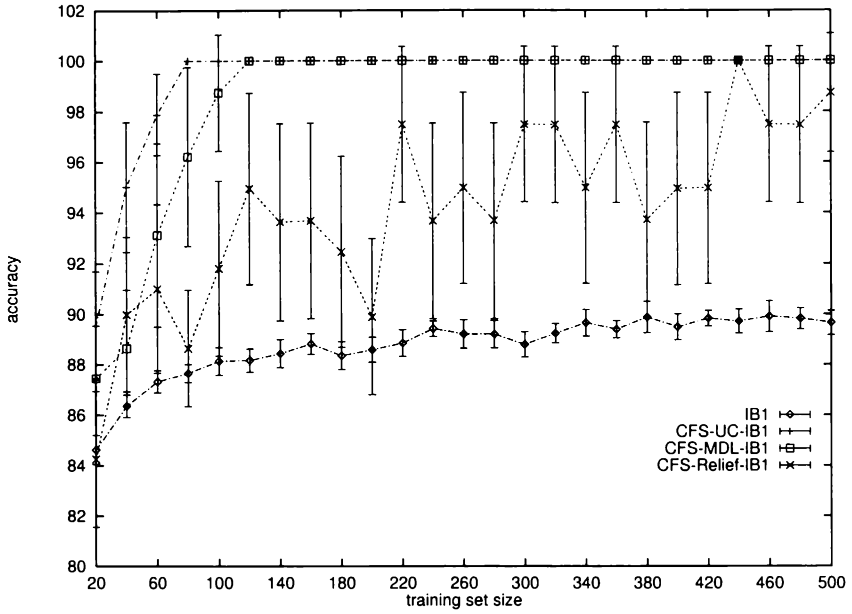


Figure 6.3: Learning curves for IB1, CFS-UC-IB1, CFS-MDL-IB1, and CFS-Relief-IB1 on concept A1 (with added irrelevant features)

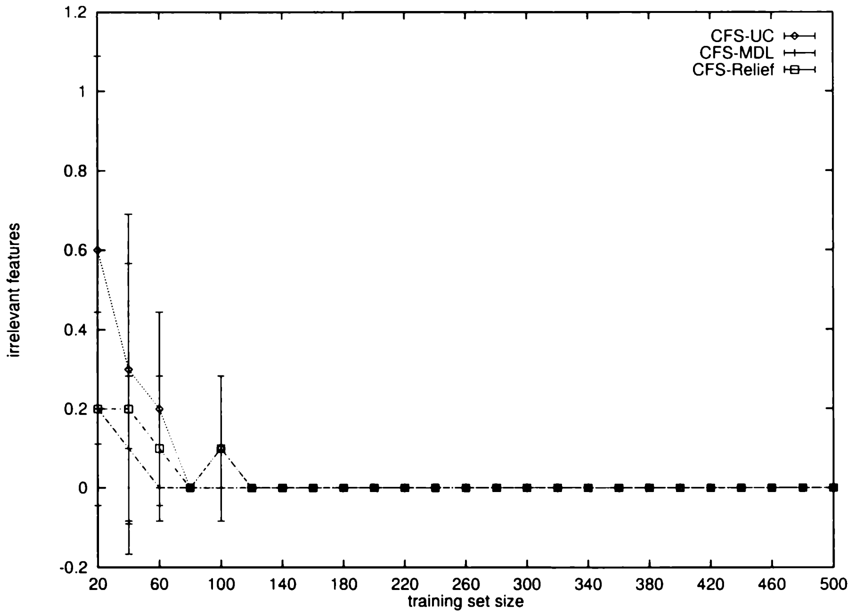


Figure 6.4: Number of irrelevant attributes selected on concept A2 (with added irrelevant features) by CFS-UC, CFS-MDL, and CFS-Relief as a function of training set size. Note: CFS-UC and CFS-Relief produce the same result.

accuracy of IB1 without feature selection increases slowly as the number of training examples increases due to the high number of irrelevant features. IB1's accuracy starts off at 84.6% (below the default accuracy for the concept) and finishes (after 500 training examples) at 89.6 In contrast, the accuracy of IB1 after feature selection by CFS-UC and CFS-MDL increases rapidly—reaching 100% after only 80 and 100 training examples respectively. The accuracy of IB1 after feature selection by CFS-Relief is still better than that of IB1 without feature selection, but does not reach 100% due to the fewer relevant features included by this method. The shape of the learning curves for IB1 with feature selection correspond very closely to the shape of the curves for the number of relevant features shown in Figure 6.2.

Concept A2 Figure 6.4 shows the number of irrelevant attributes selected by the three variations of CFS on concept A2. The results show that all three variations of CFS filter irrelevant features very quickly on this domain—fewer irrelevant features are included compared to results for concept A1. Concept A2 has a uniform distribution of classes—unlike concept A1, which has a large majority of class 0. Therefore, the differentiation between informative and non-informative attributes for concept A1 is done primarily on the basis of the small percentage of instances that have class 1. Relevant attributes will always have value 1 when the class is 1; when there are few training examples, an irrelevant attribute may match a relevant attribute for the small percentage of class 1 cases by chance, making it seem just as informative. For concept A2, a relevant attribute will have value 1 when the class is 1 for 75% of the cases; similarly, a relevant attribute will have value 0 when the class is 0 for 75% of the cases. In this case, it is less likely that an irrelevant attribute's distribution of values given the class will match that of a relevant attribute's by chance. As with concept A1, CFS-MDL screens irrelevant features faster than the other two methods.

Figure 6.5 shows the number of relevant attributes selected by the three variations of CFS on concept A2. All three variations rapidly identify the three relevant attributes. CFS-UC and CFS-Relief consistently choose all three relevant attributes once 260 training examples have been seen; CFS-MDL does the same after 320 training examples. Figure 6.6 shows the learning curves for IB1 with and without feature selection on concept A2. As with concept A1, IB1's accuracy without feature selection improves slowly as more training examples are seen—from 68% to 76% after 500 examples. Feature selection allows IB1 to learn more rapidly and achieve 100% accuracy on this concept after only a small number of training examples. As expected, the shape of the learning curves for IB1 after feature selection correspond closely to the shape of the curves for the number of relevant features selected by the three variations of CFS.

Concept A3 This concept has the highest degree of feature interaction of the three, and it is therefore expected that CFS will be unable to distinguish the relevant features from the irrelevant features. Figure 6.7 and Figure 6.8 show the number of irrelevant and relevant features selected by the three variations of CFS, respectively, on this concept. The graphs show that CFS is indeed unable to distinguish relevant from irrelevant features. CFS-UC and CFS-Relief select more features (especially irrelevant features) than CFS-MDL and exhibit a tendency to include more features as more training examples are seen. Conversely, CFS-MDL appears to favour fewer features as more training examples are seen.

For this concept CFS's assumption that features are independent, given the class, means that—assuming an ideal correlation measure and an infinitely large sample—all feature-class correlations should be zero, and, by Equation 4.16, the merit of all feature subsets should also be zero. However, when training examples are limited, features may appear to be slightly correlated with the class by chance. When there are very few training instances, a feature may stand out as somewhat better than the others. As the number of training examples increases, features will become more homogeneous—their correlations with the class will be more similar to each other (though never becoming exactly zero). In Chapter 4 it was shown that a feature will be accepted into a subset (that is, increase the merit of the subset) if its correlation with the class and average inter-correlation with the features in the subset is the same as the features already in the subset. This explains why there is a small increasing tendency for features to be included by CFS-UC and CFS-Relief. CFS-MDL on the other hand, has a small penalty associated with the size of the model for a feature—decreasing its correlation with the class and often resulting in a value less than zero. This is shown graphically by Figure 6.9, which plots the average number

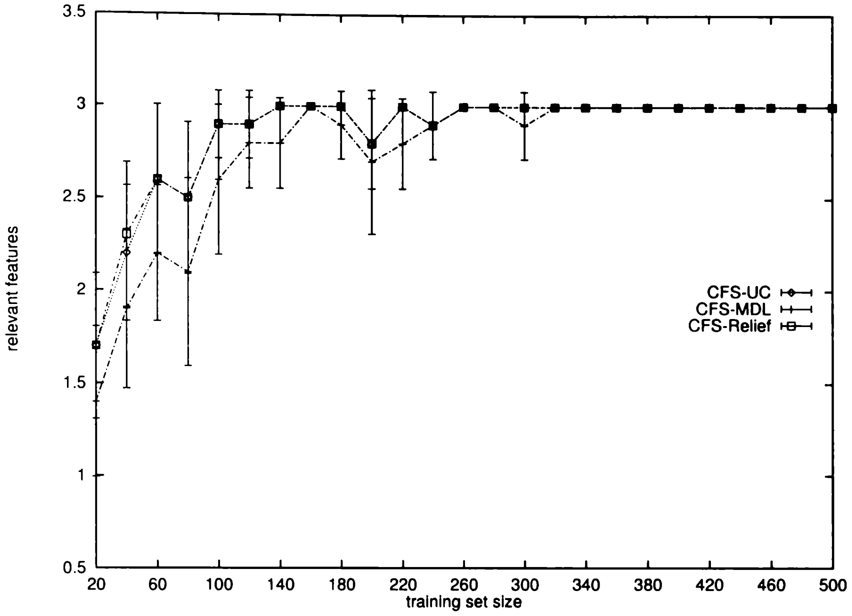


Figure 6.5: Number of relevant attributes selected on concept A2 (with added irrelevant features) by CFS-UC, CFS-MDL, and CFS-Relief as a function of training set size.

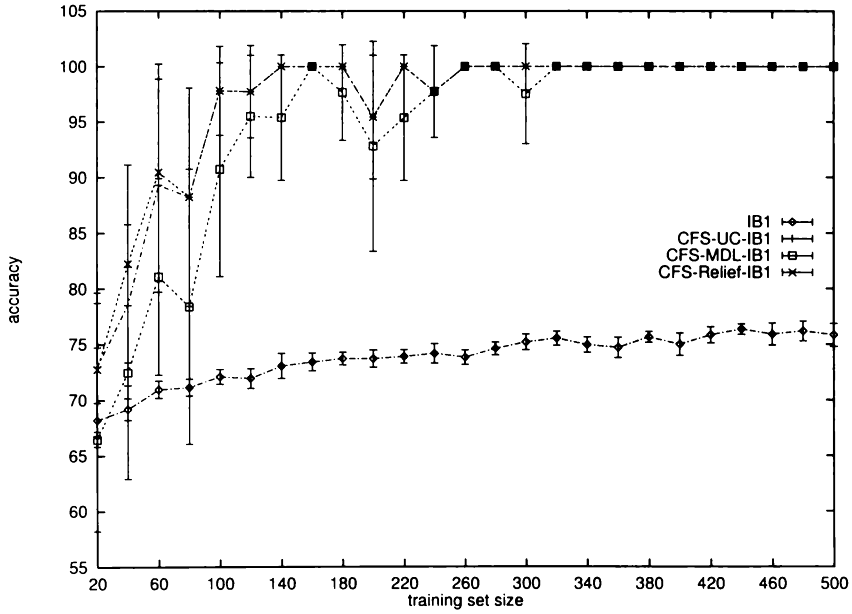


Figure 6.6: Learning curves for IB1, CFS-UC-IB1, CFS-MDL-IB1, and CFS-Relief-IB1 on concept A2 (with added irrelevant features).

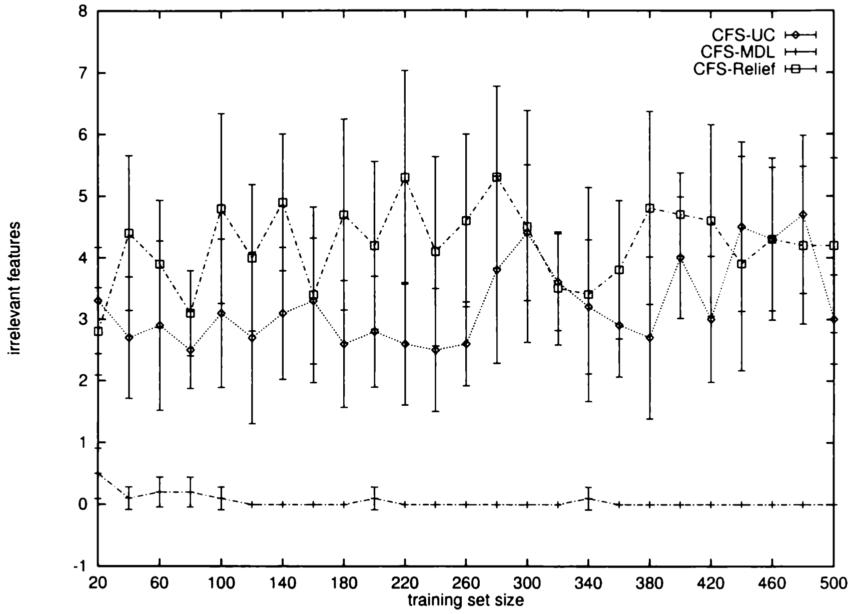


Figure 6.7: Number of irrelevant attributes selected on concept A3 (with added irrelevant features) by CFS-UC, CFS-MDL, and CFS-Relief as a function of training set size.

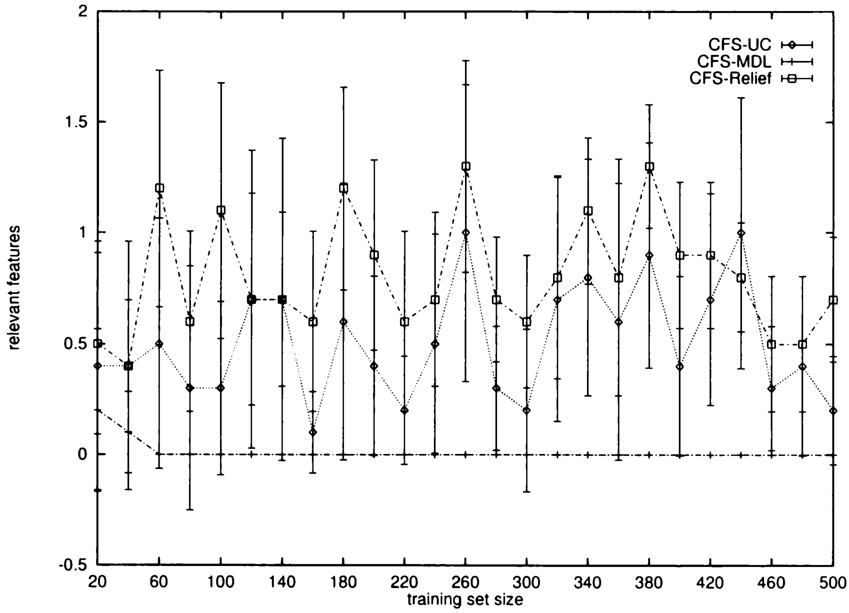


Figure 6.8: Number of relevant attributes selected on concept A3 (with added irrelevant features) by CFS-UC, CFS-MDL, and CFS-Relief as a function of training set size.

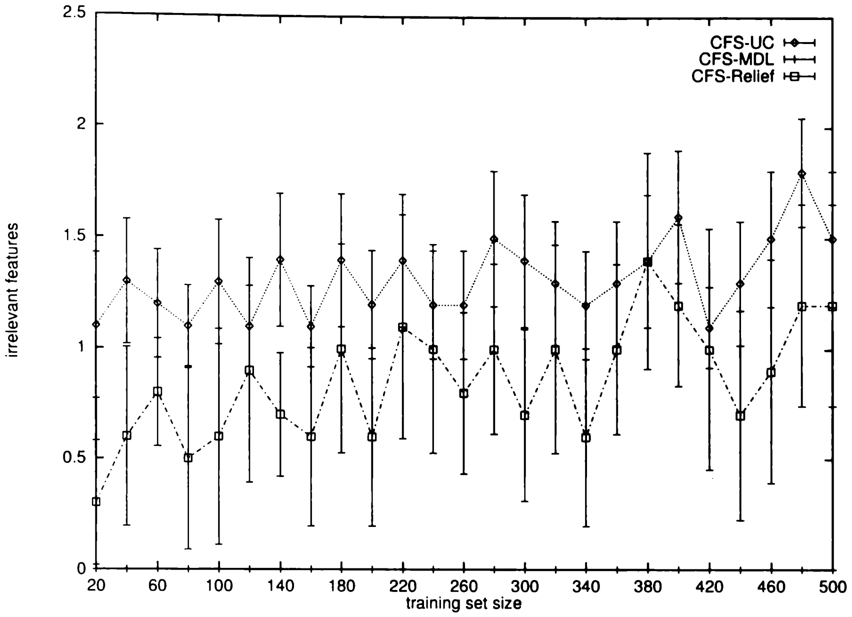


Figure 6.9: Number of irrelevant multi-valued attributes selected on concept A3 (with added irrelevant features) by CFS-UC, CFS-MDL, and CFS-Relief as a function of training set size.

of multi-valued irrelevant attributes (there is one 5-valued and one 20-valued irrelevant attribute) selected by the three variations of CFS. CFS-MDL never selects any multi-valued irrelevant attributes. The MDL measure's increased model penalty for features with more values compensates for any slight "by-chance" correlations with the class.

Figure 6.10 shows the learning curves for IB1, with and without feature selection, on concept A3. IB1 without feature selection learns slowly but consistently, unlike IB1 following feature selection, which shows no clear trend. Interestingly, feature selection using CFS-MDL results in more consistent performance than using either CFS-UC or CFS-Relief. CFS-MDL's performance is roughly 73%, which is the default accuracy for this version of the dataset. The tendency of CFS-UC and CFS-Relief to include more irrelevant features leads to an erratic performance which is often worse than the default accuracy for the dataset.

6.1.2 Redundant Attributes

These versions of the boolean domains A1-A3 each have nine redundant attributes added,

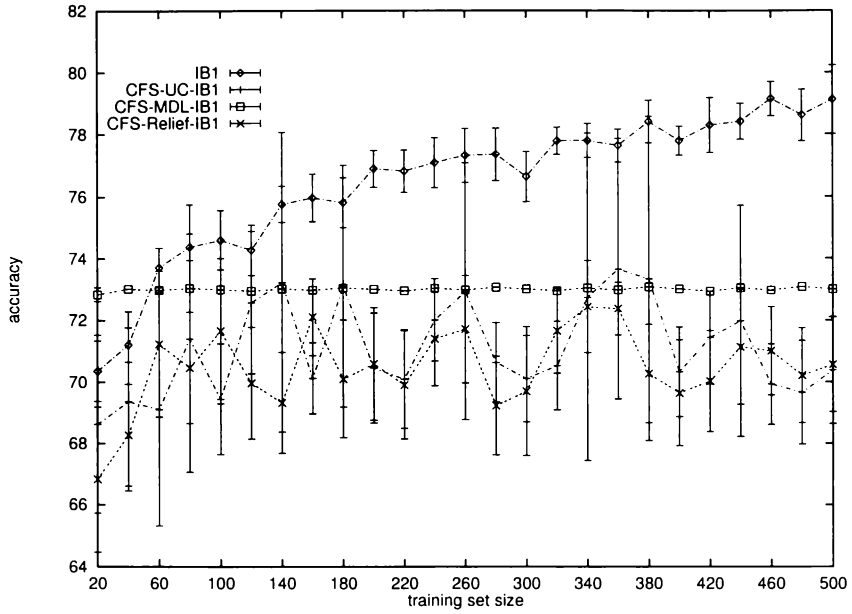


Figure 6.10: Learning curves for IB1, CFS-UC-IB1, CFS-MDL-IB1, and CFS-Relief-IB1 on concept A3 (with added irrelevant features).

for a total of twelve attributes. The redundant attributes are copies of attribute A in each of the domains[3]. There are three boolean redundant attributes, three 5-valued redundant attributes, and three 20-valued redundant attributes. In each level of redundant attribute (boolean, 5-valued, and 20-valued) there are two attributes which match attribute A 100% of the time (exact copies), and one attribute that matches A 80% of the time and is uniformly random for the remaining 20%. The multi-valued attributes are made redundant by joining the values of the attribute into two subsets. If an attribute has a values, then subsets $\{1, \ldots, (a \text{ div } 2)\}$ and $\{(a \text{ div } 2) + 1), \ldots, a)\}$ are formed. If attribute A's value is 0, then the redundant attribute's value is selected at random from the first subset; otherwise, its value is selected at random from the second subset.

Since there are six attributes that match attribute A 100% of the time, there are actually seven subsets that contain exactly three relevant features and no redundant features. The three attributes that match attribute A 80% of the time are treated as relevant but noisy, adding another three subsets that are not quite as good as the other seven.

Concept A1 Figure 6.11 shows the number of redundant attributes selected by the three

[3] Other combinations of redundant attributes—for example, some from A and some from B—were also tried. Results were similar to those reported here.

variations of CFS on concept A1. The average number of redundant attributes decreases rapidly for CFS-UC and CFS-MDL, with both including less than one redundant attribute after seeing only 40 training examples. CFS-Relief selects more redundant attributes, on average, than the other two variations. As with the corresponding dataset for concept A1 (with added irrelevant attributes) the sizes of the pure nodes for the three relevant attributes vary in this version as well. This time attribute A covers 499 examples when it has value 0, attribute B covers 488 examples, and attribute C covers 482 examples. Table 6.2 shows the feature-class correlations assigned to all the features by symmetrical uncertainty, MDL, and *relief* using the full 1000 instances for this version of concept A1. All three measures rank attribute A and its two binary copies highest. Symmetrical uncertainty and the MDL measure rank attribute B and C next highest on the list. *Relief*, however, ranks the two 5-valued attributes which match attribute A 100% of the time higher than either attribute B or C.

Relief's sensitivity to the size of an attribute's pure node in this domain has outweighed its bias against attributes with more values. Examination of the feature subsets selected by CFS-Relief shows that attributes H and I (the two 5-valued attributes) are often present. On the full dataset, the correlation between attributes H and I assigned by *relief* is lower (relative to the other features) than that assigned by symmetrical uncertainty or the MDL measure, which also helps explain why CFS-Relief often includes these features.

For smaller training set sizes (less than 220 examples), Figure 6.11 shows that, on average, CFS-MDL selects slightly more redundant features than CFS-UC. Examining the feature-feature correlation between either of the two 100% redundant boolean features and attribute A, for MDL, reveals a value of 0.982—not the value of 1.0 that symmetrical uncertainty assigns, which is what one would expect for a correlation between two features that are exact copies of each other. The explanation for this is again due to the MDL measure's extra cost for model complexity. The MDL measure can never achieve the upper bound of 1.0 due to the fact that Post_MDL, in Equation 4.12, can never be zero.

Figure 6.12 shows the number of relevant attributes selected by the three variations of CFS on concept A1. As in the case with irrelevant attributes, CFS-UC and CFS-MDL quickly asymptote to selecting three relevant features while CFS-Relief takes longer.

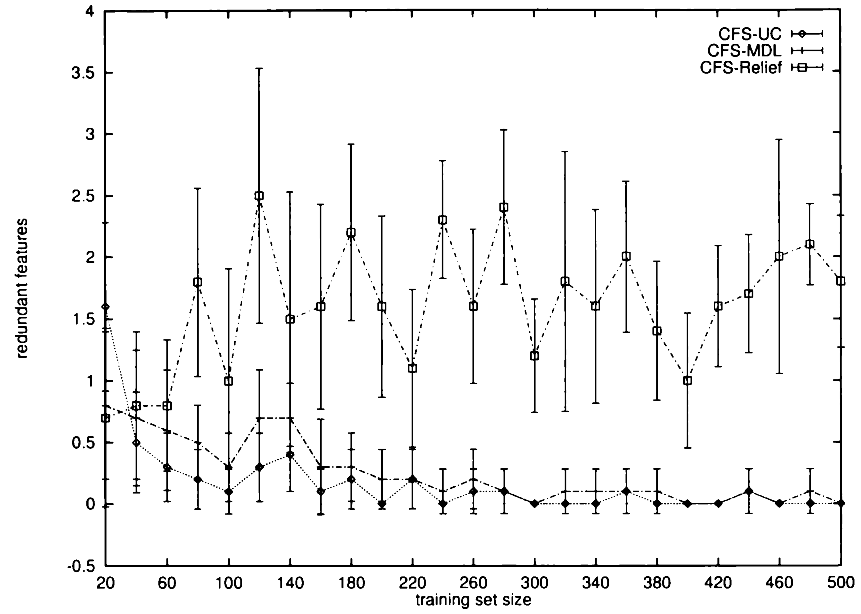


Figure 6.11: Number of redundant attributes selected on concept A1 (with added redundant features) by CFS-UC, CFS-MDL, and CFS-Relief as a function of training set size.

CFS-UC				CFS-MDL				CFS-Relief			
			symm.								
attr	vals	red.	uncert.	attr	vals	red.	MDL	attr	vals	red.	*relief*
A	2	100%	0.1957	*A*	2	100%	0.1985	*A*	2	100%	0.1046
E	2	100%	0.1957	*E*	2	100%	0.1985	*E*	2	100%	0.1046
F	2	100%	0.1957	*F*	2	100%	0.1985	*F*	2	100%	0.1046
B	2	0%	0.1893	*B*	2	0%	0.1917	*H*	5	100%	0.0907
C	2	0%	0.1860	*C*	2	0%	0.1881	*I*	5	100%	0.0896
D	2	80%	0.1280	*H*	5	100%	0.1460	*B*	2	0%	0.0882
H	5	100%	0.1092	*I*	5	100%	0.1442	*C*	2	0%	0.0795
I	5	100%	0.0677	*D*	2	80%	0.1301	*D*	2	80%	0.0738
L	20	100%	0.0691	*L*	20	100%	0.0904	*G*	5	80%	0.0614
G	5	80%	0.0689	*G*	5	80%	0.0894	*L*	20	100%	0.0137
K	20	100%	0.0647	*K*	20	100%	0.0783	*K*	20	100%	0.0130
J	20	80%	0.0447	*J*	20	80%	0.0405	*J*	20	80%	0.0072

Table 6.2: Feature-class correlations assigned by the three measures to all features in the dataset for A1 containing redundant features. The first three columns under each measure lists the attribute (A, B, and C are the original features), number of values the attribute has, and the level of redundancy.

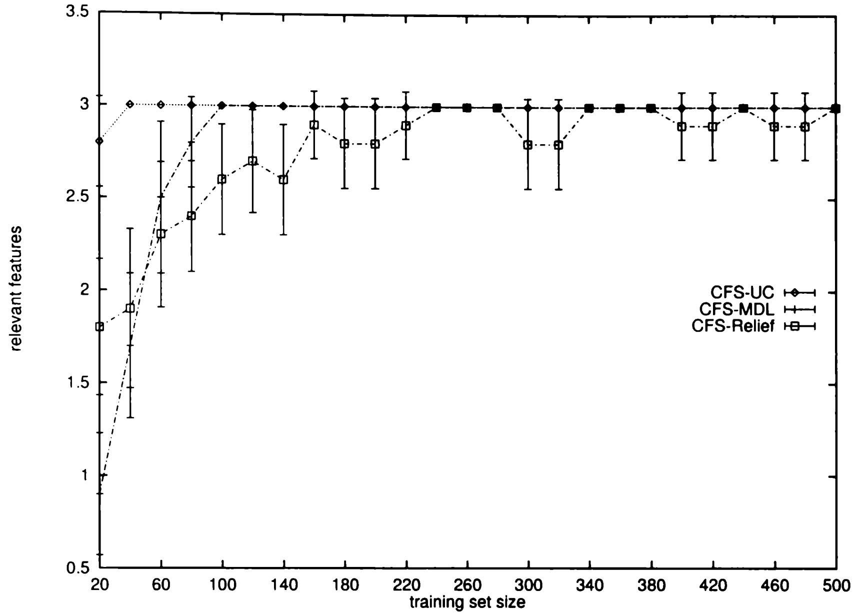


Figure 6.12: Number of relevant attributes selected on concept A1 (with added redundant features) by CFS-UC, CFS-MDL, and CFS-Relief as a function of training set size.

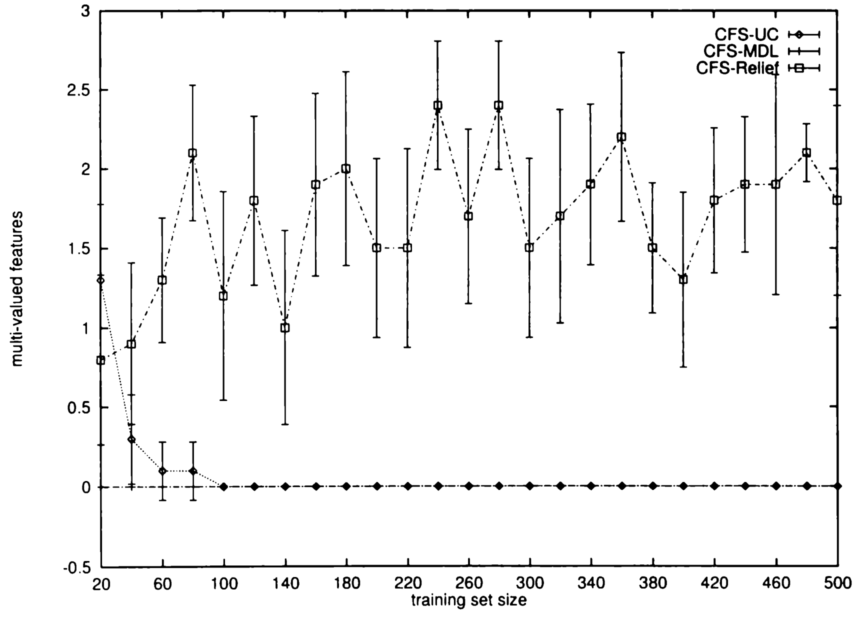


Figure 6.13: Number of multi-valued attributes selected on concept A1 (with added redundant features) by CFS-UC, CFS-MDL, and CFS-Relief as a function of training set size.

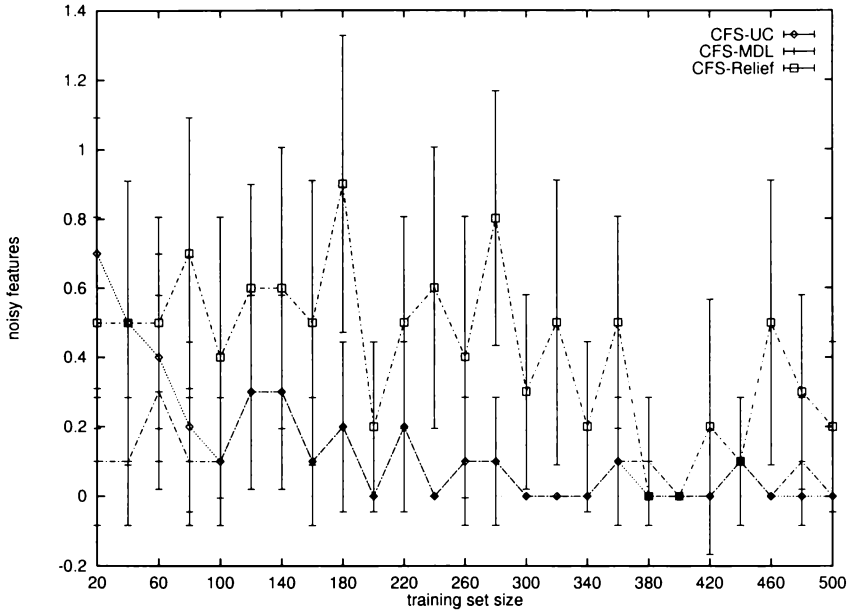


Figure 6.14: Number of noisy attributes selected on concept A1 (with added redundant features) by CFS-UC, CFS-MDL, and CFS-Relief as a function of training set size.

Figure 6.13 shows the average number of multi-valued features selected by the three variations of CFS. The results show that CFS-UC and CFS-MDL are effective at filtering the multi-valued features and preferring the boolean equivalents. CFS-Relief, due to its higher preference for the two 5-valued features, averages between 1 and 2 multi-valued features. The shape of the graph for CFS-Relief in Figure 6.11 is very similar to Figure 6.13, indicating that the redundant features selected are almost always multi-valued.

Figure 6.14 shows the average number of noisy features selected by the three variations of CFS (there are three copies of attribute A with 20% noise). The results show that all three variations of CFS prefer not to include noisy features. Less than one noisy feature is included on average by all three variations; CFS-UC and CFS-MDL perform better than CFS-Relief.

Figure 6.15 shows the learning curves for naive Bayes with and without feature selection on concept A1. For the sake of clarity, 90% confidence intervals have been omitted for CFS-Relief as they are much wider than for CFS-UC and CFS-MDL. All three variations of CFS enable naive Bayes to learn much faster than without feature selection. The points at which CFS-UC-nbayes and CFS-MDL-nbayes dip down slightly from 100% accuracy

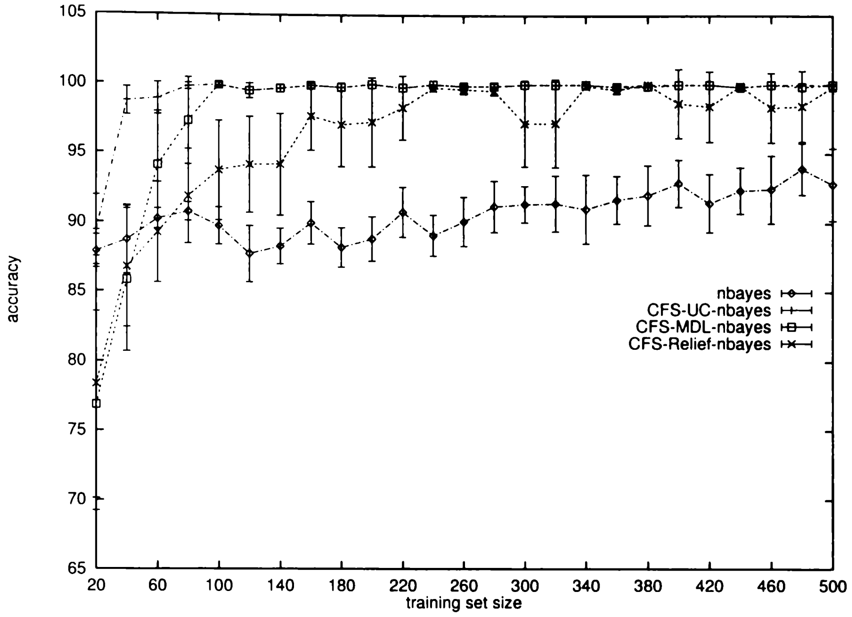


Figure 6.15: Learning curves for nbayes (naive-Bayes), CFS-UC-nbayes, CFS-MDL-nbayes, and CFS-Relief-nbayes on concept A1 (with added redundant features).

correspond roughly to the peaks of Figure 6.11, where more redundant attributes have been included in the selected subsets. The larger dips in the curve for CFS-Relief-nbayes correspond to the points on Figure 6.12, where CFS-Relief, on average, selects fewer than three relevant attributes.

Concept A2 As in the case with added irrelevant features, all three variations of CFS perform approximately equally well on concept A2. On average, less than one redundant feature is included in the subsets chosen by CFS after seeing only 40 training examples (Figure 6.16). CFS-MDL starts off by including slightly more redundant attributes than the other two methods. All three variations of CFS perform similarly in identifying the relevant attributes (Figure 6.17); all three relevant features are consistently identified after 360 training examples have been seen. This is reflected in the learning curves for naive Bayes with and without feature selection (Figure 6.18). Without feature selection naive Bayes does not improve its performance as more training examples are seen. With feature selection naive Bayes accuracy starts near 76% and rapidly increases to 100% after 220 training examples. The graphs for multi-valued and noisy attributes are not shown as very few (zero after 100 training examples) of these attributes were selected on this concept by

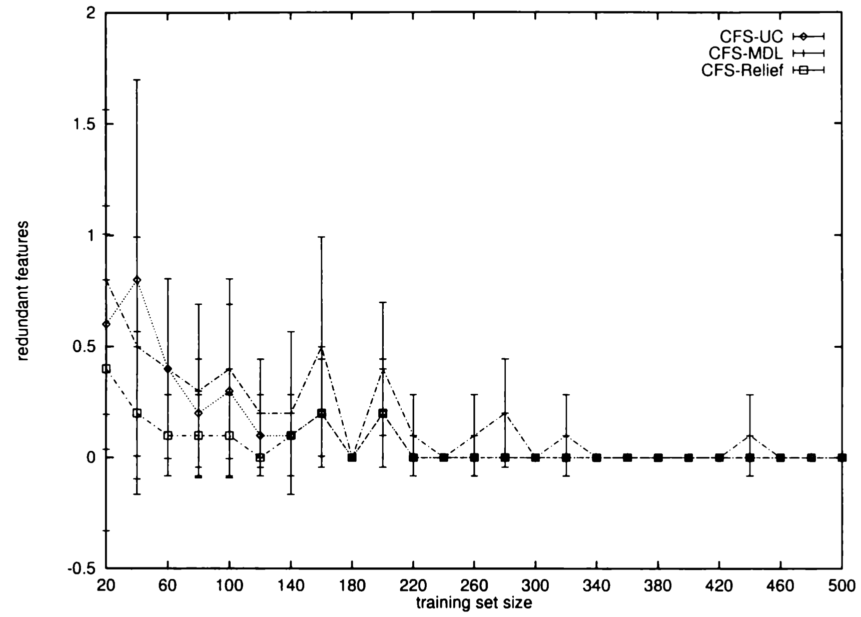


Figure 6.16: Number of redundant attributes selected on concept A2 (with added redundant features) by CFS-UC, CFS-MDL, and CFS-Relief as a function of training set size.

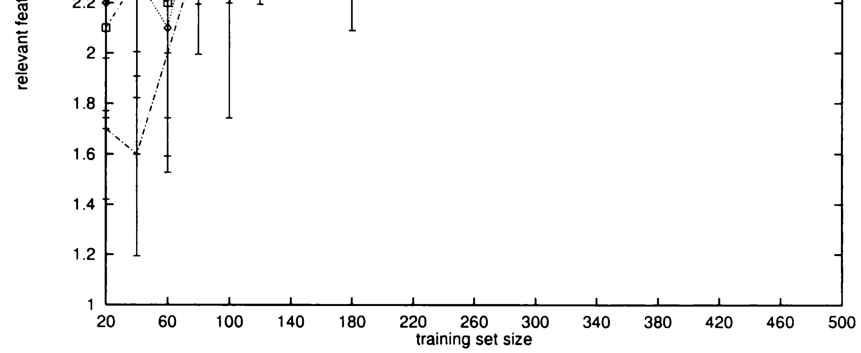


Figure 6.17: Number of relevant attributes selected on concept A2 (with added redundant features) by CFS-UC, CFS-MDL, and CFS-Relief as a function of training set size.

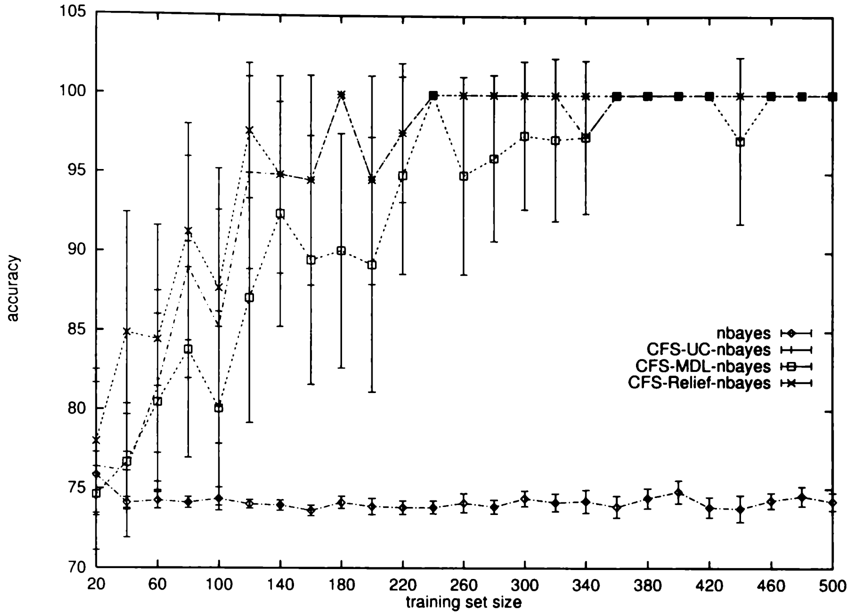


Figure 6.18: Learning curves for nbayes (naive Bayes), CFS-UC-nbayes, CFS-MDL-nbayes, and CFS-Relief-nbayes on concept A2 (with added redundant features).

all three variations of CFS.

Concept A3 The results for CFS on concept A3, with added redundant features, is very similar to the results for CFS on A3 with added irrelevant features. CFS is unable to distinguish between attributes and often fails to include all three relevant features. As in the case with added irrelevant features, CFS-UC and CFS-Relief show a tendency to include more features as the sample size increases, while CFS-MDL selects fewer features (often no features). The full set of graphs can be found in appendix B.

Figure 6.19 shows the learning curves for naive Bayes before and after feature selection. Naive Bayes cannot improve beyond the default accuracy (74.5%) for this version of concept A3 due to the extreme feature interaction. The accuracy of naive Bayes after feature selection by CFS-MDL reaches this maximum faster than the other versions due to the small number of features (on average) selected. When no features are present, naive Bayes predicts using the prior probabilities of the class values observed in the training

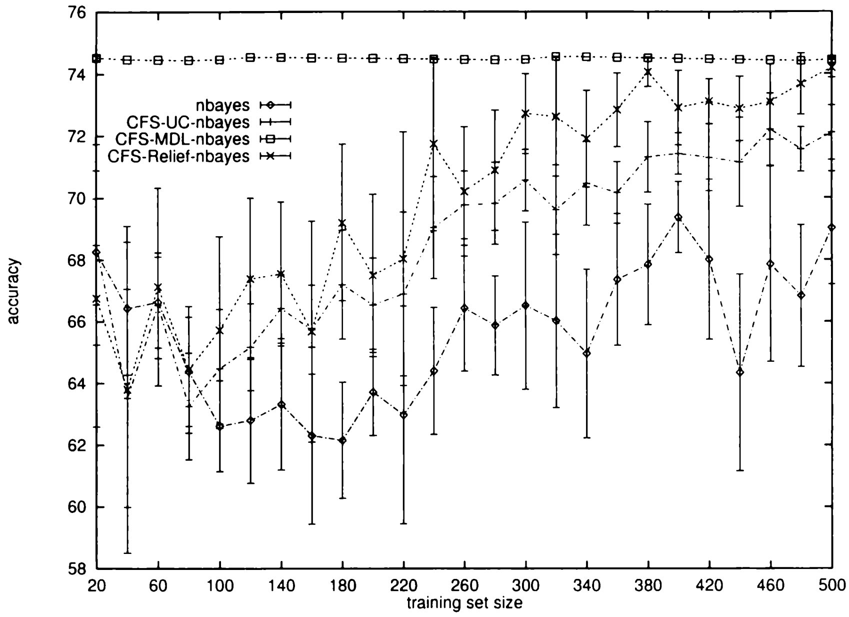


Figure 6.19: Learning curves for nbayes (naive Bayes), CFS-UC-nbayes, CFS-MDL-nbayes, and CFS-Relief-nbayes on concept A3 (with added redundant features).

data; since training and test sets are stratified, this will achieve the default accuracy.

6.1.3 Monk's problems

This section tests CFS on the three Monk's problems. Each problem uses the same representation and has six features. There are three relevant features in M1 and M3; M2 uses all six features.

There are 432 examples of each problem and each has a pre-defined training set. The training sets contain 124, 169, and 122 examples, respectively; the full datasets are used for testing [TBB$^+$91]. In the experiments below, training sets of the same size as the pre-defined sets are generated using the random subsampling method described in the previous chapter; tesing still uses the full dataset as done by Thrun et. al. [TBB$^+$91]. The results are averaged over 50 trials.

Table 6.3 shows the average number of features selected by the three variations of CFS on the Monk's problems. All three variations are unable to select all the relevant features for M1 and M2 due to the high order feature interactions. The jacket-colour feature is consistently selected for M1 and is one of the three relevant features in this concept. All six

Domain	CFS-UC	CFS-MDL	CFS-Relief
M1	1.0 ± 0.0	1.0 ± 0.0	1.0 ± 0.0
M2	2.9 ± 1.2	0.3 ± 0.6	2.8 ± 1.1
M3	2.0 ± 0.0	2.0 ± 0.0	1.9 ± 0.3

Table 6.3: Average number of features selected by CFS-UC, CFS-MDL, and CFS-Relief on the Monk's problems.

features are relevant for M2 and all interact. CFS assigns feature-class correlations close to zero for all features, which results in approximately half the features being selected by CFS-UC and CFS-Relief. CFS-MDL often assigns feature class-correlations less than zero, which accounts for it selecting fewer features than the other two. On M3, CFS-UC and CFS-MDL consistently choose body-shape and jacket-colour, which together give the second conjunction of the concept (jacket-colour $\neq$ blue and body-shape $\neq$ octagon). CFS-Relief occasionally omits one of these features.

Tables 6.4 – 6.6 show the results of three machine learning algorithms with and without feature selection on the Monk's problems.

Domain	naive Bayes	CFS-UC-nbayes	CFS-MDL-nbayes	CFS-Relief-nbayes
M1	72.55 ± 2.0	75.00 ± 0.0+	75.00 ± 0.0+	75.00 ± 0.0+
M2	62.75 ± 2.7	64.93 ± 2.0+	67.04 ± 0.7+	64.47 ± 2.2+
M3	97.17 ± 0.4	97.17 ± 0.4	97.17 ± 0.4	95.83 ± 4.6−

+, − statistically significant improvement or degradation

Table 6.4: Comparison of naive Bayes with and without feature selection on the Monk's problems.

Domain	IB1	CFS-UC-IB1	CFS-MDL-IB1	CFS-Relief-IB1
M1	76.71 ± 1.2	75.00 ± 0.0−	75.00 ± 0.0−	75.00 ± 0.0−
M2	79.06 ± 0.8	66.94 ± 0.6−	67.13 ± 0.0−	67.16 ± 1.7−
M3	79.41 ± 1.5	97.22 ± 0.0+	97.22 ± 0.0+	93.22 ± 13.7+

+, − statistically significant improvement or degradation

Table 6.5: Comparison of IB1 with and without feature selection on the Monk's problems.

CFS is able to improve the accuracy of naive Bayes on the first two Monk's problems by eliminating all or some of the interacting features in these concepts. On M1, the removal of the two interacting features (head-shape and body-shape), which together yield the first conjunct of the concept, allows 75% accuracy to be achieved with just the jacket-colour feature. On M2, removing features allows naive Bayes to approach the default

Domain	C4.5	CFS-UC-C4.5	CFS-MDL-C4.5	CFS-Relief-C4.5
M1	83.87 ± 6.2	75.00 ± 0.0−	75.00 ± 0.0−	75.00 ± 0.0−
M2	66.81 ± 0.8	67.06 ± 0.3+	67.10 ± 0.0+	67.10 ± 0.0+
M3	97.54 ± 1.8	96.42 ± 1.6−	96.42 ± 1.6−	95.26 ± 4.5−

+, − statistically significant improvement or degradation

Table 6.6: Comparison of C4.5 with and without feature selection on the Monk's problems.

accuracy for the dataset—CFS-MDL achieves the best result as it removes more features, on average, than the other two variations. On M3, CFS is unable to improve accuracy, indicating that the holding feature and the first conjunct of this concept is of no use to naive Bayes.

CFS degrades the performance of IB1 on the first two Monk's problems. Unlike naive Bayes, IB1 is able to make use of strongly interacting relevant features; removing these features results in worse performance. CFS degrades IB1's accuracy on M1 by less than 2%, however, IB1's performance without feature selection is affected by the three totally *irrelevant* features in this concept; if these features are removed, IB1 can achieve close to 100% accuracy. On M2, the more features that are removed by CFS, the closer IB1's accuracy is to the default for the dataset. CFS improves the accuracy of IB1 dramatically on M3 (from 79.41% to 97.22%). This is due to the removal of the totally irrelevant features. Accuracy can be improved further by approximately 1% if the holding feature (omitted by CFS) is included as well.

CFS degrades the performance of C4.5 on both M1 and M3. On M1, C4.5 is able to make use of the two interacting relevant features. Removing these two features results in 75% accuracy (the maximum achievable with just the jacket-colour feature). C4.5 is unable to learn the M2 concept and achieves less than the default accuracy of 67.1%. CFS-MDL and CFS-Relief are able to increase C4.5's performance to match the default accuracy.

6.1.4 Discussion

From experiments with CFS on these artificial domains, it can be concluded that feature selection based on correlation—specifically, the hypothesis proposed in Chapter 3 that good feature subsets contain features correlated with the class and uncorrelated with each other—can indeed select relevant features, and furthermore, can do so under conditions of moderate feature interaction, that is, as long as relevant features are *individually* predictive of the class at least some of the time. As expected, CFS fails to select relevant features in cases where there are strong feature interactions (features whose predictive ability is only apparent in the context of other features). While strong feature interaction is certainly possible in natural domains, the results presented in the next section and in appendix F suggest that it is not common—at least for datasets similar to those in the UCI collection.

The results show that CFS handles irrelevant and redundant features, noise, and avoids attributes with more values—traits that are likely to improve the performance of learning algorithms that are sensitive to such conditions.

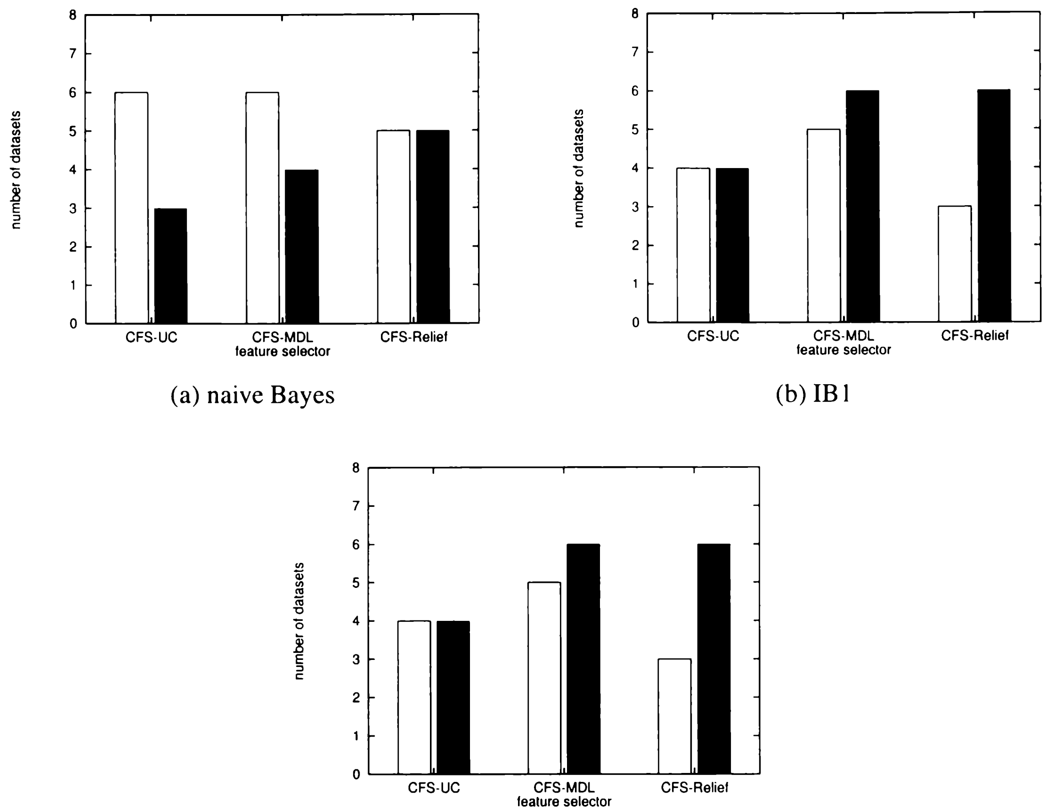
Of the three correlation measures tested with CFS, symmetrical uncertainty and MDL are superior to *relief*. In some cases, *relief* underestimates the worth of relevant attributes relative to others—a situation that can lead to relevant features being omitted from subsets selected by CFS. In other cases, attribute estimation by *relief* causes CFS to include more noise and redundancy in feature subsets.

The next section presents experiments designed to show if CFS's performance on artificial domains carries over to natural domains.

6.2 Natural Domains

This section describes the results of testing CFS on twelve natural domains. Since the relevant features are often not known in advance for these domains, the performance of learning algorithms with and without feature selection is taken as an indication of CFS's success in selecting useful features.

As with the experiments on artificial domains, stratified random subsampling is used to create training and test sets and the results reported are an average of 50 trials with each algorithm on each dataset.



(a) naive Bayes

(b) IB1

(c) C4.5

Figure 6.20: Number of natural domains for which CFS improved accuracy (left) and degraded accuracy (right) for naive Bayes (a), IB1 (b), and C4.5 (c).

For each machine learning algorithm, Figure 6.20 shows on how many natural domains accuracy was improved and degraded (significantly) by the three variations of CFS. The graphs show that CFS-UC and CFS-MDL perform better than CFS-Relief. CFS-MDL improves accuracy on the most datasets, followed by CFS-UC and CFS-Relief. CFS-UC degrades accuracy on fewer datasets than either CFS-MDL or CFS-Relief.

When the accuracies of the three variations of CFS are compared, CFS-UC is better (on average) than CFS-Relief for 3.6 datasets and worse (on average) for 2.6 datasets. When compared to CFS-MDL, CFS-UC is better (on average) for 3 datasets and worse (on average) for 2.3 datasets.

As was the case with the artificial domains, CFS-UC and CFS-MDL clearly outperform CFS-Relief. While the performance of CFS-UC and CFS-MDL was very similar on the artificial domains, CFS-UC is slightly better than CFS-MDL on the natural domains. The datasets for which CFS-MDL does not do as well as CFS-UC tend to be those with fewer training instances.

These results suggest that CFS-UC should be the preferred, or standard, version of CFS. For the remainder of this section, the results for CFS-UC are analyzed and discussed in detail. Full tabulated results comparing all three variations of CFS can be found in appendix C.

Table 6.7 shows the performance of naive Bayes, IB1, and C4.5 with and without feature selection by CFS-UC. Results using the 5×2cv test recomended by Dietterich [Die88] can be found in appendix D. CFS maintains or improves the accuracy of naive Bayes for nine datasets and degrades its accuracy for three. For IB1 and C4.5, CFS maintains or improves accuracy for eight datasets and degrades for four. Figure 6.21 shows that CFS reduces the size of the trees induced by C4.5 on nine of the twelve domains.

CFS appears to have difficulty on domains with the highest number of classes (especially au and sb). The worst performance is on audiology. CFS reduces the accuracy of naive Bayes on this dataset from 80.24% to 66.13%—the worst of the results. This domain has 24 classes and only 226 instances; however, CFS's performance here is much worse than on the primary tumour domain (pt), which has similar characteristics to audiology. This indicates that there are possibly other factors, apart from the number of classes and dataset size, affecting the performance of CFS.

Interestingly, CFS has resulted in worse performance by both IB1 and C4.5 on the chess end-game domain (kr), in contrast to naive Bayes, for which it improves performance. A similar situation occurs on the mushroom domain (mu), but to a lesser extent than on chess end-game. The improvement to naive Bayes indicates that there are some redundant features in these domains. On the chess end-game domain, CFS finds three features (out of 37) that give $\approx 90\%$ accuracy regardless of learning algorithm. However, IB1 and C4.5 are able to achieve $\approx 94\%$ and $\approx 99\%$ accuracy, respectively, without feature selection. Unlike audiology, this domain has only two classes and all but one of the features is binary. Furthermore, there are more than 3000 examples in this domain. This adds further

Dom	naive Bayes	CFS-nbayes	IB1	CFS-IB1	C4.5	CFS-C4.5
mu	94.75 ± 0.7	98.49 ± 0.1+	99.94 ± 0.1	98.48 ± 0.1−	99.59 ± 0.4	98.48 ± 0.1−
vo	90.25 ± 1.5	95.60 ± 1.0+	92.18 ± 1.3	95.60 ± 1.0+	95.23 ± 1.4	95.67 ± 1.0+
v1	87.20 ± 1.8	89.04 ± 1.7+	88.62 ± 2.0	88.35 ± 2.1	88.71 ± 2.0	88.37 ± 2.2
cr	78.21 ± 1.5	85.60 ± 1.0+	79.82 ± 1.9	85.61 ± 1.0+	83.69 ± 1.5	85.61 ± 1.0+
ly	82.12 ± 4.8	82.16 ± 6.1	79.89 ± 5.4	80.01 ± 4.8	75.80 ± 5.4	76.51 ± 5.3
pt	46.87 ± 3.1	45.83 ± 3.5−	39.63 ± 3.4	40.40 ± 2.8	40.99 ± 4.4	41.51 ± 3.5
bc	72.16 ± 2.7	71.86 ± 3.6	71.09 ± 3.8	70.67 ± 3.8	71.77 ± 3.3	70.97 ± 3.2
dna	89.21 ± 5.0	90.53 ± 4.5	80.31 ± 6.4	86.94 ± 4.7+	74.58 ± 6.5	77.20 ± 6.3+
au	80.24 ± 4.0	66.13 ± 3.2−	75.28 ± 3.2	67.60 ± 3.6−	78.48 ± 3.8	72.56 ± 2.8−
sb	91.30 ± 1.7	87.63 ± 2.5−	90.49 ± 1.6	84.24 ± 2.6−	89.16 ± 1.6	81.28 ± 2.9−
hc	83.13 ± 3.2	87.35 ± 3.7+	80.60 ± 3.2	86.89 ± 2.6+	84.02 ± 3.0	86.05 ± 3.5+
kr	87.33 ± 1.2	90.40 ± 0.6+	94.64 ± 0.8	90.41 ± 0.7−	99.16 ± 0.3	90.41 ± 0.7−

+, − statistically significant improvement or degradation

Table 6.7: Naive Bayes, IB1, and C4.5 with and without feature selection on 12 natural domains.

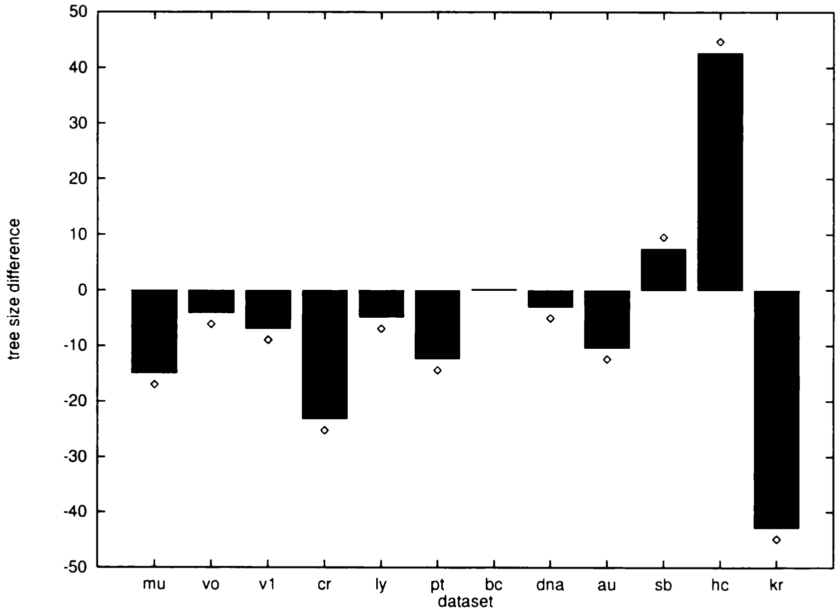


Figure 6.21: Effect of feature selection on the size of the trees induced by C4.5 on the natural domains. Bars below the zero line indicate feature selection has reduced tree size. Dots show statistically significant results.

support to the notion that there is some factor other than dataset size and number of classes affecting CFS's performance.

One possibility is that the best first search is getting trapped in a local maximum on the chess end-game domain and, perhaps, on the other domains where CFS has degraded accuracy. To see if this is the case, CFS was re-run on audiology, soybean, and chess end-game using a best first search with the stopping condition of the number of consecutive non-improving subsets increased from 5 to 500. Results show no significant improvement on any of the datasets, indicating that it is unlikely that the algorithm is getting trapped in local maxima.

A second possibility is that there are interacting features on the chess end-game domain. Examination of CFS-MDL's output on this domain show one trial where a fourth feature was included, resulting in $\approx$94% accuracy for all three learning algorithms *including naive Bayes*. While there may still be some interacting features, the fact that naive Bayes is able make use of it, shows that this fourth feature is not strongly interacting.

A third possibility is that CFS's heuristic merit function is too heavily biased in favour of small feature subsets, resulting in feature selection that is overly aggressive for the chess end-game domain and the other domains where CFS has degraded accuracy. Figure 6.22 shows the number of features in the original datasets and the number of features selected by CFS. On all but the primary tumour dataset (pt), CFS has reduced the number of features by more than half. In the case of audiology and chess end-game, the number of features has been reduced by more than 90%—a strong indication that feature selection has been too aggressive on these datasets.

To explore the bias of CFS's merit function, and to get an idea of how merit corresponds with actual accuracy of a learning algorithm, merit versus accuracy was plotted for randomly selected feature subsets on the natural domains. 50 subsets of $1, 2, 5, 10, 15$, etc. features were randomly selected from a single training split of each dataset (if there were fewer than 50 subsets possible for a given size, all were generated). For each subset, the heuristic merit was measured by CFS, using the training data, and the actual accuracy was estimated by first training a learning algorithm using the features in the subset, and then evaluating its performance on the test set[4]. Figure 6.23 shows plots of CFS-UC's merit versus naive Bayes' accuracy on a selection of datasets (chess end-game, horse colic, audiology, and soybean). Plots for the remaining datasets—and for when IB1 and C4.5 are used to measure accuracy—can be found in appendix E.

The first thing that is apparent from Figure 6.23 is that a correspondence between merit and actual accuracy does exist—even for domains on which feature selection by CFS degrades accuracy (audiology and soybean). Another thing that is apparent is that there are indeed a number of feature subsets for naive Bayes that result in close to 95% accuracy on the chess end-game domain (Figure 6.23a). Examining the size of the feature subsets represented by the points on these graphs reveals CFS's bias towards subsets with fewer features. For example, on the horse colic dataset (Figure 6.23b) this bias is effective—the rightmost (highest merit) point on the graph for horse colic represents a subset containing 2 features which together give an accuracy of 89.68% on the test set. There are three subsets that have close to this accuracy, two of which contain 10 features and one 5 features; both are assigned a much lower merit than the 2 feature subset. On the other hand, for audiology (Figure 6.23c) and soybean (Figure 6.23d) CFS favours smaller, moderately accurate subsets over larger subsets with higher accuracy. From the graphs for audiology and soybean, it is clear that there are many subsets with high accuracy that have merit close to the rightmost (highest merit) points.

From the above analysis it can be concluded that CFS's poor performance on some datasets can be traced to the merit formulation rather than the search. Its aggressive bias favouring small feature subsets may result in some loss of accuracy.

To see if performance could be improved on the datasets that cause CFS difficulty, a method of merging subsets was introduced, with the aim of increasing feature set size by incorporating those subsets whose merit is close to the best subset. The method works as follows: instead of simply returning the best subset at the end of the search, the top 50 subsets (sorted in order of merit) discovered during the search are recorded. At the end of

[4] Averaging runs on multiple training and testing splits would give more reliable estimates of merit and accuracy for feature subsets, but is time consuming. Using a single training and test set provides a rough idea of the merit and accuracy, but may generate outliers, that is, feature subsets that, by chance, are predictive of the test set, or have high merit with respect to the training set.

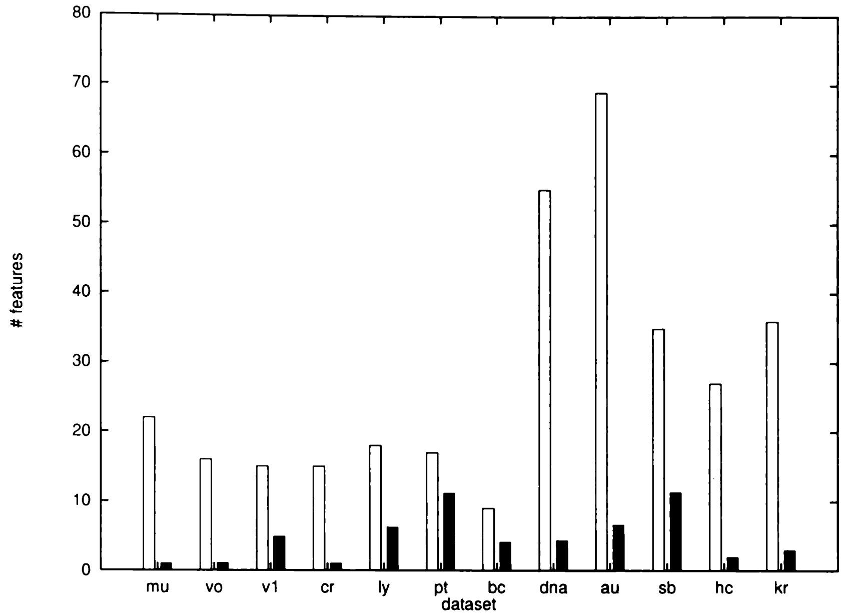


Figure 6.22: The original number of features in the natural domains (left), and the average number of features selected by CFS (right).

the search the second best subset is merged with the best subset and the merit of this new composite subset is recalculated. If the merit is within 10% of the merit of the best subset, the composite is accepted. The third best subset is then merged with the composite from the previous step, and so forth. This continues as long as the merit of the new composite is within 10% of the best subset.

Table 6.8 shows the accuracy of naive Bayes, IB1, and C4.5 with and without feature selection by CFS-UC using this subset-merging scheme. For naive Bayes there are now eleven datasets for which accuracy is maintained or significantly improved and only one dataset for which accuracy is significantly degraded. Merging feature subsets has made the result for lymphography better (where before there was no significant difference) and the results for soybean and primary tumour no different (where before both worse). For IB1, merging feature subsets has changed the result on the audiology domain—there is now no significant difference in accuracy on this domain (where before it was worse). For C4.5, merging subsets has degraded performance. The main difference is on the horse colic domain, which has gone from being improved to degraded. This is surprising, as merging subsets improves CFS's performance for naive Bayes on this dataset.

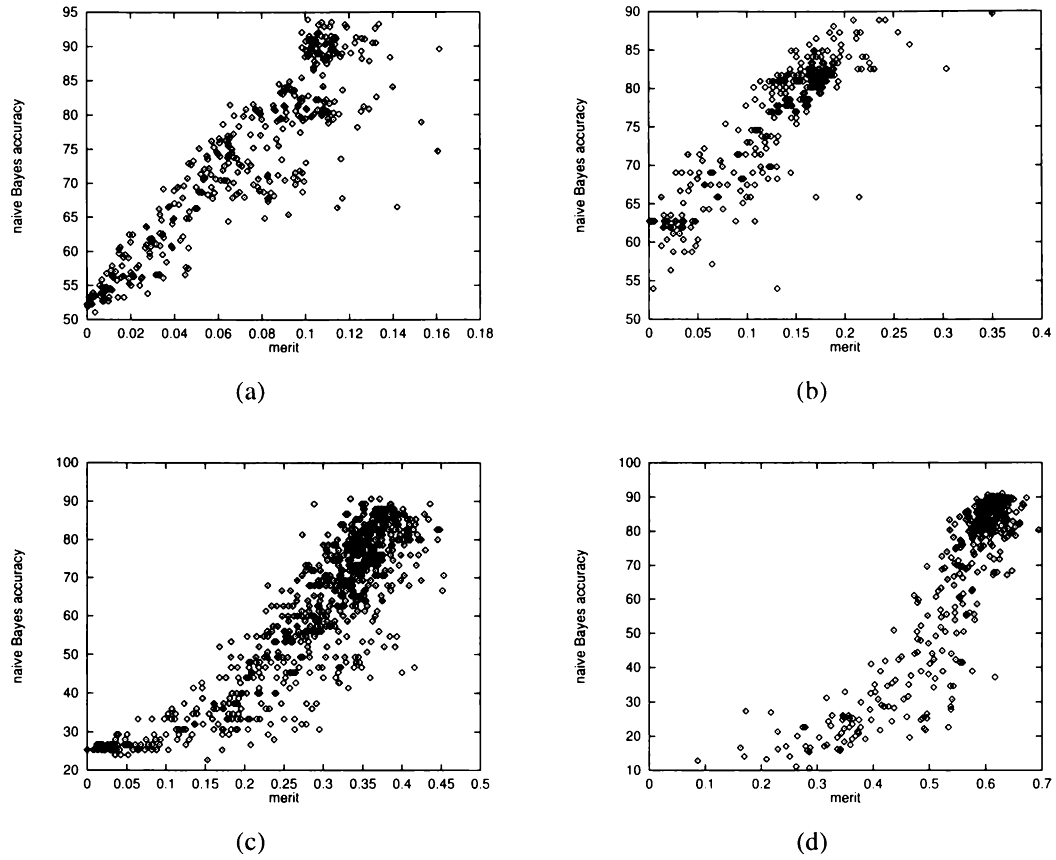


Figure 6.23: Heuristic merit (CFS-UC) vs actual accuracy (naive Bayes) of randomly selected feature subsets on chess end-game (a), horse colic (b), audiology (c), and soybean (d). Each point represents a single feature subset.

It turns out that C4.5's attribute selection heuristic is responsible for the degraded performance. C4.5 chooses an attribute to split on by selecting from among those attributes with an average-or-better information gain, the attribute that maximises the gain ratio. When presented with the full feature set, C4.5 chooses the attribute "surgery" to split on at the root of the tree as this attribute has above average gain and the highest gain ratio (information gain normalised by the entropy of the attribute). CFS chooses this attribute and one other ("type of lesion 1") which also has a high gain ratio. Merging subsets often adds "HospitalID" which has a large number of distinct values and a reasonably high gain ratio but in actuality is a poor predictor. This last attribute has by far the highest information gain due to its many values. When all features are present, the average information gain is low enough (due to there being many poor attributes with low gain) for "surgery" to have above average information gain and thus be preferred over "HostpitalID". In the case of the reduced subset provided by merging, the average information gain is much higher, making "HostpitalID" the *only* attribute with above average information gain. Consequently, "HostpitalID" is chosen for the root of the tree and a low accuracy results.

Dom	naive Bayes	CFS-nbayes	IB1	CFS-IB1	C4.5	CFS-C4.5
mu	94.75 ± 0.7	97.53 ± 0.8+	99.94 ± 0.1	99.68 ± 0.3−	99.59 ± 0.4	99.37 ± 0.6−
vo	90.25 ± 1.5	92.93 ± 2.0+	92.18 ± 1.3	94.34 ± 1.6+	95.23 ± 1.4	95.58 ± 1.0+
v1	87.20 ± 1.8	87.71 ± 1.9+	88.62 ± 2.0	88.51 ± 2.0	88.71 ± 2.0	88.71 ± 2.0
cr	78.21 ± 1.5	85.59 ± 1.0+	79.82 ± 1.9	85.60 ± 1.0+	83.69 ± 1.5	85.61 ± 1.0+
ly	82.12 ± 4.8	84.94 ± 5.1+	79.89 ± 5.4	80.40 ± 4.5	75.80 ± 5.4	76.90 ± 5.5
pt	46.87 ± 3.1	46.87 ± 3.1	39.63 ± 3.4	39.63 ± 3.4	40.99 ± 4.4	40.99 ± 4.4
bc	72.16 ± 2.7	71.94 ± 2.8	71.09 ± 3.8	70.57 ± 4.2	71.77 ± 3.3	71.90 ± 3.3
dna	89.21 ± 5.0	89.47 ± 5.0	80.31 ± 6.4	83.93 ± 5.5+	74.58 ± 6.5	76.05 ± 6.4+
au	80.24 ± 4.0	75.55 ± 3.7−	75.28 ± 3.2	74.37 ± 4.1	78.48 ± 3.8	77.14 ± 3.8−
sb	91.30 ± 1.7	91.18 ± 1.5	90.49 ± 1.6	87.91 ± 1.5−	89.16 ± 1.6	86.80 ± 1.8−
hc	83.13 ± 3.2	88.76 ± 2.5+	80.60 ± 3.2	85.45 ± 2.6+	84.02 ± 3.0	78.79 ± 10.9−
kr	87.33 ± 1.2	90.20 ± 2.0+	94.64 ± 0.8	94.10 ± 0.5−	99.16 ± 0.3	94.13 ± 0.5−

+, − statistically significant improvement or degradation

Table 6.8: Comparison of three learning algorithms with and without feature selection using merged subsets.

Figure 6.24 shows the difference in accuracy between CFS *with* merged subsets and CFS *without* merged subsets for the three learning algorithms. Large improvements in accuracy can clearly be seen on the lymphography, audiology, soybean, and chess end-game datasets. However, it is also apparent that merging feature subsets degraded results on some datasets as well (most notably on vote, vote1, dna, and horse colic). This is most apparent for naive Bayes, which has the most significant degradations[5] (mushroom, vote, and vote1 shown in Figure 6.24). These results suggest that, while more useful features are being included through merging subsets, some harmful redundancy is also being included.

One question that immediately springs to mind is: why is merging feature subsets necessary? If other subsets contain features that are useful, and can increase the accuracy of the best subset found during the search, then why are these features not included in the best subset? To shed some light on this question, the chess end-game dataset is examined. Recall that there are three features selected by CFS that give $\approx 90\%$ accuracy for all

[5] Although significantly degraded by merging subsets, these results are still significantly better than naive Bayes without feature selection (see Table 6.8)

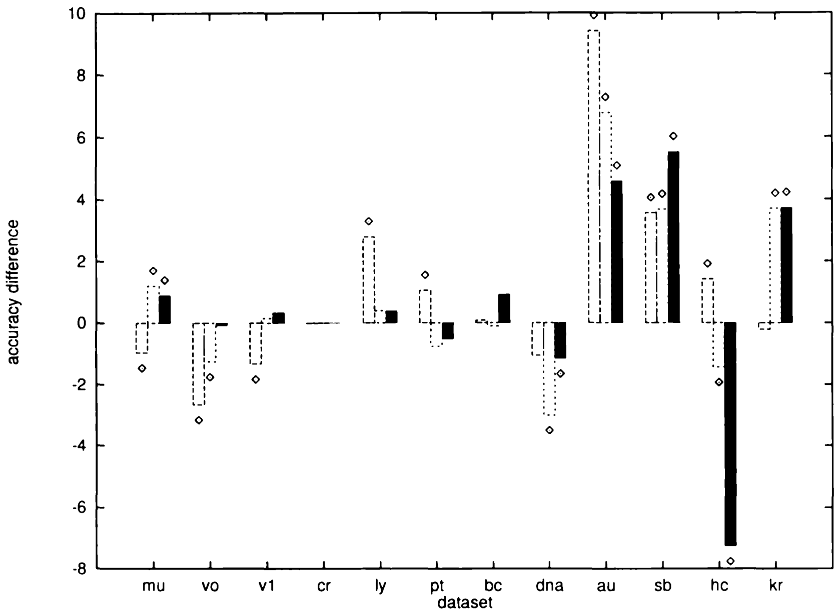


Figure 6.24: Absolute difference in accuracy between CFS-UC with merged subsets and CFS-UC for naive Bayes (left), IB1 (middle), and C4.5 (right). Dots show statistically significant results.

three learning algorithms, and there was one trial out of 50 in which CFS-MDL included a fourth feature that gives $\approx 94\%$ accuracy. This indicates that this fourth feature may be very close to the borderline for acceptance into the best subset and, in fact, it is included on all trials by CFS-UC and CFS-MDL using merged subsets.

CFS-UC		CFS-MDL	
	symmetrical		
attribute	uncertainty	attribute	MDL
22	0.2354	22	0.2396
11	0.1146	11	0.1132
34	0.1008	34	0.0991
33	0.0475	33	0.0630
9	0.0454	30	0.0619
16	0.0380	9	0.0445
17	0.0303	17	0.0399
30	0.0286	16	0.0351

Table 6.9: Top eight feature-class correlations assigned by CFS-UC and CFS-MDL on the chess end-game dataset.

Table 6.9 shows the top eight feature-class correlations calculated by symmetrical uncertainty and MDL on the entire chess end-game dataset. The first three features (22, 11, and 34) are consistently selected by CFS. They have the highest correlation with the class and have very low inter-correlation (0.007, 0.001, and 0.01 between features 22 and 11, 22 and 34, and 11 and 34 respectively). Feature 33 is the feature that, when added to 11, 22, and 34, gives 94% accuracy. As can be seen from the table, CFS-MDL assigns a higher correlation (comparative to the other features) to this feature than CFS-UC, which might explain why this feature was actually included in one trial by CFS-MDL and not by CFS-UC. Using 1R to produce a rule for attribute 33 on the full dataset gives:

Rule for attribute-33:

```
        class(NoWin) :- attribute-33(t). (covers 155 out of 175 examples)
        class(Won) :- attribute-33(f). (covers 1649 out of 3021 examples)
```

The first clause of this rule is highly predictive of a small number of examples—it achieves 88.5% accuracy over 175 examples. The 155 examples it gets correct account for 4.8% of the dataset. It is possible that the first clause of this rule—when attribute 33 has value "true"—is responsible for the 4% improvement in the learning schemes' accuracy on this dataset. To see if this is indeed the case, the predictions of naive Bayes using the feature subset $\{11, 22, 34\}$ and feature subset $\{11, 22, 33, 34\}$ are compared using a random training and testing split of the dataset. Results show a 4.1% improvement that corresponds *entirely* to the instances where attribute 33 has the value "true". Furthermore, there are no new errors introduced through the inclusion of attribute 33. The correlations between attribute 33 and attributes 11, 22, and 34 are very low, indicating that very little redundancy is introduced with this feature.

Analyzing attribute 30 reveals a similar pattern. The 1R rule for this feature shows that it has one value that covers 47 out of 47 examples—again a very small percentage of the instance space. Examining the predictions of naive Bayes with and without this feature included (in the same manner as above) shows a 1.4% improvement in accuracy that is solely attributable to this single value of the attribute; again, no new errors are introduced and the correlations between attribute 30 and 11, 22, and 34 are very low. Attribute 9 is a different story. Including this attribute results in accuracy degrading 4%. Analyzing this feature in the same manner as the previous two shows that it has one value that covers 486 out of 696 examples. This value is responsible for $\approx$ 1% of the test set that was originally misclassified to be classified correctly, but, unfortunately, it causes $\approx$ 5% of the test set to be misclassified. The correlations between attributes 9 and 11, 22, and 34 are much higher than those of attribute 33 and 30, indicating that too much redundancy has been introduced, which in turn affects the accuracy of naive Bayes.

The above analysis indicates that some datasets may contain features that have a few values that are strongly predictive locally (in a small area of the instance space), while their remaining values may have low predictive power, or be irrelevant, or partially correlated with other features. Since CFS measures a feature's merit globally (over the entire training instance space), its bias toward small feature subsets may prevent these features from being included—especially if they are overshadowed by other strong, globally predictive features. A number of features such as this could cumulatively cover a significant proportion of a dataset[6]. CFS's large improvement on the audiology dataset through the use of merged feature sets supports this conjecture. Inspection of the 1R rules produced for the audiology dataset reveals a number of features with values that are highly predictive of a small number of instances. While not all of these features may be useful, it is likely that some of them cover instances not covered by the stronger features.

The experiments presented in this section show that CFS's ability to select useful features does carry over from artificial to natural domains. Out of the three learning algorithms, CFS improves naive Bayes the most. This is most likely due to the fact that CFS deals effectively with redundant attributes, and, like naive Bayes, treats features as independent of one another given the class.

Analysis of the results on the natural domains has revealed a weakness with CFS. Attributes that are locally predictive in small areas of the instance space may not be selected—especially if they are overshadowed by other strongly predictive features. From this it can be concluded that subsets selected by CFS should not be treated as definitive—instead, they represent a good indication of the most important features in a dataset.

Appendix F presents results for CFS-UC applied to the suite of UCI domains used for evaluation as part of the WEKA (Waikato Environment for Knowledge Analysis) workbench [HDW94]. These results show a similar pattern as those for the smaller set of domains analyzed here.

[6] Because C4.5 subdivides the training instance space as it constructs a decision tree, it has a greater chance of identifying these features, which explains its superior performance on the chess end-game dataset.

6.3 Chapter Summary

This chapter presents experiments which test the claim that CFS is a method by which redundant and irrelevant features can be removed from learning data, and that CFS can be of use to common machine learning algorithms.

Experiments on artificial domains show CFS to be effective at screening both irrelevant and redundant features and, as long as there are no extreme feature interactions, CFS is able to quickly identify relevant features. Furthermore, the results of section 6.1.2 show CFS to prefer relevant features with fewer values and less noise. Learning curves show how ML algorithms that increase in accuracy slowly on these artificial domains, in the presence of irrelevant and redundant information, can have their accuracy dramatically improved by using CFS to select features.

Experiments on a selection of natural domains show that CFS can be of use to ML algorithms in terms of improving accuracy and, in the case of C4.5, improving the comprehensibility of the induced model. Since the dimensionality of the data is reduced, all three learning algorithms execute noticeably faster. Examination of cases where using CFS to select features results in worse performance reveals a shortcoming in the the approach. Because correlations are estimated globally (over all training instances), CFS tends to select a "core" subset of features that is highly predictive of the class, but may fail to include subsidiary features that are locally predictive in a small area of the instance space. Experiments show that a version of CFS that merges subsets to include features from subsets with merit close to the best subset is able to incorporate some of these subsidiary features. However, merging feature subsets may also allow some harmful redundancy to be included.

Of the three versions of CFS tested, the versions using the symmetrical uncertainty coefficient and the MDL measure perform the best overall. By selecting fewer features, CFS-MDL tends to be more cautious than CFS-UC when there are few training instances. This is why CFS-UC performs slightly better than CFS-MDL. For larger datasets CFS-MDL performs as well, if not slightly better, than CFS-UC.

Chapter 7
Comparing CFS to the Wrapper

This chapter compares CFS's performance with that of the wrapper approach to feature selection. The wrapper is one of the simplest feature selectors conceptually (though not computationally) and has been found to generally out-perform filter methods [JKP94, CF94, AB94]. Comparing CFS to the wrapper is a challenging test because the wrapper is driven by the estimated performance of a target learning algorithm and is tuned to its inductive bias.

7.1 Wrapper Feature Selection

The rationale behind wrapper feature selectors is that the induction method that will ultimately use the feature subset should provide a better estimate of accuracy than a separate measure that has a different inductive bias. It is possible that the "optimal" feature subset for a given induction algorithm does not contain all the relevant features[1], and advocates of the wrapper approach claim that using the target learning algorithm as the feature evaluation function is the easiest way to discover this [Koh95b, KJ96]. It is undoubtedly true that, short of designing a feature evaluation measure that mimics the behaviour of a particular induction algorithm, using the induction algorithm itself as the measure stands the best chance of identifying the "optimal" feature subset. However, wrapper feature selectors are not without fault—cross-validation accuracy estimates are highly variable when the number of instances is small (indicative of overfitting) [Koh95b], and cross-validation is prohibitively slow on large datasets [CF94, ML94].

[1] Kohavi [Koh95b] gives an example where withholding a relevant attribute from naive Bayes on a m-of-n concept results in better performance than including it.

The $\mathcal{MLC}$++ machine learning library [KJL+94] was used to provide results for the wrapper, as this library was used in Kohavi's feature selection experiments reported in the literature. The version of $\mathcal{MLC}$++ utilities used has an implementation of the naive Bayes learner[2] built in and provides support for C4.5 through scripts. Unfortunately, the built in version of IB1 does not support nominal features, which makes it unsuitable for use on the datasets chosen for evaluating CFS, nor is there easy support for calling an external learning algorithm (apart from C4.5). For that reason, IB1 was not used in the experiments described in this chapter. Nevertheless, C4.5 and naive Bayes represent two diverse approaches to learning, and results with these algorithms provide an indication as to how the feature selection methods will generalize to other algorithms.

In order to allow a fair comparison between CFS and the wrapper, the same training and testing splits used to generate the results shown in in chapter 6 are presented to $\mathcal{MLC}$++. Furthermore, the same search strategy and stopping condition are used. Accuracy estimation for the wrapper is achieved though 10-fold cross validation of the training set. Figure 7.1 shows the stages of the wrapper feature selector.

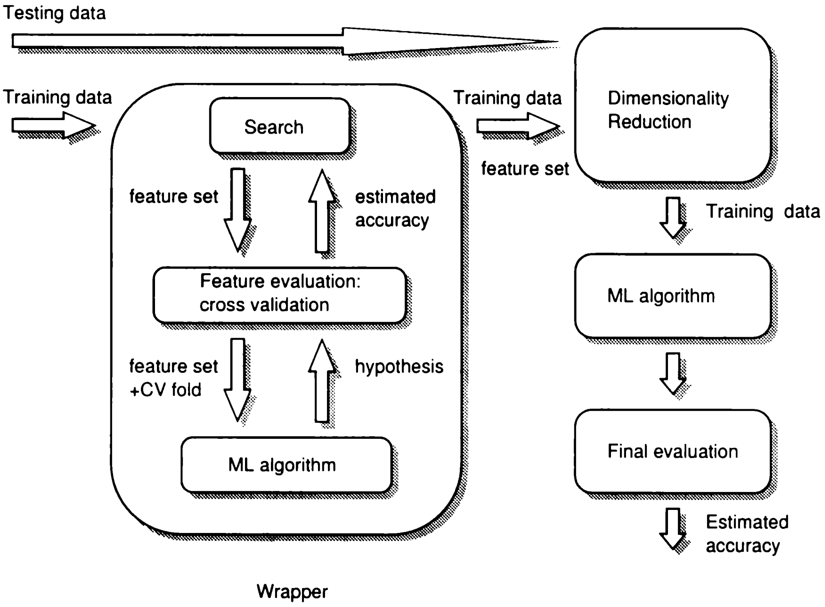


Figure 7.1: The wrapper feature selector.

[2] The version of naive Bayes in $\mathcal{MLC}$++ is the implementation that is used in all the experiments described in Chapter 6

7.2 Comparison

Table 7.1 shows the accuracy of naive Bayes without feature selection and naive Bayes with feature selection by the wrapper and CFS-UC on all the domains. Training set sizes for the artificial datasets, with added irrelevant and redundant features (A1i–A3i and A1r–A3r respectively), were chosen by examining the learning curves in Chapter 6. For concepts A1 and A2, the point at which both all the relevant and none of the irrelevant/redundant features are selected by CFS was noted and training sets of this size are used (100, 300, 300, and 400 instances for A1i, A2i, A1r, and A2r respectively). As CFS is unable to select the relevant features on A3, the training set size for A3i and A3r is set to 500 (exactly half of the dataset).

From the table it can be seen that the wrapper improves naive Bayes on thirteen datasets and degrades on six. CFS improves naive Bayes on fourteen datasets and degrades on three.

It is clear that the wrapper has difficulty on those datasets with fewer training instances (lymphography, primary tumour, breast cancer, dna-promoter, and A1i). This is especially noticeable on the artificial domains with added irrelevant features where it actually degrades naive Bayes performance. On the equivalent domains with added redundant attributes its performance is much better. On these datasets all features are relevant, suggesting that the combination of few training instances and irrelevant features is to blame for the wrapper's performance on A1i and A2i. The wrapper tends to outperform CFS on datasets with more training examples (mushroom, soybean, and chess end-game) and on those domains with features that are locally predictive in small areas of the instance space (chess end-game and audiology). Where the wrapper does better it always selects more features than CFS. The features in common between CFS and the wrapper are fairly stable between trials on these datasets. This also suggests that the wrapper is able to detect additional locally predictive features where CFS can not.

Unlike the wrapper, CFS does not need to reserve part of the training data for evaluation purposes and, in general, tends to do better on the smaller datasets than the wrapper.

Figure 7.2 shows the difference in accuracy between CFS and the wrapper. Bars above the zero line show where naive Baye's average accuracy using feature subsets selected by CFS is higher than its average accuracy using feature subsets selected by the wrapper. CFS is better than the wrapper for eight domains and is worse for seven domains. The wrapper outperforms CFS on the M2 domain because it discovers that the best performance for naive Bayes is given by eliminating all the features. Chapter 6 showed that CFS with the MDL measure (CFS-MDL) achieves this result also.

Dom	naive Bayes	wrapper	CFS
mu	94.75 ± 0.7	98.86 ± 0.4+	98.49 ± 0.1+
vo	90.25 ± 1.5	95.24 ± 1.2+	95.60 ± 1.0+
v1	87.20 ± 1.8	88.95 ± 2.2+	89.04 ± 1.7+
cr	78.21 ± 1.5	85.16 ± 1.2+	85.60 ± 1.0+
ly	82.12 ± 4.8	76.00 ± 5.0−	82.16 ± 6.1
pt	46.87 ± 3.1	42.31 ± 3.9−	45.83 ± 3.5−
bc	72.16 ± 2.7	70.96 ± 3.4−	71.86 ± 3.6
dna	89.21 ± 5.0	82.05 ± 8.1−	90.53 ± 4.5
au	80.24 ± 4.0	79.33 ± 3.4	66.13 ± 3.2−
sb	91.30 ± 1.7	92.99 ± 1.5+	87.63 ± 2.5−
hc	83.13 ± 3.2	87.70 ± 2.5+	87.35 ± 3.7+
kr	87.33 ± 1.2	94.36 ± 0.5+	90.40 ± 0.6+
A1i	91.95 ± 0.9	87.44 ± 0.0−	99.49 ± 1.8+
A2i	94.96 ± 1.8	88.31 ± 12.8−	100.00 ± 0.0+
A3i	71.50 ± 1.5	72.97 ± 0.2+	72.04 ± 1.3+
A1r	84.31 ± 3.0	100.00 ± 0.0+	100.00 ± 0.0+
A2r	73.74 ± 1.0	99.89 ± 0.8+	100.00 ± 0.0+
A3r	68.68 ± 3.1	74.67 ± 1.5+	72.28 ± 1.6+
M1	72.55 ± 2.0	74.27 ± 1.8+	75.00 ± 0.0+
M2	62.75 ± 2.7	67.02 ± 0.6+	64.93 ± 2.0+
M3	97.17 ± 0.4	97.26 ± 0.4	97.17 ± 0.4

+, − statistically significant improvement or degradation

Table 7.1: Comparison between naive Bayes without feature selection and naive Bayes with feature selection by the wrapper and CFS.

Table 7.2 shows the CPU time taken (as measured on a Sparc server 1000) to complete *one* trial on each dataset by the wrapper and CFS[3]. As can be seen, CFS is much faster than the wrapper.

Figure 7.3 shows the number of features selected by the wrapper and CFS. CFS generally selects a similar sized feature set as the wrapper. In many cases the number of features is reduced by more than half by both methods.

[3] CPU times reported for CFS are for a non-optimized version (Chapter 4 suggests some simple methods of optimization).

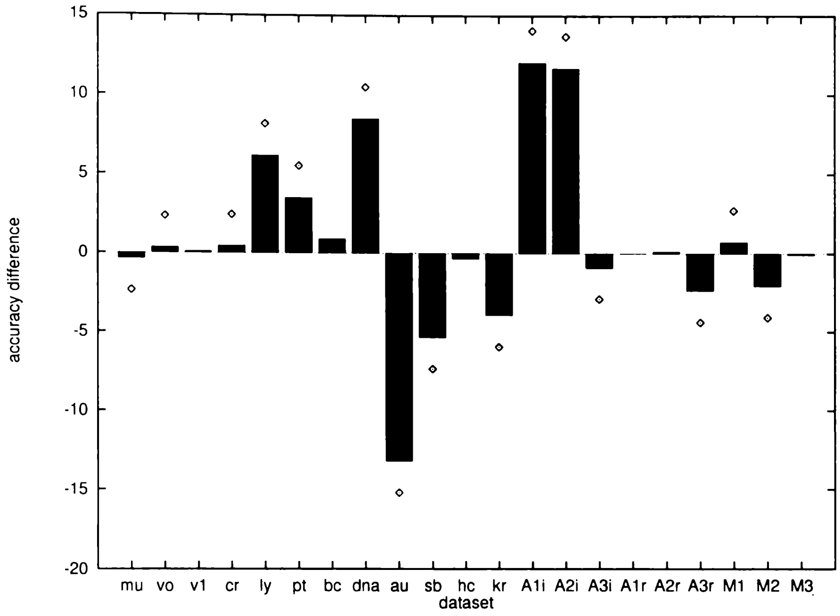


Figure 7.2: Comparing CFS with the wrapper using naive Bayes: Average accuracy of naive Bayes using feature subsets selected by CFS minus the average accuracy of naive Bayes using feature subsets selected by the wrapper. Dots show statistically significant results.

Domain	wrapper	CFS
mu	2154.0	1.0
vo	193.0	< 1.0
v1	397.0	< 1.0
cr	411.0	< 1.0
ly	187.1	< 1.0
pt	3036.0	< 1.0
bc	123	< 1.0
dna	1145.0	< 1.0
au	5362.0	1.0
sb	8904.0	1.0
hc	1460.0	1.0
kr	9427.0	8.0
A1i	83.0	< 1.0
A2i	598.0	1.0
A3i	357.0	1.0
A1r	328.0	< 1.0
A2r	237.0	< 1.0
A3r	562.0	< 1.0
M1	22.0	< 1.0
M2	26.0	< 1.0
M3	26.0	< 1.0

Table 7.2: Time taken (CPU units) by the wrapper and CFS for a single trial on each dataset.

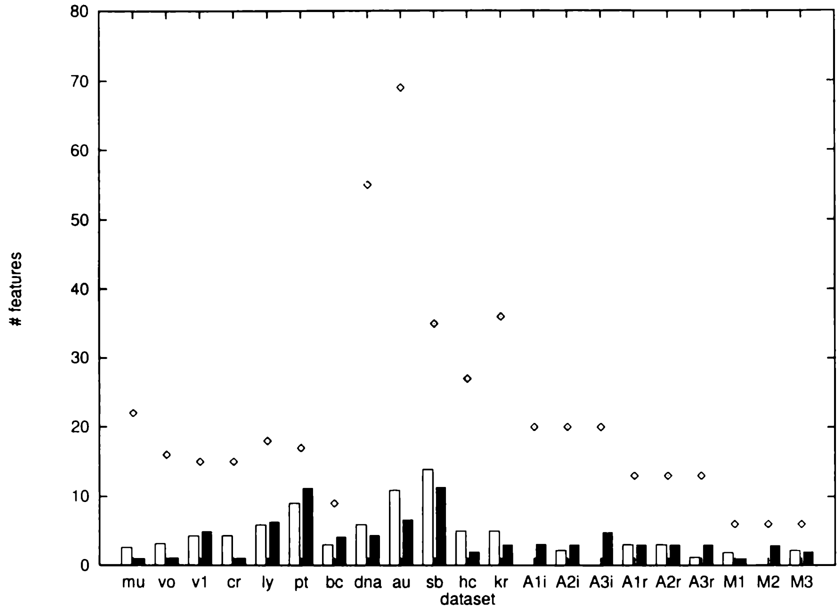


Figure 7.3: Number of features selected by the wrapper using naive Bayes (left) and CFS (right). Dots show the number of features in the original dataset.

Table 7.3 shows the accuracy of C4.5 without feature selection and C4.5 with feature selection by the wrapper and CFS-UC. The wrapper improves C4.5 on six datasets and degrades on seven. CFS improves C4.5 on five datasets and degrades on seven.

The wrapper is more successful than CFS on those artificial datasets that have interacting features. It is able to improve C4.5's performance on A3r and M1, where CFS cannot. It is interesting to note that, while the wrapper does better than CFS on the chess end-game dataset, it still degrades C4.5's performance. One possible explanation for this is that there are high order—perhaps higher than pairwise—feature interactions in this dataset. While the wrapper using a forward best first search stands a good chance of discovering pairwise interactions, backward searches are needed to discover higher than pairwise feature interactions [LS94b].

Figure 7.4 shows the difference in accuracy between CFS and the wrapper. Bars above the zero line show where C4.5's average accuracy using feature subsets selected by CFS is higher than its average accuracy using feature subsets selected by the wrapper. CFS is better than the wrapper for six domains and is worse for eight domains. As was the case with naive Bayes, CFS is superior to the wrapper on those artificial domains with added

Dom	C4.5	wrapper	CFS
mu	99.59 ± 0.4	99.03 ± 0.6−	98.48 ± 0.1−
vo	95.23 ± 1.4	95.33 ± 1.1	95.67 ± 1.0+
v1	88.71 ± 2.0	87.82 ± 2.7−	88.37 ± 2.2
cr	83.69 ± 1.5	84.75 ± 1.3+	85.61 ± 1.0+
ly	75.80 ± 5.4	80.86 ± 9.4+	76.51 ± 5.3
pt	40.99 ± 4.4	39.53 ± 3.4−	41.51 ± 3.5
bc	71.77 ± 3.3	71.11 ± 2.7	70.97 ± 3.2
dna	74.58 ± 6.5	74.21 ± 5.0	77.20 ± 6.3+
au	78.48 ± 3.8	75.50 ± 3.6−	72.56 ± 2.8−
sb	89.16 ± 1.6	89.65 ± 2.2	81.28 ± 2.9−
hc	84.02 ± 3.0	85.56 ± 3.1+	86.05 ± 3.5+
kr	99.16 ± 0.3	97.19 ± 1.2−	90.41 ± 0.7−
A1i	100.00 ± 0.0	87.40 ± 0.0−	100.00 ± 0.0
A2i	100.00 ± 0.0	94.87 ± 10.4−	100.00 ± 0.0
A3i	75.64 ± 9.4	73.00 ± 0.0	74.70 ± 6.5
A1r	100.00 ± 0.0	99.17 ± 3.3	100.00 ± 0.0
A2r	100.00 ± 0.0	99.40 ± 4.2	100.00 ± 0.0
A3r	95.74 ± 5.0	100.00 ± 0.0+	75.16 ± 3.4−
M1	83.87 ± 6.2	91.71 ± 7.0+	75.00 ± 0.0−
M2	66.81 ± 0.8	67.10 ± 0.0+	67.06 ± 0.3+
M3	97.54 ± 1.8	97.71 ± 1.9	96.42 ± 1.6−

+, − statistically significant improvement or degradation

Table 7.3: Comparison between C4.5 without feature selection and C4.5 with feature selection by the wrapper and CFS.

irrelevant features. The wrapper consistently identifies the three interacting features in the A3r domain and the two interacting features in the M1 domain most of the time—resulting in superior performance over CFS on these domains. However, the wrapper still fails on A3i because of the small sample size combined with the presence of many irrelevant features.

CPU times for the wrapper with C4.5 are similar to those for naive Bayes. The soybean dataset took the longest at just over four hours to complete one trial; M2 took the least amount of CPU time at around one and half minutes. As CFS is independent of the learning algorithm, its execution time remains the same.

Figure 7.5 shows how feature selection by the wrapper and CFS affects the size of the trees induced by C4.5. Bars below the zero line indicate that feature selection has *reduced* the size of the trees. The graph shows that both feature selectors reduce the size of the trees induced by C4.5 more often than not. CFS affords similar reductions in tree size as the wrapper. The wrapper was particularly successful on the lymphography domain—

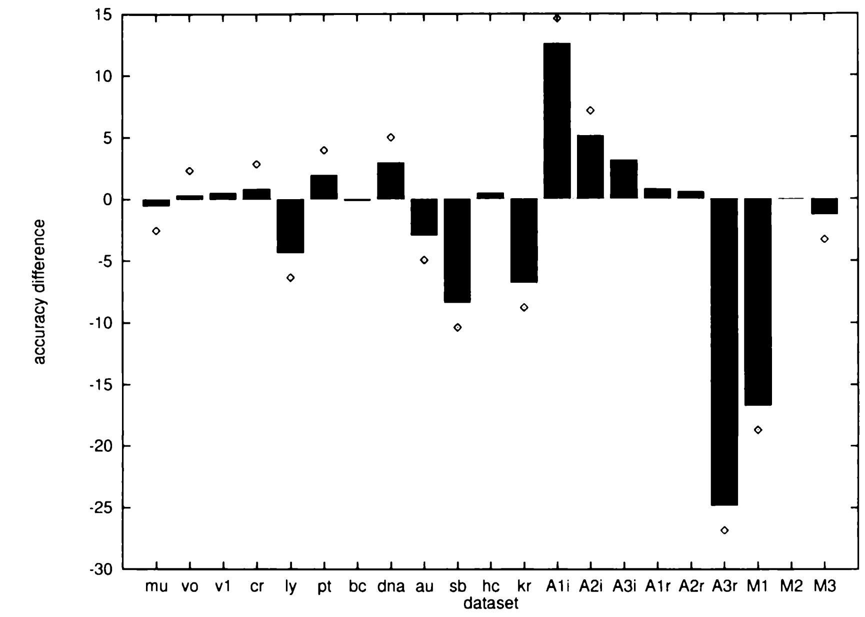


Figure 7.4: Comparing CFS with the wrapper using C4.5: Average accuracy of C4.5 using feature subsets selected by CFS minus the average accuracy of C4.5 using feature subsets selected by the wrapper. Dots show statistically signifcant results

not only did it increase C4.5's accuracy and outperform subsets chosen by CFS, but it also resulted in the smallest average tree size (interestingly, the wrapper was the poorest performer on this dataset for naive Bayes).

The wrapper tends to select slightly smaller feature subsets when used with C4.5 than CFS (CFS's subsets are, of course, the same for C4.5 as they are for naive Bayes).

7.3 Chapter Summary

This chapter compares CFS with the wrapper feature selector. Although CFS and the implementation of the wrapper used herein share the same search strategy, they represent two completely different paradigms for feature selection—wrappers evaluate feature subsets by statistical estimation of their accuracy with respect to a learning algorithm, while CFS (a filter) evaluates feature subsets by a heuristic measure based on correlation. Wrappers are generally considered to be superior to filters as they are tuned to the specific interaction between a learning algorithm and its training data and stand the best chance of

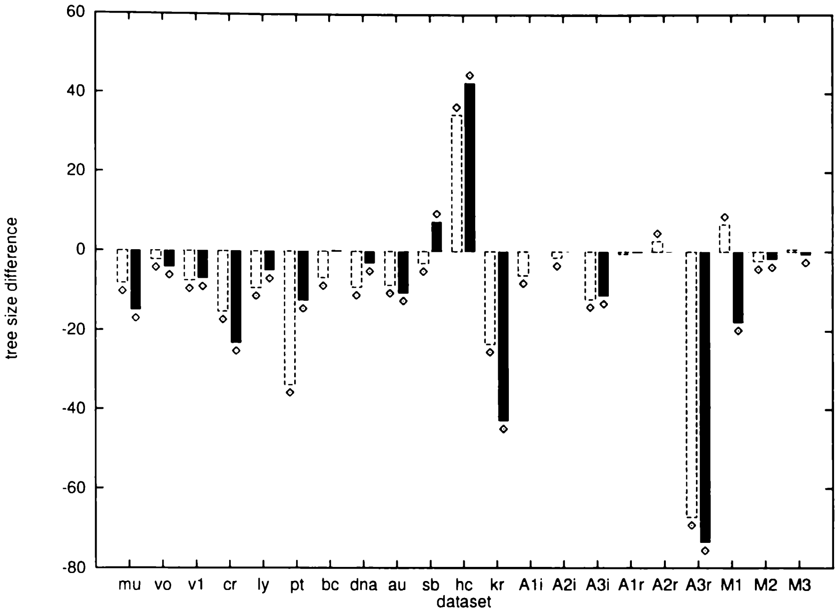


Figure 7.5: Average change in the size of the trees induced by C4.5 when features are selected by the wrapper (left) and CFS (right). Dots show statistically significant results.

finding the "optimal" feature subset.

Experiments in this chapter show CFS to be competitive with wrapper in many cases. The cases for which the wrapper out-performs CFS are generally those for which CFS's assumption of attribute independence given the class is grossly violated, or contain features that are locally predictive in small areas of the instance space (as shown in Chapter 6). Because CFS makes use of all the training data at once, it can give better results than the wrapper on small datasets—especially if there are many irrelevant features present. Results show that both methods select similar sized feature subsets and both reduce C4.5's trees in a similar manner.

Some might argue that using a backward search would give better results for the wrapper. While this could be true, backward searches are *very* slow for wrappers. Kohavi [Koh95b, KJ96] discusses the use of "compound" search space operators that propel the search more quickly toward the relevant features and make backward searches possible—however, his experiments show little improvement over forward searches except for artificial domains with high order feature interactions. Although compound operators do reduce the execution time of the wrapper, CFS is still many times faster.

Chapter 8

Extending CFS: Higher Order Dependencies

The experiments on artificial domains in Chapter 6 showed that CFS is able to detect relevant features under moderate levels of interaction—that is, when the relevant features are *individually* predictive of the class at least some of the time. However, features whose ability to predict the class is always dependent on others will appear irrelevant to CFS because it assumes feature independence given the class.

Detecting high order feature dependencies is difficult because the probabilities in question are apt to be very small. There is also a risk of overfitting because these probabilities may not be reliably represented when data is limited. Furthermore, research in the area of Bayesian networks has shown that inducing an optimal Bayesian classifier is NP-hard—even when each feature (node in the network) is constrained to to be dependent on at most two other features [Sah96]. For these reasons, a limited "pairwise" approach to detecting feature interactions is one method explored in this chapter. As a second approach, the instance based attribute estimation method RELIEF (see Chapter 4) is used as a replacement for the entropy-based feature-class correlation measure used in CFS-UC. RELIEF has the potential (given sufficient data) to detect higher than pairwise feature interactions.

8.1 Related Work

The feature selection method of Koller and Sahami [KS96b] greedily eliminates features one by one so as to least disrupt the original conditional class distribution. Because it is not reliable to estimate high order probability distributions from limited data, their approach assumes that features involved in high order dependencies will also exhibit some pairwise dependency (see Chapter 3 for details).

Like CFS, the naive Bayesian classifier assumes features are independent given the class. This assumption makes it a very efficient and simple learning algorithm. Furthermore, naive Bayes classification performance has been found to be very competitive with more sophisticated learning schemes [DP96]. These qualities have prompted a number of attempts to reduce the "naivety" of the algorithm to further improve performance on domains with class conditional feature dependencies. Four methods (described below) for improving naive Bayes have taken a pairwise approach (for computational reasons) to detecting and incorporating feature dependencies.

Pazzani [Paz95] combines dependence detection and feature selection in a wrapper approach for improving naive Bayes. Forward and backward hill climbing searches are used—at each stage the search considers adding/subtracting a feature or joining a pair of features. In this manner, the algorithm can join more than two features but has to do so in multiple steps. Joining more than two features will not occur unless the first two result in an increase in accuracy.

Instead of joining whole attributes—that is, where every possible combination of values for two attributes is considered jointly—Kononenko [Kon91] argues that allowing just some combinations of attributes' values to be considered jointly (while others remain independent) is more flexible. His semi-naive Bayesian classifier uses an exhaustive search to determine which pairs of attribute values are worth considering jointly (a probabilistic reliability measure is used to screen joined values); again more than two values can be joined by this algorithm, but doing so requires multiple iterations.

KDB [Sah96, KS97] is an algorithm for constructing limited Bayesian networks that allows k-order feature dependencies. The dependencies are specified in the network in a greedy fashion: for each feature X, network arcs are added to the K other features that X is most dependent on, where dependency is measured in a pairwise fashion using a metric of class conditional mutual information $I(X; Y|C)$. KDB requires that all features be binary; best results are had when KDB is used with a mutual information threshold which prevents “spurious” dependencies from being included in the network.

The TAN (Tree Augmented Naive Bayes) algorithm [FG96] also constructs a Bayesian network but restricts each node (feature) in the network to have at most one additional parent (other than the class). This allows an optimal classifier to be found in quadratic time. Like KDB, TAN uses class conditional mutual information to measure dependency between attributes.

8.2 Joining Features

A straightforward and computationally feasible extension to CFS for detecting higher order dependencies is to consider pairs of features. Joining two features X and Y results in a derived attribute with one possible value corresponding to each combination of values of X and Y. For example, if attribute X has values $\{a, b, c\}$ and attribute Y has values $\{y, z\}$ then the joined attribute XY will have values $\{ay, az, by, bz, cy, cz\}$. An algorithm that considers all possible pairwise combinations of features in this manner is quadratic in the original number of features.

Once new attributes are created, corresponding to each possible pairwise combination of features, the feature-class correlations can be calculated in the normal fashion using any of the measures described in Chapter 4. A derived feature is a *candidate* for feature selection if its correlation with the class is higher than both of its constituent features, otherwise it is discarded. After all the derived features have been screened in this fashion, the feature-feature inter-correlations are calculated for the new feature space and feature selection proceeds as in the original algorithm. It is important to note that this extension to CFS does not perform constructive induction—that is, it does not alter the input space for a machine learning algorithm in any fashion other than discarding some number of the original features. If the best feature subset found by CFS contains derived features, then what is passed on to a learning algorithm are the individual features that comprise the derived features.

Initial experiments with this enhancement to CFS (dubbed CFS-P) showed that more derived features became candidates for feature selection when the number of training instances was small, often leading to larger final feature subsets with inferior performance compared with those chosen by standard CFS. Experiments in Chapter 4 showed how correlation measures tended to increase as the number of training examples decreased—more so for attributes with with a greater number of values than those with fewer values. In this situation, probability estimates are less reliable for attributes with more values, and they may appear more useful than would be warranted by the amount of available data. To counter this trend toward overfitting small training sets, a reliability constraint often applied to chi-square tests for independence is used. Equation 8.1 shows a statistic based on the chi-square distribution:

$$\chi^2 = \sum_i \sum_j \frac{(E_{ij} - O_{ij})}{E_{ij}}, \tag{8.1}$$

where O_{ij} is the observed number of training instances from class C_i having the j-th value of the given attribute, and E_{ij} is the expected number of instances if the null hypothesis (of no association between the two attributes) is true:

$$E_{ij} = \frac{n_{.j} n_{i.}}{n}. \tag{8.2}$$

In Equation 8.2, $n_{.j}$ is the number of training instances with the j-th value of the given attribute, $n_{i.}$ is the number of training instances of class C_i, and n is the total number of training instances.

Equation 8.1 is unreliable and becomes over-optimistic in detecting association when expected frequencies are small. It is recommended that the chi-square test not be used if more than 20% of the expected frequencies are less than 5 [Sie56, WW77]. For CFS, a derived feature is screened by subjecting the expected frequencies for each of its values in each of the possible classes to this constraint—if more than 20% are less than 5, then the derived feature does not become a candidate for selection because its association with the class (with respect to a particular correlation measure) is likely to be overestimated.

8.3 Incorporating RELIEF into CFS

One drawback to considering pairs of features is that the feature subset space can be enlarged considerably and, in turn, can take longer to explore. A second approach to detecting feature interactions—one that does not expand the feature subset space (but of course does incur further computational expense)—is to incorporate the RELIEF [KR92, Kon94] algorithm for estimating feature relevance. The feature estimates provided by RELIEF are used to replace the feature-class correlations in CFS-UC; feature-feature inter-correlations are still calculated as normal using the symmetrical uncertainty coefficient. This version of CFS is called CFS-RELIEF (as opposed to CFS-*Relief* which used a context *insensitive* simplification of RELIEF to calculate feature correlations, as described in Chapters 4 and 5).

RELIEF is an instance-based algorithm that imposes a ranking on features by assigning each a weight. The weight for a particular feature reflects its relevance in distinguishing the classes. RELIEF can be used for feature selection in its own right, but, because it does not explicitly select a subset, a relevance threshold must be set (on a domain by domain basis) by which some number of the features can be discarded. Furthermore, RELIEF makes no attempt to deal with redundant features.

One advantage of RELIEF is that it is sensitive to feature interactions and can detect higher than pairwise interactions, given enough data. Kononenko [Kon94] notes than RELIEF assigns a weight to a feature X by approximating the following difference of probabilities

$$\begin{aligned} W_X &= P(\text{different value of } X \mid \text{nearest instance of different class}) \\ &- P(\text{different value of } X \mid \text{nearest instance of same class}) \end{aligned} \tag{8.3}$$

RELIEF incrementally updates the weights for features by repeatedly sampling instances from the training data. Instance-based similarity metrics are used to find the "nearest" instances to the one sampled, and, because these metrics take all the features into account, the weight for a given feature is estimated in context of the other features. The following is the original RELIEF algorithm [KR92], which operates on two-class domains.

RELIEF:

```
set all feature weights $W[X] = 0$
for $i = 1$ to $m$
    randomly select an instance $R$
    find $R$'s nearest hit $H$ (same class) and nearest miss $M$ (different class)
    for each attribute $X$ do
        $W[X] = W[X] -$ diff$(X, R, H)/m +$ diff$(X, R, M)/m$
```

The function diff calculates the difference between the values of attribute X for two instances. For nominal attributes the difference is either 0 (the attribute has the same value in both instances) or 1 (the value of the attribute differs between the two instances). For continuous attributes the difference is the squared arithmetic difference normalized to the interval $[0, 1]$. Diff is also used to calculate the difference between instances when finding nearest hits and misses. The difference between two instances is simply the sum of the attribute differences.

The version of RELIEF incorporated into CFS is an extended version (RELIEF-F) described by Kononenko [Kon94], which generalizes RELIEF to multiple classes and handles noise. To increase the reliability of RELIEF's weight estimation, RELIEF-F finds the k nearest hits and misses for a given instance ($k = 5$ is used here). For multiple class problems, RELIEF-F finds nearest misses from each different class (with respect to the given instance) and averages their contribution for updating $W[X]$. The average is weighted by the prior probability of each class. The version of RELIEF-F used herein runs the outer loop of the RELIEF algorithm over all the available training instances rather than randomly sampling some number m of them. This results in less variation in RELIEF's estimation of feature weights at the cost of increased computation.

8.4 Evaluation

This section evaluates the performance of the two extended forms of CFS (CFS-P and CFS-RELIEF) described above. Of particular interest is performance on those artificial domains with strong feature interactions, that is, A3, M1, and M2. The results of feature selection for IB1 are presented first as this algorithm is sensitive to interacting features and can achieve higher accuracy on the artificial domains (given the correct feature set) than either naive Bayes or C4.5.

Table 8.1 compares the performance of enhanced CFS (CFS-P and CFS-RELIEF) with standard CFS-UC on artificial domains using IB1 as the induction algorithm. The results show that CFS-P consistently identifies all three relevant features in A3i and A3r, selects the three relevant features for the M1 domain, and averages 5.4 out of the 6 relevant features for the M2 domain. Incorporating derived features into CFS has not resulted in worse performance on any of the artificial domains. For the A1–A3 domains with added redundant features, CFS-P selects, on average, less than one redundant feature. Examination of the subsets selected by CFS-P show that it sometimes selects a joined feature that correctly captures one of the dependencies but, when converted back to its constituent features, results in the inclusion of a redundant feature. For example, the feature set $(AB$,BC,$CE)$ correctly captures the pairwise dependencies in the A2r dataset, but resolves to the feature set (A, B, C, E) where feature E is a copy of feature A. The difficulty is that the derived feature AC is equivalent to CE—both represent the same dependency but the algorithm cannot tell that one is more appropriate than the other (only that one of the two is necessary). However, the inclusion of the occasional redundant feature on these domains has no affect on the accuracy of IB1. Although CFS-P selects the correct subsets for domains A1i–A3i, it often accepts more derived features as candidates for selection than are necessary. Chapter 4 showed how symmetrical uncertainty and *relief* correlation favour non-informative attributes with more values over those with few values. A derived attribute comprised of two non-informative attributes therefore appears more predictive to symmetrical uncertainty and *relief* than either of its constituent attributes. This does not happen if the MDL measure is used because it assigns a value less than zero to non-informative attributes (given sufficient data).

The results for CFS-RELIEF show it to be less effective than CFS-P. Although it improves over standard CFS-UC on the same datasets as CFS-P does, it degrades on two datasets, while CFS-P does not degrade on any. Some irrelevant features are included by CFS-RELIEF on the A1i domain, resulting lower accuracy than either CFS-P or standard

	IB1					
Domain	CFS-UC		CFS-P		CFS-RELIEF	
A1i	100.00 ± 0.0	{3.1}	100.00 ± 0.0	{3.1}	94.38 ± 4.9−	{8.1}
A2i	100.00 ± 1.4	{3.0}	100.00 ± 0.0	{3.0}	100.00 ± 0.0	{3.0}
A3i	47.07 ± 16.4	{4.8}	100.00 ± 0.0+	{3.0}	100.00 ± 0.0+	{3.0}
A1r	100.00 ± 0.0	{3.0}	100.00 ± 0.0	{3.9}	100.00 ± 0.0	{3.3}
A2r	100.00 ± 0.0	{3.0}	100.00 ± 0.0	{3.3}	76.09 ± 1.2−	{2.0}
A3r	53.44 ± 17.0	{3.0}	100.00 ± 0.0+	{3.7}	75.96 ± 1.2+	{2.0}
M1	75.00 ± 0.0	{1.0}	99.33 ± 1.2+	{3.0}	98.22 ± 4.8+	{2.9}
M2	66.94 ± 0.6	{2.9}	74.67 ± 5.9+	{5.4}	69.12 ± 4.1+	{4.0}
M3	97.22 ± 0.0	{2.0}	97.22 ± 0.0	{2.0}	97.22 ± 0.0	{2.0}

+, − statistically significant improvement or degradation

Table 8.1: Performance of enhanced CFS (CFS-P and CFS-RELIEF) compared with standard CFS-UC on artificial domains when IB1 is used as the induction algorithm. Figures in braces show the average number of features selected.

CFS-UC. On M1 and M2, CFS-RELIEF selects fewer relevant features on average than CFS-P, resulting in lower accuracy on these domains. It is interesting, given RELIEF's ability to detect high order interactions, that CFS-RELIEF does not do as well as CFS-P—especially on M2 which has the most feature interaction. Kira and Rendell [KR92] note that as the amount of feature interaction increases, the amount of training data must also increase in order for RELIEF to reliably estimate feature relevance. This was tested with CFS-RELIEF by increasing the number of instances in the M2 dataset. After quadrupling the size of the dataset CFS-RELIEF was able to reliably select all 6 relevant features.

CFS-RELIEF is effective on A3i, but fails to select all three relevant features on A3r and A2r. The problem lies with the RELIEF algorithm—examination of the feature relevances assigned by RELIEF on these datasets show that feature A (and its exact copies in each domain) are consistently assigned relevance 0. The selection of nearest neighbours is very important to RELIEF as it attempts to find nearest neighbours with respect to "important" attributes. Averaging the contribution of k nearest neighbours improves RELIEF's attribute estimates by helping counteract the effect of irrelevant attributes, redundant attributes, and noise on nearest neighbour selection. However, this has not helped in the case of A2r and A3r . To see why, consider the weight update for attribute A from the A2 domain $((A \wedge B) \vee (A \wedge C) \vee (B \wedge C))$, given the instance $(A = 0, B = 0, C = 0)$. Furthermore, assume that there are at least two copies of each instance in the dataset (as there would be in any but very small samples). Table 8.2(a) shows all possible instances in this domain, along with their distance from instance $(A = 0, B = 0, C = 0)$. The nearest neighbour of the same class as $(A = 0, B = 0, C = 0)$ is another copy of itself. Since the value of A is the same here for both instances, there is no change to the weight for A. There are three nearest instances—each differing by two feature values—equally close to $(A = 0, B = 0, C = 0)$ from the opposite class: $(A = 0, B = 1, C = 1), (A = 1, B = 0, C = 1)$, and $(A = 1, B = 1, C = 0)$. Out of the three, A's value differs for the latter two, which results in the weight for A being incremented (on average) two out of three times the instance $(A = 0, B = 0, C = 0)$ is sampled from the training data. If, however, there is a fourth feature D (see Table 8.2(b)), which is an exact copy of A, then the nearest neighbour of the opposite class to $(A = 0, B = 0, C = 0, D = 0)$ is $(A = 0, B = 1, C = 1, D = 0)$. This instance has the same value for A, which results in no change to the weight for A. This situation (the nearest instance of the opposite class having the same value for feature A) occurs for every instance in domain A2 when there is a copy of feature A present. The result is that the weight for A (and its copy D) are never changed from the initial value of 0. In this example $k = 1$ has been used for simplicity. Increasing the value of k can help, but, in this case, k would have to be increased in proportion to the number of training instances[1]. Another remedy would be to restrict an instance to appearing only once in the list of nearest neighbours.

Tables 8.3 and 8.4 compare the performance of enhanced CFS (CFS-P and CFS-RELIEF) with standard CFS-UC on the artificial domains when C4.5 and naive Bayes are used as the induction algorithms.

In the case of C4.5, CFS-P improves over standard CFS-UC on A3 and on M1. CFS-P does not result in worse performance on any dataset when compared to CFS-UC. CFS-RELIEF improves over standard CFS on A3i and M1, but does not do as well as CFS-P on A2r and A3r because it does not select all three relevant features.

In the case of naive Bayes, both enhancements to CFS result in some degraded results compared to standard CFS-UC, although the effect is less dramatic for CFS-P. The in-

[1] There are 400 training instances for A2r. Assuming instances are uniformly distributed, k would have to be set in excess of $400 \div 8 = 50$ in order to start detecting the relevance of attribute A.

Inst.	class	A	B	C	Dist. from 1
1	0	0	0	0	0
2	0	0	0	1	1
3	0	0	1	0	1
4	1	0	1	1	2
5	0	1	0	0	1
6	1	1	0	1	2
7	1	1	1	0	2
8	1	1	1	1	3

(a)

Inst.	class	A	B	C	D	Dist. from 1
1	0	0	0	0	0	0
2	0	0	0	1	0	1
3	0	0	1	0	0	1
4	1	0	1	1	0	2
5	0	1	0	0	1	2
6	1	1	0	1	1	3
7	1	1	1	0	1	3
8	1	1	1	1	1	4

(b)

Table 8.2: An example of the effect of a redundant attribute on RELIEF's distance calculation for domain A2. Table (a) shows instances in domain A2 and Table (b) shows instances in domain A2 with an added redundant attribute. The column marked "Dist. from 1" shows how far a particular instance is from instance #1.

clusion of the occasional redundant feature on A1r and A2r is responsible for the slight decrease in the performance of CFS-P on these domains. On M1, the inclusion of the two interacting features by CFS-P and CFS-RELIEF results in slightly worse performace than leaving them out.

	C4.5		
Domain	CFS-UC	CFS-P	CFS-RELIEF
A1i	100.00 ± 0.0	100.00 ± 0.0	99.50 ± 2.5
A2i	100.00 ± 0.0	100.00 ± 0.0	100.00 ± 0.0
A3i	74.70 ± 6.5	100.00 ± 0.0+	100.00 ± 0.0+
A1r	100.00 ± 0.0	100.00 ± 0.0	100.00 ± 0.0
A2r	100.00 ± 0.0	100.00 ± 0.0	75.96 ± 1.4−
A3r	75.16 ± 3.4	100.00 ± 0.0+	75.64 ± 1.1
M1	75.00 ± 0.0	93.66 ± 5.3+	93.32 ± 5.9+
M2	67.06 ± 0.3	67.06 ± 0.3	67.02 ± 0.3
M3	96.42 ± 1.6	96.03 ± 1.7	96.03 ± 1.7

+, − statistically significant improvement or degradation

Table 8.3: Performance of enhanced CFS (CFS-P and CFS-RELIEF) compared to standard CFS-UC on artificial doamins when C4.5 is used as the induction algorithm.

CFS-P and CFS-RELIEF were also trialed on the natural domains. Tables 8.5 through 8.7 compare the performance of CFS-P and CFS-RELIEF with standard CFS-UC on the natural domains when IB1, C4.5 and naive Bayes are used as the induction algorithms.

Subsets selected by CFS-P are practically identical to those selected by standard CFS-UC for all datasets with the exception of mushroom. As a result, there is no significant

	naive Bayes		
Domain	CFS-UC	CFS-P	CFS-RELIEF
A1i	99.49 ± 1.8	99.49 ± 1.8	95.63 ± 3.2−
A2i	100.00 ± 0.0	100.00 ± 0.0	100.00 ± 0.0
A3i	72.04 ± 1.3	73.00 ± 0.0+	73.00 ± 0.0+
A1r	100.00 ± 0.0	99.76 ± 0.5−	100.00 ± 0.0
A2r	100.00 ± 0.0	97.08 ± 7.5−	76.14 ± 1.3−
A3r	72.28 ± 1.6	74.45 ± 0.0+	74.45 ± 0.0+
M1	75.00 ± 0.0	74.22 ± 2.5−	73.00 ± 6.4−
M2	64.93 ± 2.0	64.30 ± 2.0	65.52 ± 1.8
M3	97.17 ± 0.4	97.17 ± 0.4	97.17 ± 0.4

+, − statistically significant improvement or degradation

Table 8.4: Performance of enhanced CFS (CFS-P and CFS-RELIEF) compared to standard CFS-UC on artificial doamins when naive Bayes is used as the induction algorithm.

accuracy difference on any but the mushroom dataset for any of the induction algorithms when using subsets selected by CFS-P. Very few derived features were considered by CFS-P on the natural domains, from which it can be concluded that either there is little pairwise dependency in these datasets, or that, in many cases, there is not enough training data from which to make reliable estimations. The one exception to this was the chess end-game dataset, where over one hundred derived features were considered, on average. However, many of these derived features were clearly irrelevant[2]. The strongest pairwise dependencies were between the three features selected by standard CFS on this dataset. It is possible that these strong dependencies overshadow some that are useful in only a small area of the instance space (the same problem that occured with normal features for standard CFS in chapter 6).

CFS-RELIEF is better than standard CFS and CFS-P on the mushroom domain for IB1 and C4.5 but not for naive Bayes. This suggests that it has detected more feature interaction than CFS-P on this dataset. The result for audiology is better than standard CFS for IB1 and naive Bayes but not for C4.5. Using subsets provided by CFS-RELIEF has resulted in worse performance than standard CFS on three datasets for C4.5 and on four datasets for both naive Bayes and IB1. The datasets for which CFS-RELIEF has degraded accuracy are among those with fewer instances. This suggests that RELIEF's attribute estimation is less reliable for small datasets.

[2] In comparison, CFS-P using the MDL measure considered on average less than 50 derived attributes on the chess end-game dataset.

	IB1		
Domain	CFS-UC	CFS-P	CFS-RELIEF
mu	98.48 ± 0.1	98.64 ± 0.3+	99.72 ± 0.2+
vo	95.60 ± 1.0	95.60 ± 1.0	95.12 ± 1.2−
vl	88.35 ± 2.1	88.35 ± 2.1	88.17 ± 1.9
cr	85.61 ± 1.0	85.61 ± 1.0	85.61 ± 1.0
ly	80.01 ± 4.8	80.01 ± 4.8	79.38 ± 6.0
pt	40.40 ± 2.8	40.40 ± 2.8	39.75 ± 2.8
bc	70.67 ± 3.8	70.89 ± 3.7	69.34 ± 3.5
dna	86.94 ± 4.7	86.94 ± 4.7	85.36 ± 4.9−
au	67.60 ± 3.6	67.60 ± 3.6	69.23 ± 3.5+
sb	84.24 ± 2.6	84.24 ± 2.6	82.07 ± 2.1−
hc	86.89 ± 2.5	86.89 ± 2.5	86.14 ± 3.2−
kr	90.41 ± 0.7	90.41 ± 0.7	90.41 ± 0.7

+, − statistically significant improvement or degradation

Table 8.5: Performance of enhanced CFS (CFS-P and CFS-RELIEF) compared to standard CFS-UC on natural domains when IB1 is used as the induction algorithm.

	C4.5		
Domain	CFS-UC	CFS-P	CFS-RELIEF
mu	98.48 ± 0.1	98.61 ± 0.3+	99.44 ± 0.6+
vo	95.67 ± 1.0	95.67 ± 1.0	95.25 ± 1.3−
vl	88.37 ± 2.2	88.37 ± 2.2	87.90 ± 2.0−
cr	85.61 ± 1.0	85.61 ± 1.0	85.61 ± 1.0
ly	76.51 ± 5.3	76.51 ± 5.3	77.65 ± 5.8
pt	41.51 ± 3.5	41.51 ± 3.5	41.27 ± 3.5
bc	70.97 ± 3.2	71.18 ± 3.2	70.38 ± 2.5
dna	77.20 ± 6.3	77.20 ± 6.3	77.68 ± 6.3
au	72.56 ± 2.8	72.56 ± 2.8	69.91 ± 3.0−
sb	81.28 ± 2.9	81.28 ± 2.9	81.17 ± 2.0
hc	86.05 ± 3.5	86.05 ± 3.5	86.54 ± 3.2
kr	90.41 ± 0.7	90.41 ± 0.7	90.41 ± 0.7

+, − statistically significant improvement or degradation

Table 8.6: Performance of enhanced CFS (CFS-P and CFS-RELIEF) compared to standard CFS-UC on natural domains when C4.5 is used as the induction algorithm.

	naive Bayes		
Domain	CFS-UC	CFS-P	CFS-RELIEF
mu	98.49 ± 0.1	98.60 ± 0.2+	97.71 ± 0.2−
vo	95.60 ± 1.0	95.60 ± 1.0	95.61 ± 1.0
vl	89.04 ± 1.7	89.04 ± 1.7	88.85 ± 1.7
cr	85.60 ± 1.0	85.60 ± 1.0	85.60 ± 1.0
ly	82.16 ± 6.1	82.16 ± 6.1	79.45 ± 5.4−
pt	45.83 ± 3.5	45.83 ± 3.5	45.58 ± 3.7
bc	71.86 ± 3.6	71.84 ± 3.5	70.49 ± 2.9−
dna	90.53 ± 4.5	90.53 ± 4.5	89.74 ± 4.8
au	66.13 ± 3.2	66.13 ± 3.2	69.87 ± 3.0+
sb	87.63 ± 2.5	87.63 ± 2.5	84.82 ± 2.4−
hc	87.35 ± 3.7	87.35 ± 3.7	87.86 ± 3.5
kr	90.40 ± 0.6	90.40 ± 0.6	90.40 ± 0.6

+, − statistically significant improvement or degradation

Table 8.7: Performance of enhanced CFS (CFS-P and CFS-RELIEF) compared with standard CFS-UC on natural doamins when naive Bayes is used as the induction algorithm.

8.5 Discussion

This chapter presents two methods of extending CFS to detect feature interaction: CFS-P considers pairs of features and CFS-RELIEF replaces standard CFS's feature-class correlation with attribute estimates provided by the RELIEF algorithm. From experiments comparing these enhancements to standard CFS the following conclusions can be drawn:

- Both CFS-P and CFS-RELIEF can improve accuracy over standard CFS on domains where there are pairwise feature interactions.
- In general, CFS-P does not degrade accuracy compared to standard CFS.
- CFS-RELIEF does not perform as well as CFS-P. RELIEF's estimates for attributes are less reliable when there are fewer training instances and in some cases are affected by the presence of redundant attributes. Both of these factors have an impact on the feature subsets selected by CFS-RELIEF.

Since considering pairs of features does not degrade the performance of CFS, CFS-P is preferred over CFS-RELIEF if it is suspected that a dataset contains feature interactions. For larger datasets, the MDL measure is the preferred correlation measure to use with CFS-P, because its ability to clearly identify non-informative attributes will result in fewer "spurious" derived features being considered for selection.

It should be noted, however, that CFS-P will not detect interaction of a higher order than pairwise. RELIEF, on the other hand, has been shown (on parity concepts) to be able to detect higher than pairwise interactions, given sufficient training data [KR92].

Both CFS-P and CFS-RELIEF are more computationally expensive than standard CFS. In the worst case CFS-P may square the number features under consideration if every pairwise combination of original features is accepted as a candidate for selection (this is unlikely to happen in practice). Because it finds the nearest neighbours of each training instance, the version of RELIEF used in CFS-RELIEF is quadratic in the number of training instances. However, both enhanced versions of CFS are still much faster than the wrapper. For example, a single trial on the mushroom dataset took 7 units of CPU time for CFS-P (31 derived features were considered as candidates) and 33 units of CPU time for CFS-RELIEF; the same trial took 2154 units of cpu time for the wrapper.

Chapter 9
Conclusions

9.1 Summary

The central claim of this thesis is that feature selection for supervised machine learning can be accomplished on the basis of correlation between features. A feature selection algorithm has been implemented and empirically tested to support this claim.

Chapter 4 outlined the rationale for a correlation-based approach to feature selection, with ideas and an evaluation formula adapted from test theory. The evaluation formula awards high merit to feature subsets that contain features predictive of the class (measured by the average of the correlations between the individual features and the class), and a low level of redundancy (as measured by the average inter-correlation between features). An implementation of a correlation-based feature selection algorithm (CFS) incorporating this evaluation function was described. Three methods of measuring association between nominal features were reviewed as candidates for the feature correlations required in the evaluation function. Experiments on artificial data showed that all three measures prefer predictive features with fewer values—a bias that is compatible with that of decision tree algorithms such as C4.5 that prefer smaller trees over larger ones. Two of the measures (*relief* and symmetrical uncertainty) give optimistic estimates and may over-estimate multi-valued attributes when data is limited. The MDL measure gives pessimistic estimates when data is limited—a situation that may result in a preference for smaller feature subsets when used in the evaluation function.

CFS was empirically tested using artificial and natural machine learning datasets. Experiments on artificial datasets showed that CFS can effectively screen irrelevant, redundant, and noisy features. CFS selects relevant features as long as they do not strongly interact with other features. Of the three correlation measures reviewed in Chapter 4, symmetrical uncertainty and the MDL measure were superior to *relief*, when used in CFS. In cases where attributes divide training data into pure subsets, *relief* was shown to be more sensitive to the size of a pure subset than either symmetrical uncertainty or MDL—a situation that can lead to the underestimation and omission of relevant features. Experiments with common machine learning algorithms on natural domains showed that, in many cases, CFS improves performance and reduces the size of induced knowledge structures. Again, the symmetrical uncertainty and MDL correlation measures were found to give better results than *relief*. Symmetrical uncertainty was chosen as the preferred correlation measure for CFS because it gave slightly better results, on small datasets, than the more cautious MDL. Results on several datasets showed that CFS is sometimes overly aggressive in feature selection. In particular, CFS may fail to select features that are locally predictive in small areas of the instance space—especially if they are overshadowed by other strong, globally predictive features. A method of merging top ranked feature subsets partially mitigates this problem; the method is not completely satisfactory, however, because it allows redundant features to be re-included in the final feature set.

Further tests compared CFS with a wrapper approach to feature selection. In many cases, CFS gives results comparable to the wrapper, and generally outperforms the wrapper on small datasets. Datasets on which the wrapper clearly outperforms CFS are those that contain strong feature interactions or have features that are locally predictive for small numbers of instances. CFS is faster than the wrapper—often by more than 2 orders of magnitude.

Chapter 8 investigated two methods of extending CFS to detect feature interaction. Both improved results on some datasets. The first method (CFS-P), which considers pairwise combinations of features, gives more reliable results than the second method (CFS-RELIEF), which uses weights estimated by the RELIEF algorithm as correlations between features and classes. However, CFS-RELIEF has the potential (given enough data) to detect higher than pairwise feature interactions.

9.2 Conclusions

No single learning algorithm is superior to all others for all problems. Research in machine learning attempts to provide insight into the strengths and limitations of different algorithms. Armed with such insight, and background knowledge for a particular problem, practitioners can choose which algorithms to apply. Such is the case with CFS—in many cases CFS can enhance (or not degrade) the performance of machine learning algorithms, while at the same time achieving a reduction in the number of features used in learning. CFS may fail to select relevant features, however, when data contains strongly interacting features or features with values predictive of a small area of the instance space.

CFS is a component of the WEKA workbench [HDW94], which itself is part of ongoing research at the University of Waikato to produce a high quality process model for machine learning. CFS has been applied to a number of problems, most notably to select features for a musical compression system [BI98].

9.3 Future Work

The greatest limitation of CFS is its failure to select features that have locally predictive values when they are overshadowed by strong, globally predictive features. While a single feature such as this may account for only a very small proportion of a dataset, a number of such features may cumulatively cover a significant proportion of the dataset. Merging feature subsets allows redundancy to be re-introduced. While redundancy is less likely to affect algorithms such as C4.5 and IB1, it can have a detrimental effect on naive Bayes. An ideal solution (with naive Bayes in mind) would identify those attributes that are both locally predictive of instances not covered by already selected attributes, and have low correlation with already selected attributes. Of course, attributes such as these (locally predictive and low correlation with others) are likely to have some *irrelevant* values—a number of such features are likely to degrade the performance of instance based learners. Domingos [Dom97] addresses the problem (specifically for instance based learners) by using a wrapper to select a different feature set for each instance. For CFS (and global filters in general) it is a case of being able to "please some of the people some of the time, but not all of the people all of the time".

It would be interesting to apply a "boosting" technique to the problem of detecting locally predictive features. Boosting methods [FS96, SFBL97, Bre96a] improve classification performance by combining the predictions of knowledge induced from multiple runs of a learning algorithm. In each iteration, a learning algorithm is focused on those areas of the training instance space that the learner from the previous iteration found difficult to predict. Such an approach necessitates the use of a particular learning algorithm and, when combined with CFS, would result in a hybrid system (wrapper + filter). To begin with, standard CFS would select an initial set of features. A learning algorithm (using the selected features) could then be applied to predict and hence weight the training instances. CFS would then be applied to the weighted training instances to select a secondary set of features, and so forth. Any locally predictive features that are genuinely useful will help in predicting instances that the learner from the previous iteration had difficulty with.

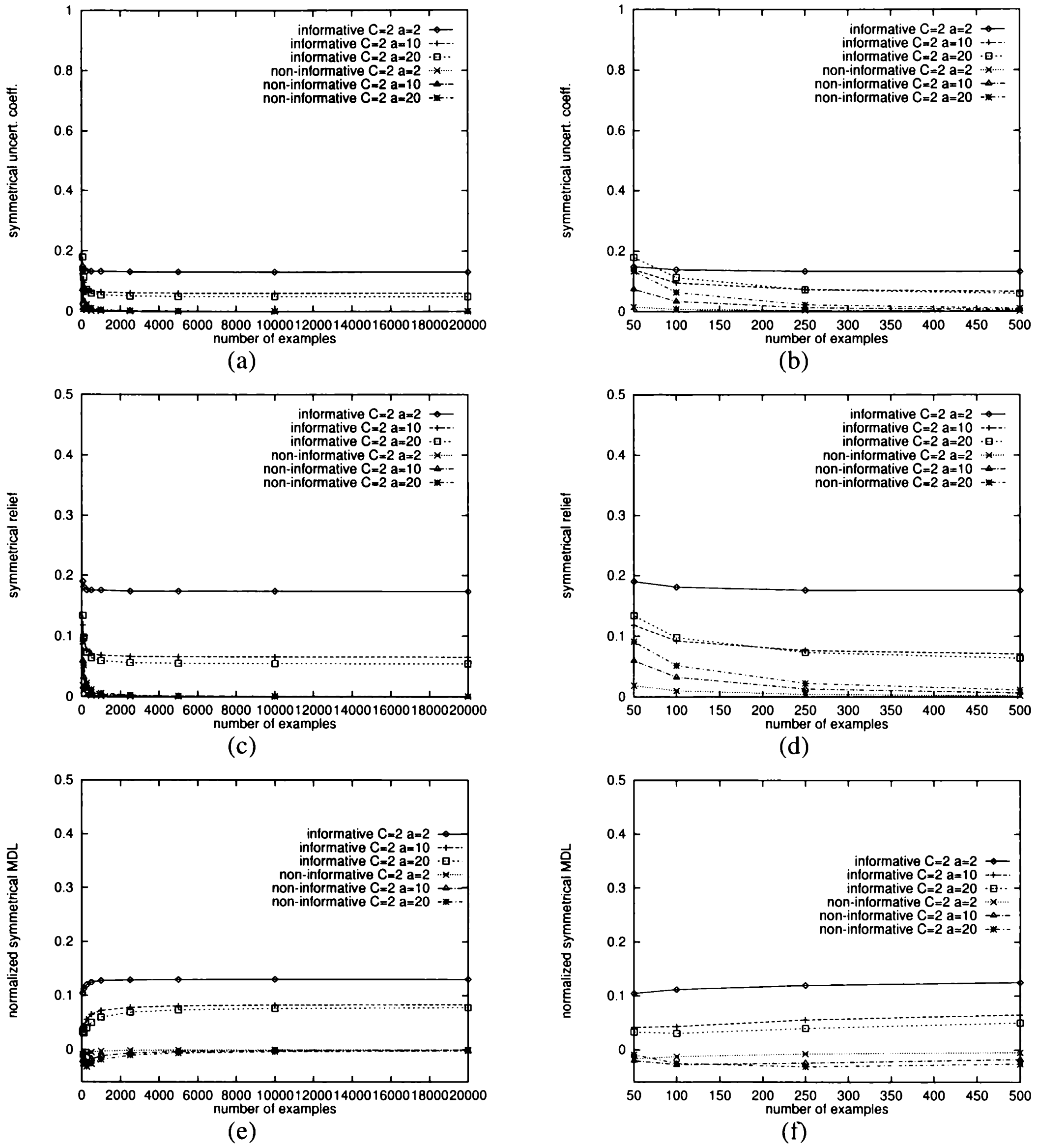
Features selected by CFS generally represent a good "core" subset of features. It would be interesting to see how a wrapper feature selector would fare when started using a feature subset selected by CFS. In this case, a bidirectional search that considers both additions and deletions of features would be more appropriate for the wrapper than either a forward or backward search. Since the search would be initiated from an intelligent start point, the computational expense of the wrapper should be reduced because fewer subsets would be evaluated. This approach may also improve the wrapper's performance on smaller datasets where less reliable accuracy estimates cause it to become trapped in local maxima.

Another area for future work is in trying (or developing) other measures of correlation for use with CFS. Measures of correlation that operate on nominal variables were explored in this thesis. The justification for this was that (a) it is desirable to treat different types of features in a uniform manner in order to provide a common basis for computing correlation, and (b) discretization has been shown to improve (or at least not significantly degrade) the performance of learning algorithms [DKS95]. Ting [Tin95] describes a method of converting nominal attributes to numeric attributes—the opposite of discretization. The method replaces each nominal value of an attribute with its estimated prior probability from the training data. When all attributes (including the class) are numeric, Pearson's linear correlation can be used with CFS. Future experiments will evaluate CFS on domains where all attributes are numeric with learning algorithms such as K* [CT95] and M5′ [WW97] that are capable of predicting continuous class variables.

Appendix A

Graphs for Chapter 4

Figure A.1 shows the behaviour of symmetrical uncertainty, *relief*, and MDL as the number of training instances vary when there are 2 classes.



(a) (b) (c) (d) (e) (f)

Figure A.1: The effect of varying the training set size on symmetrical uncertainty (a & b), symmetrical *relief* (c & d), and normalized symmetrical MDL (e & f) when attributes are informative and non-informative. The number of classes is 2; curves are shown for 2, 10, and 20 valued attributes.

Appendix B

Curves for Concept A3 with Added Redundant Attributes

Figures B.1 through B.4 show curves for CFS-UC, CFS-MDL, and CFS-Relief on concept A3 with added redundant attributes.

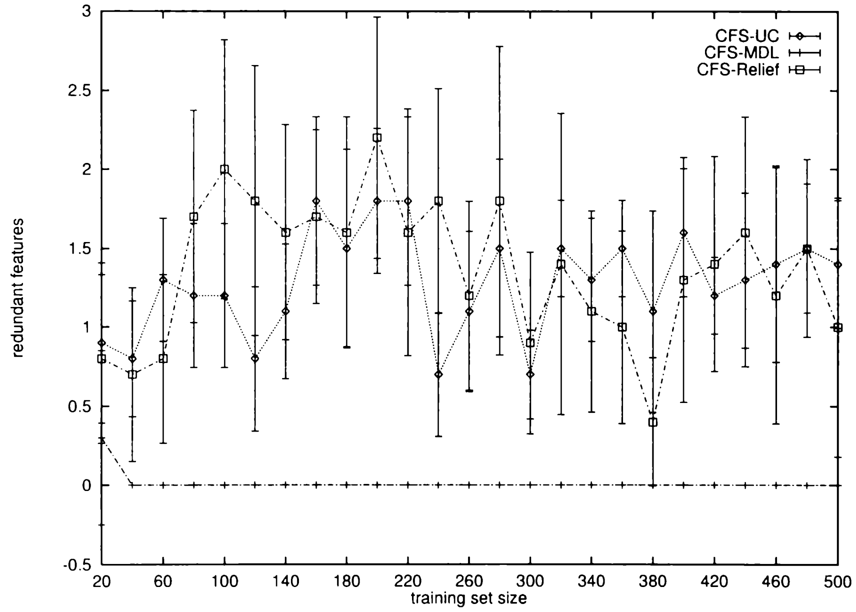


Figure B.1: Number of redundant attributes selected on concept A3 by CFS-UC, CFS-MDL, and CFS-Relief as a function of training set size.

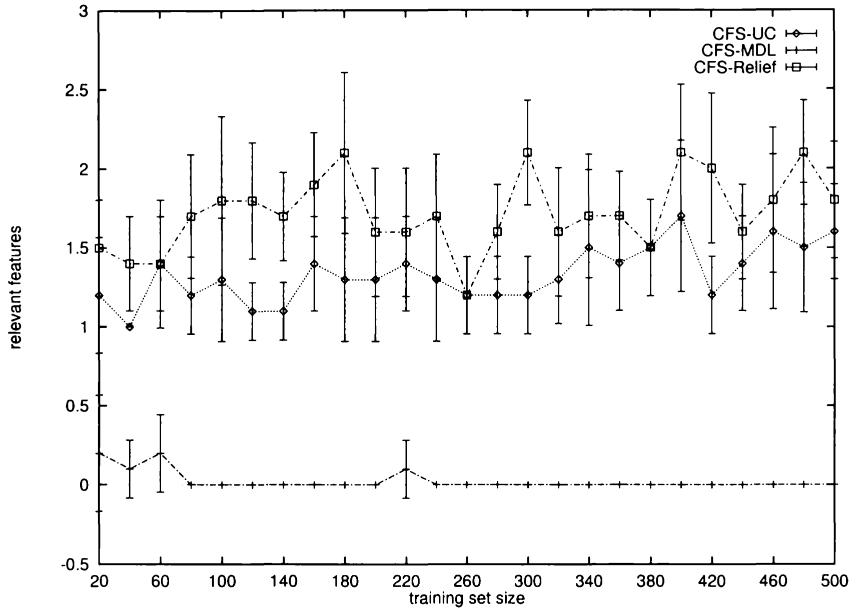


Figure B.2: Number of relevant attributes selected on concept A3 by CFS-UC, CFS-MDL, and CFS-Relief as a function of training set size.

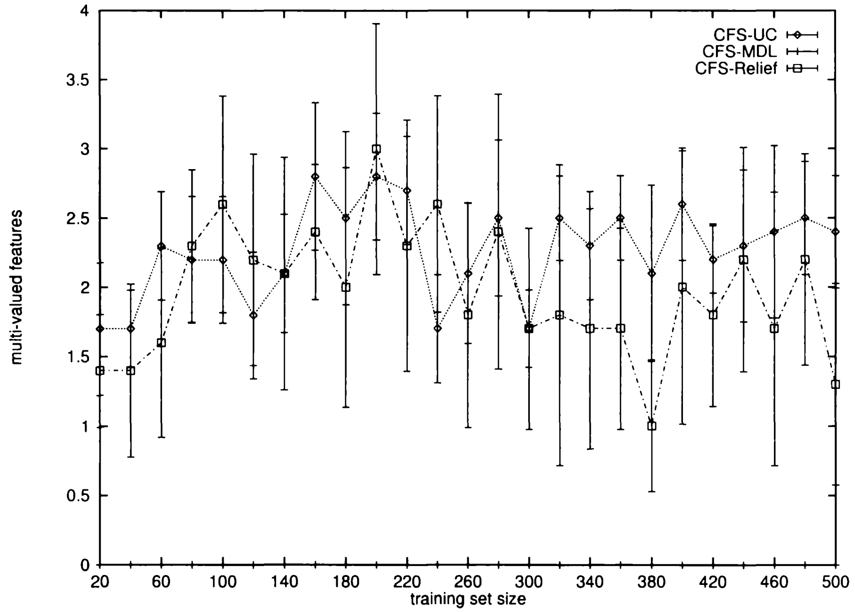


Figure B.3: Number of multi-valued attributes selected on concept A3 by CFS-UC, CFS-MDL, and CFS-Relief as a function of training set size.

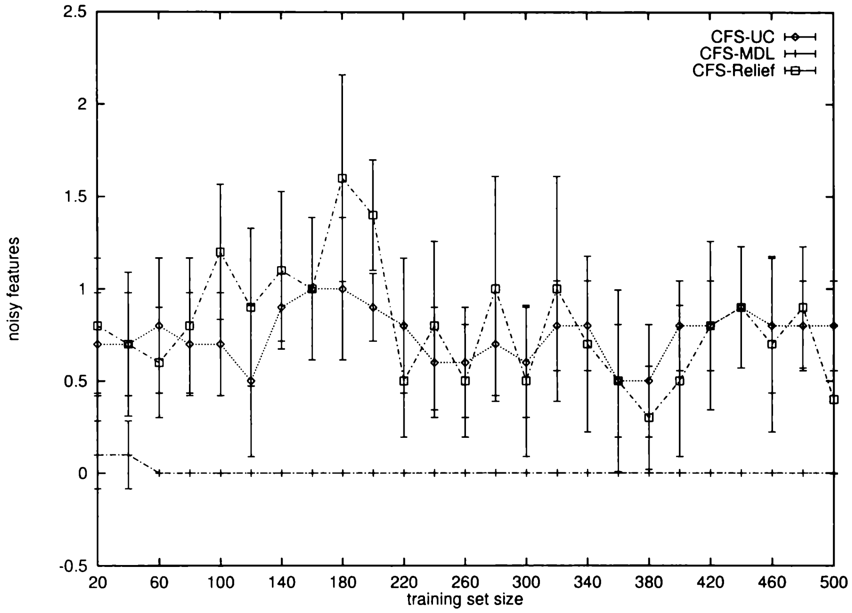


Figure B.4: Number of noisy attributes selected on concept A3 by CFS-UC, CFS-MDL, and CFS-Relief as a function of training set size.

Appendix C

Results for CFS-UC, CFS-MDL, and CFS-Relief on 12 Natural Domains

Domain	CFS-UC-nbayes	CFS-MDL-nbayes	CFS-Relief-nbayes
mu	98.49 ± 0.1	98.49 ± 0.1	98.45 ± 0.9
vo	95.60 ± 1.0	95.32 ± 1.3	95.55 ± 1.2
v1	89.04 ± 1.7	88.94 ± 1.7	89.49 ± 1.9+
cr	85.60 ± 1.0	85.60 ± 1.0	85.60 ± 1.0
ly	82.16 ± 6.1	77.45 ± 5.0−	77.22 ± 5.5−
pt	45.83 ± 3.5	40.61 ± 4.8−	38.59 ± 3.7−
bc	71.86 ± 3.6	73.08 ± 3.6+	72.02 ± 3.8
dna	90.53 ± 4.5	90.68 ± 3.9	88.84 ± 6.1−
au	66.13 ± 3.2	65.36 ± 1.9	72.40 ± 3.9+
sb	87.63 ± 2.5	89.26 ± 1.7+	88.46 ± 3.4
hc	87.35 ± 3.7	88.25 ± 2.6	81.62 ± 3.0−
kr	90.40 ± 0.6	90.13 ± 2.6	90.40 ± 0.6

+, − statistically significant result—better or worse

Table C.1: Accuracy of naive Bayes with feature selection by CFS-UC compared with feature selection by CFS-MDL and CFS-Relief.

Domain	CFS-UC-IB1	CFS-MDL-IB1	CFS-Relief-IB1
mu	98.48 ± 0.1	98.48 ± 0.1	99.06 ± 0.4+
vo	95.60 ± 1.0	95.49 ± 1.3	95.67 ± 1.0
v1	88.35 ± 2.1	88.38 ± 2.1	88.75 ± 1.8
cr	85.61 ± 1.0	85.61 ± 1.0	85.61 ± 1.0
ly	80.01 ± 4.8	77.18 ± 5.0−	76.51 ± 6.8−
pt	40.40 ± 2.8	36.37 ± 4.7−	37.53 ± 3.6−
bc	70.67 ± 3.8	73.69 ± 3.5+	71.73 ± 3.3+
dna	86.94 ± 4.7	86.47 ± 4.8	84.99 ± 5.9−
au	67.60 ± 3.6	65.55 ± 2.6−	72.13 ± 4.4+
sb	84.24 ± 2.6	87.27 ± 1.6+	85.15 ± 2.5+
hc	86.89 ± 2.5	86.34 ± 2.5−	81.62 ± 3.0−
kr	90.41 ± 0.7	90.13 ± 2.6	90.41 ± 0.7

+, − statistically significant result—better or worse

Table C.2: Accuracy of IB1 with feature selection by CFS-UC compared with feature selection by CFS-MDL and CFS-Relief.

Domain	CFS-UC-C4.5	CFS-MDL-C4.5	CFS-Relief-C4.5
mu	98.48 ± 0.1	98.48 ± 0.1	98.84 ± 0.4+
vo	95.67 ± 1.0	95.67 ± 1.0	95.67 ± 1.0
v1	88.37 ± 2.2	88.20 ± 2.2	88.56 ± 1.9
cr	85.61 ± 1.0	85.61 ± 1.0	85.61 ± 1.0
ly	76.51 ± 5.3	75.21 ± 5.6−	75.34 ± 6.3
pt	41.51 ± 3.5	36.82 ± 4.7−	38.85 ± 3.9−
bc	70.97 ± 3.2	73.39 ± 3.3+	70.97 ± 3.3
dna	77.20 ± 6.3	77.10 ± 6.4	77.41 ± 6.5
au	72.56 ± 2.8	65.36 ± 1.9−	69.47 ± 3.6−
sb	81.28 ± 2.9	86.11 ± 1.7+	84.03 ± 3.0+
hc	86.05 ± 3.5	87.16 ± 2.8+	81.62 ± 3.0−
kr	90.41 ± 0.7	90.13 ± 2.6	90.41 ± 0.7

+, − statistically significant result—better or worse

Table C.3: Accuracy of C4.5 with feature selection by CFS-UC compared with feature selection by CFS-MDL and CFS-Relief.

Appendix D

5×2cv Paired *t* test Results

Dietterich [Die88] has shown that the common approach of using a paired-differences t-test based on random subsampling has an elevated chance of Type I error—that is incorrectly detecting a difference when no difference exists. He recommends using the "5×2cv" test instead, although warns that this test has an increased chance of type II error—that is failing to detect a difference when one actually does exist. This test, based on 5 iterations of 2-fold cross validation, uses a modified *t*-statistic to overcome the Type I problem. The 5×2cv *t*-statistic is

$$\tilde{t} = \frac{p_1^{(1)}}{\sqrt{\frac{1}{5}\sum_{i=1}^{5} s_i^2}} \tag{D.1}$$

where $p_1^{(1)}$ is the difference in accuracy from the first fold of the first replication of 2-fold cross validation and s_i^2 is the variance computed from the i-th replication.

Table D.1 shows results for naive Bayes, IB1 and C4.5 before and after feature selection by CFS-UC. The 5×2cv test has been applied. These results are similar in pattern to those presented in Chapter 6 in that CFS improves the performance of naive Bayes for more datasets than it does for either IB1 or C4.5. It can be seen that there are fewer significant results than before indicating that CFS safely removes attributes with out adversely affecting the accuracy of learning algorithms.

Table D.2 shows the accuracy of naive Bayes without feature selection and naive Bayes with feature selection by the wrapper and CFS-UC on all the domains. The 5×2cv test has been applied. Similarly, Table D.3 shows the accuracy of C4.5 with and without feature

Dom	naive Bayes	CFS-nbayes	IB1	CFS-IB1	C4.5	CFS-C4.5
mu	96.17 ± 0.1	98.52 ± 0.0+	100.0 ± 0.0	98.52 ± 0.0−	100.0 ± 0.0	98.52 ± 0.0−
vo	90.26 ± 0.28	95.13 ± 1.01	92.45 ± 0.5	95.64 ± 0.0	95.17 ± 0.8	95.64 ± 0.0
v1	87.22 ± 0.37	88.42 ± 0.91	89.19 ± 1.2	88.60 ± 0.8	89.20 ± 0.9	88.92 ± 0.9
cr	78.00 ± 0.53	85.51 ± 0.00+	81.71 ± 0.5	85.50 ± 0.0	84.16 ± 1.0	85.50 ± 0.0
ly	82.18 ± 0.81	77.96 ± 1.95−	77.05 ± 4.1	77.85 ± 2.7	74.60 ± 3.5	73.92 ± 2.0
pt	45.62 ± 1.56	44.44 ± 1.10	37.20 ± 1.6	39.46 ± 0.7	39.04 ± 2.1	40.26 ± 1.5
bc	71.04 ± 1.04	71.18 ± 1.44	71.16 ± 1.9	72.28 ± 1.2	71.17 ± 1.3	70.75 ± 2.0
dna	86.27 ± 1.73	89.30 ± 2.83+	78.30 ± 2.3	87.54 ± 2.0	74.14 ± 3.1	74.52 ± 2.8+
au	77.11 ± 2.70	66.36 ± 1.97	71.24 ± 4.9	66.05 ± 1.4	74.90 ± 1.5	70.54 ± 0.8
sb	90.90 ± 0.38	87.74 ± 0.41	89.97 ± 0.8	83.36 ± 1.5	87.18 ± 0.5	79.61 ± 1.8
hc	81.74 ± 1.04	87.72 ± 0.65	80.45 ± 0.9	85.87 ± 1.8	81.84 ± 1.2	81.5 ± 0.0
kr	87.05 ± 0.22	88.80 ± 3.26	93.08 ± 0.4	88.80 ± 3.3	98.87 ± 0.1	88.80 ± 3.3

+, − statistically significant improvement or degradation

Table D.1: Naive Bayes, IB1, and C4.5 with and without feature selection on 12 natural domains. A 5×2cv test for significance has been applied.

selection by the wrapper and CFS-UC. These results follow a pattern similar to those presented in Chapter 7 in that CFS does a better job for naive Bayes than the wrapper does, and that the wrapper does a better job for C4.5 than CFS does. For C4.5, three of the four datasets that CFS degrades accuracy on have strong attribute interactions; worse results than the wrapper are to be expected in such cases.

Dom	naive Bayes	wrapper	CFS
mu	96.17 ± 0.1	99.68 ± 0.1+	98.52 ± 0.0+
vo	90.26 ± 0.3	94.80 ± 0.8	95.13 ± 1.0
v1	87.22 ± 0.4	89.89 ± 1.2+	88.42 ± 0.9
cr	78.00 ± 0.53	85.51 ± 0.4+	85.51 ± 0.0+
ly	82.18 ± 0.81	78.37 ± 1.0−	77.96 ± 2.0−
pt	45.62 ± 1.6	41.29 ± 1.1−	44.44 ± 1.1
bc	71.04 ± 1.0	71.61 ± 2.1	71.18 ± 1.4
dna	86.27 ± 1.7	81.40 ± 0.7	89.30 ± 2.8+
au	77.11 ± 2.7	74.06 ± 2.3	66.36 ± 2.0
sb	90.90 ± 0.4	92.18 ± 0.7	87.74 ± 0.4
hc	81.74 ± 1.0	85.60 ± 1.4	87.72 ± 0.7
kr	87.05 ± 0.2	94.34 ± 0.0+	88.80 ± 3.3
A1i	96.24 ± 0.5	87.40 ± 0.0−	100.0 ± 0.8+
A2i	98.34 ± 1.2	92.00 ± 10.9−	100.0 ± 0.0+
A3i	71.38 ± 0.3	72.78 ± 0.4	71.74 ± 0.6
A1r	84.79 ± 0.7	99.94 ± 0.1+	100.0 ± 0.0+
A2r	73.90 ± 0.0	100.0 ± 0.0+	100.0 ± 0.0+
A3r	64.60 ± 4.3	75.64 ± 1.3	72.84 ± 0.8
M1	72.55 ± 2.0	74.31 ± 1.4+	75.00 ± 0.0+
M2	62.64 ± 2.7	67.13 ± 0.0	65.14 ± 1.8
M3	97.23 ± 0.6	97.23 ± 0.6	97.23 ± 0.6

+, − statistically significant improvement or degradation

Table D.2: Comparison between naive Bayes without feature selection and naive Bayes with feature selection by the wrapper and CFS. A 5×2cv test for significance has been applied.

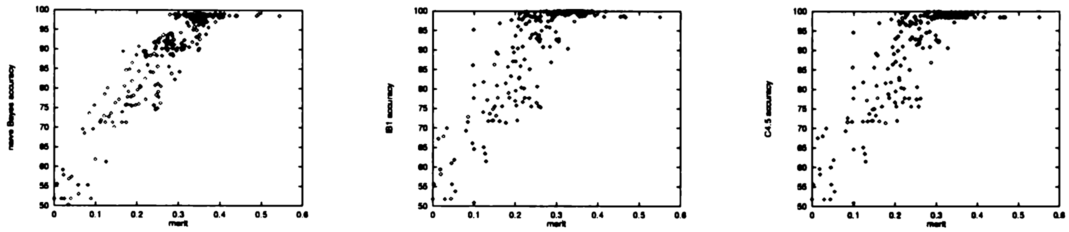
Dom	C4.5	wrapper	CFS
mu	100.0 ± 0.0	99.91 ± 0.1	98.52 ± 0.0−
vo	95.17 ± 0.8	95.72 ± 0.3	95.64 ± 0.0
v1	88.20 ± 0.9	88.38 ± 0.8	88.92 ± 0.9
cr	84.16 ± 1.0	84.50 ± 0.7	85.5 ± 0.0
ly	74.60 ± 3.5	74.86 ± 1.9	73.92 ± 2.0
pt	39.04 ± 2.1	39.51 ± 1.8	40.26 ± 1.5
bc	71.17 ± 1.3	71.74 ± 1.9	70.75 ± 2.0
dna	74.14 ± 3.1	75.05 ± 1.8	74.52 ± 2.8+
au	74.90 ± 1.5	70.96 ± 1.1	70.54 ± 0.8
sb	87.18 ± 0.5	87.38 ± 2.0	79.61 ± 1.8
hc	81.84 ± 1.2	83.38 ± 1.5	81.53 ± 0.0
kr	98.87 ± 0.14	97.21 ± 0.6−	88.80 ± 3.3
A1i	100.0 ± 0.0	100.0 ± 0.0	100.00 ± 0.0
A2i	100.0 ± 0.0	92.01 ± 10.9−	100.00 ± 0.0
A3i	83.90 ± 5.2	73.00 ± 0.0	78.58 ± 6.5
A1r	100.0 ± 0.0	100.0 ± 0.0	100.00 ± 0.0
A2r	100.0 ± 0.0	100.0 ± 0.0	100.00 ± 0.0
A3r	97.58 ± 2.2	100.00 ± 0.0	78.86 ± 5.4−
M1	91.11 ± 5.7	100.0 ± 0.0+	75.00 ± 0.0−
M2	67.10 ± 0.0	67.10 ± 0.0	67.10 ± 0.0
M3	98.67 ± 1.8	98.67 ± 1.7	96.73 ± 1.0−

+, − statistically significant improvement or degradation

Table D.3: Comparison between C4.5 without feature selection and C4.5 with feature selection by the wrapper and CFS. A 5×2cv test for significance has been applied.

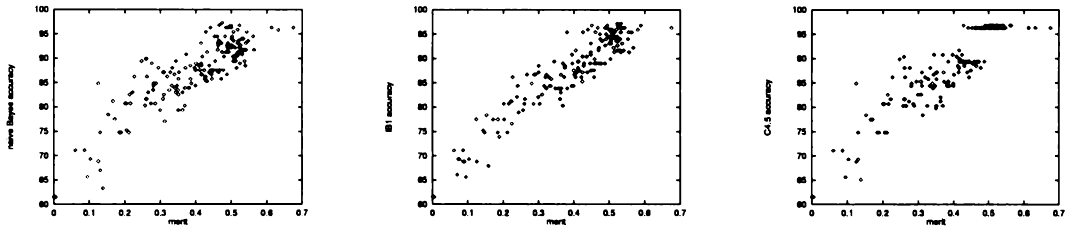
Appendix E

CFS Merit Versus Accuracy



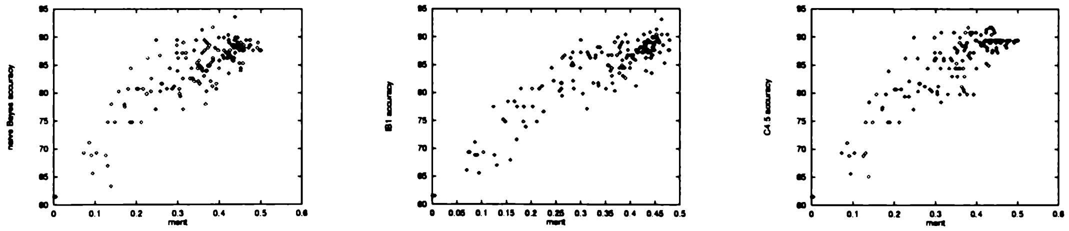
(a) naive Bayes (b) IB1 (c) C4.5

Figure E.1: Mushroom (mu).



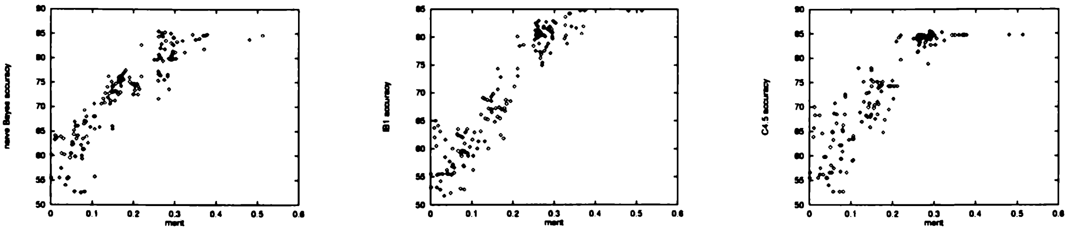
(a) naive Bayes (b) IB1 (c) C4.5

Figure E.2: Vote (vo).



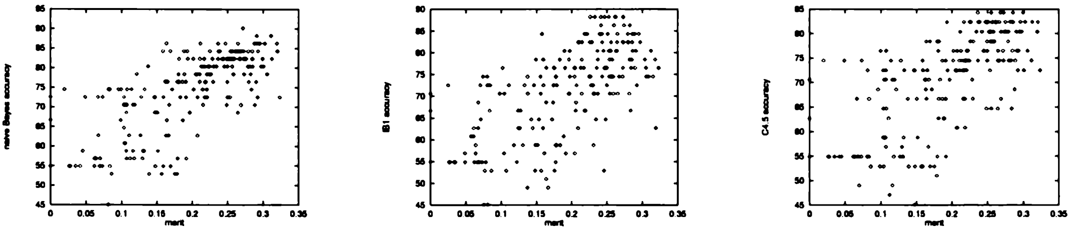
(a) naive Bayes (b) IB1 (c) C4.5

Figure E.3: Vote1 (v1).



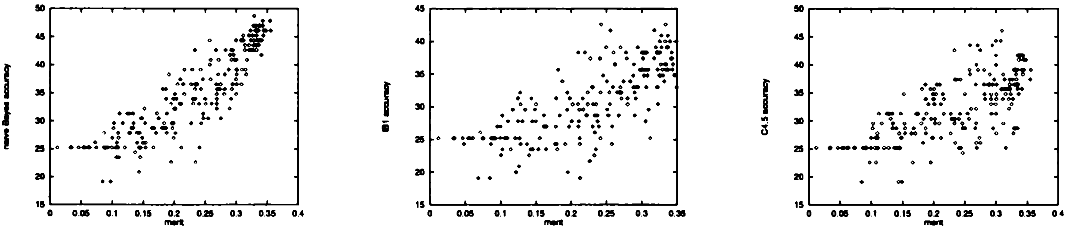
(a) naive Bayes (b) IB1 (c) C4.5

Figure E.4: Australian credit screening (cr).



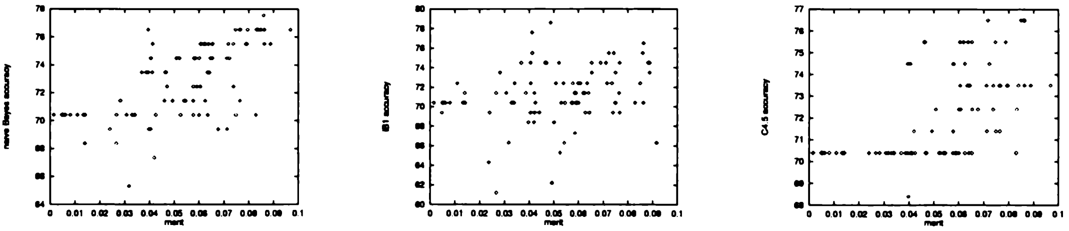
(a) naive Bayes (b) IB1 (c) C4.5

Figure E.5: Lymphography (ly).



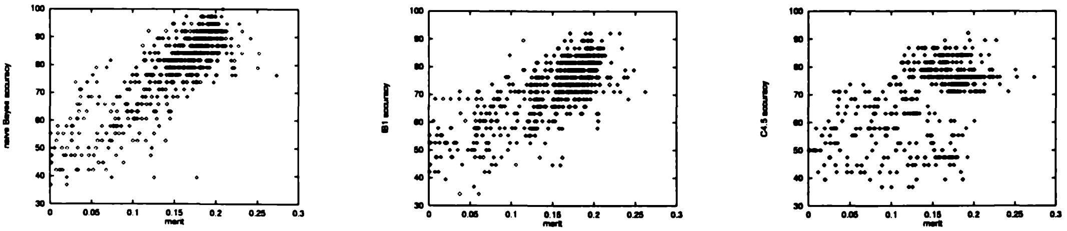
(a) naive Bayes (b) IB1 (c) C4.5

Figure E.6: Primary tumour (pt).



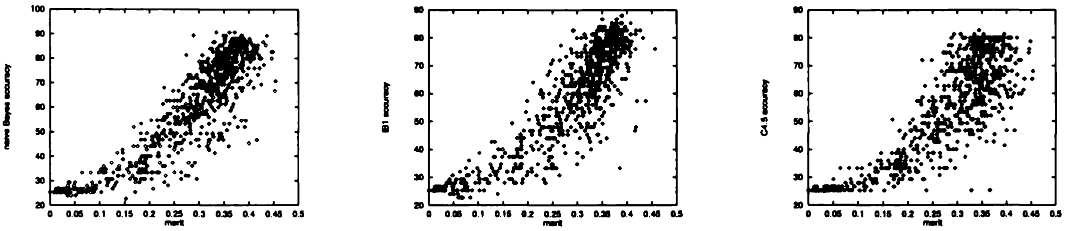
(a) naive Bayes (b) IB1 (c) C4.5

Figure E.7: Breast cancer (bc).



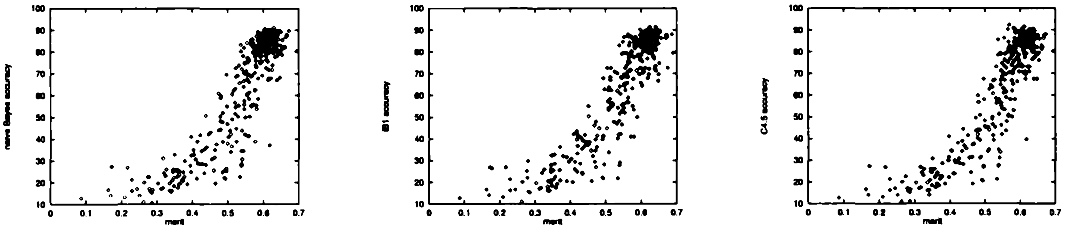
(a) naive Bayes (b) IB1 (c) C4.5

Figure E.8: Dna-promoter (dna).



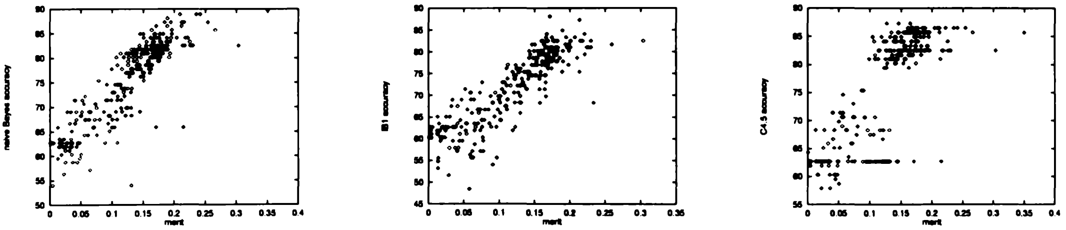
(a) naive Bayes (b) IB1 (c) C4.5

Figure E.9: Audiology (au).



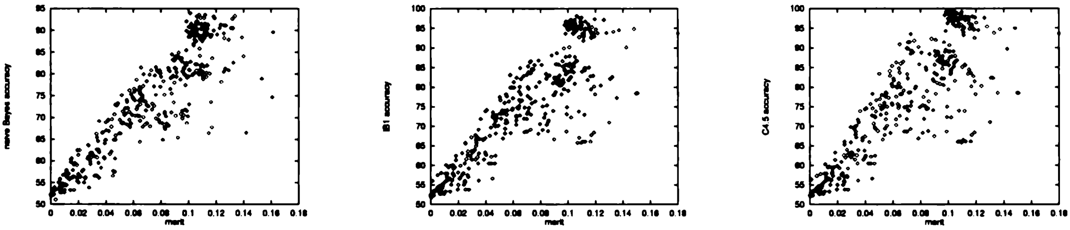
(a) naive Bayes (b) IB1 (c) C4.5

Figure E.10: Soybean-large (sb).



(a) naive Bayes (b) IB1 (c) C4.5

Figure E.11: Horse colic (hc).



(a) naive Bayes

(b) IB1

(c) C4.5

Figure E.12: Chess end-game (kr).

Appendix F

CFS Applied to 37 UCI Domains

Table F.1 shows results for three machine learning algorithms with and without feature selection by CFS-UC on 37 UCI domains. These domains make up a test suite used for experiments with the WEKA workbench [HDW94], and are representative of those available from the UCI repository. Each accuracy in the table is the average of 50 train and test trials using a $2/3$ training and $1/3$ testing split of the data. Colic, colic.ORIG and hc are all versions of the horse colic dataset. Colic and hc have 22 and 27 attributes respectively and "surgical lesion" as the class. Colic.ORIG has 27 attributes and "pathology cp data" as the class. The version of naive Bayes used in these experiments is part of the WEKA workbench.

Figure F.1 shows the average number of features selected by CFS-UC on these domains. Figure F.2 shows the effect of feature selection by CFS-UC on the size of the trees induced by C4.5.

	Domain	naive Bayes	CFS-UC	IB1	CFS-UC	C4.5	CFS-UC
1	anneal	86.93	84.02−	98.14	97.79	98.33	97.46−
2	audiology	76.85	67.57−	74.96	71.94−	76.87	70.00−
3	autos	60.54	56.80−	69.69	73.57+	70.48	71.66
4	balance-scale	89.27	89.27	83.54	83.54	78.71	78.71
5	breast-cancer	71.35	71.39	69.55	70.55	71.64	71.25
6	breast-w	95.73	95.76	95.55	95.58	94.43	94.44
7	colic	79.35	81.49+	79.86	81.38+	84.21	81.75−
8	colic.ORIG	77.49	81.75+	63.59	60.52−	66.70	66.70
9	hc	83.38	87.44+	80.47	86.08+	84.78	84.61
10	credit-a	81.20	86.00+	81.27	86.00+	84.89	86.00+
11	credit-g	75.05	72.49−	70.11	69.05−	70.73	72.27+
12	diabetes	74.98	76.15+	69.41	69.83	72.29	72.92
13	glass	51.01	53.10+	70.49	72.52+	67.42	66.71
14	heart-c	82.98	82.31	76.31	78.11+	73.11	76.10+
15	heart-h	84.12	84.00	78.56	80.60+	78.70	80.40+
16	heart-statlog	83.78	81.63−	75.00	77.03+	75.48	78.80+
17	hepatitis	84.11	82.34−	80.67	80.49	79.96	80.29
18	hypothyroid	95.52	94.24−	90.81	86.06−	99.50	96.38−
19	ionosphere	74.97	82.87+	86.59	89.12+	89.88	89.92
20	iris	95.02	96.51+	95.49	96.31+	94.98	94.94
21	kr-vs-kp	87.66	90.33+	94.62	90.32−	99.21	90.32−
22	letter	64.96	64.33−	73.14	72.34−	85.82	85.89
23	lymph	83.14	80.24−	79.89	80.94	77.13	75.06−
24	mushroom	96.47	98.54+	100.00	98.54−	100.00	98.54−
25	primary-tumor	45.30	44.83	38.20	39.41+	39.91	40.13
26	promoters	88.74	91.84+	82.36	88.89+	75.79	78.53+
27	segment	79.29	81.61+	96.33	96.48	95.71	95.66
28	sick	94.40	93.84−	95.47	96.12+	98.56	96.08−
29	sonar	69.55	70.82	84.81	81.47−	68.55	71.16+
30	soybean	93.19	90.94−	90.64	83.74−	89.55	81.02−
31	splice	95.18	93.58−	75.22	89.70+	93.46	93.20−
32	vehicle	47.78	48.00	68.80	62.32−	70.80	66.56−
33	vote	90.34	95.50+	92.60	95.61+	95.46	95.73+
34	vote1	87.41	89.14+	88.87	89.28	89.51	89.22
35	vowel	56.99	54.25−	97.32	65.96−	75.03	60.15−
36	waveform-5000	82.12	82.50+	73.33	79.31+	74.14	76.58+
37	zoo	93.09	87.71−	95.53	95.01	93.71	92.34−
	Average:	79.98	80.13	81.55	81.39	82.04	81.01

+, − statistically significant improvement or degradation

Table F.1: Comparison of three learning algorithms with and without feature selection using CFS-UC.

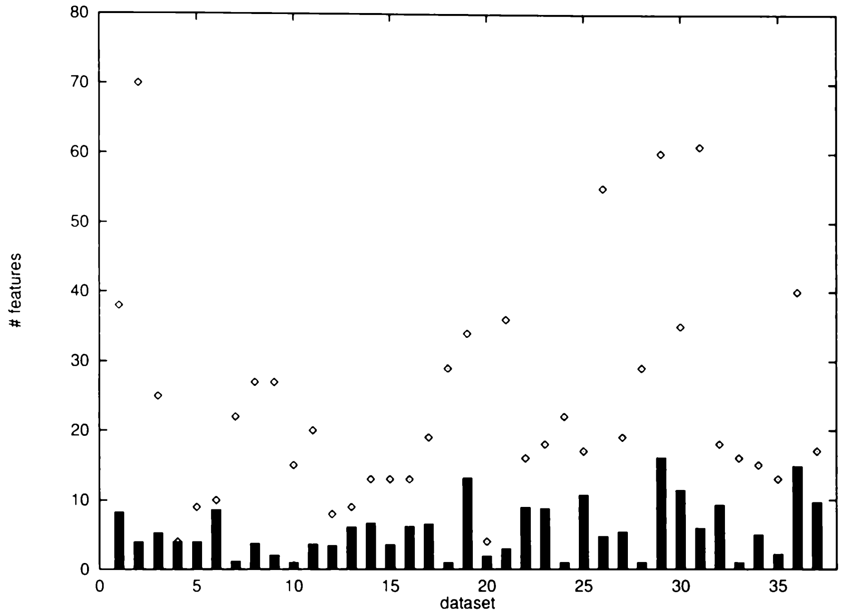


Figure F.1: Average number of features selected by CFS on 37 UCI domains. Dots show the original number of features.

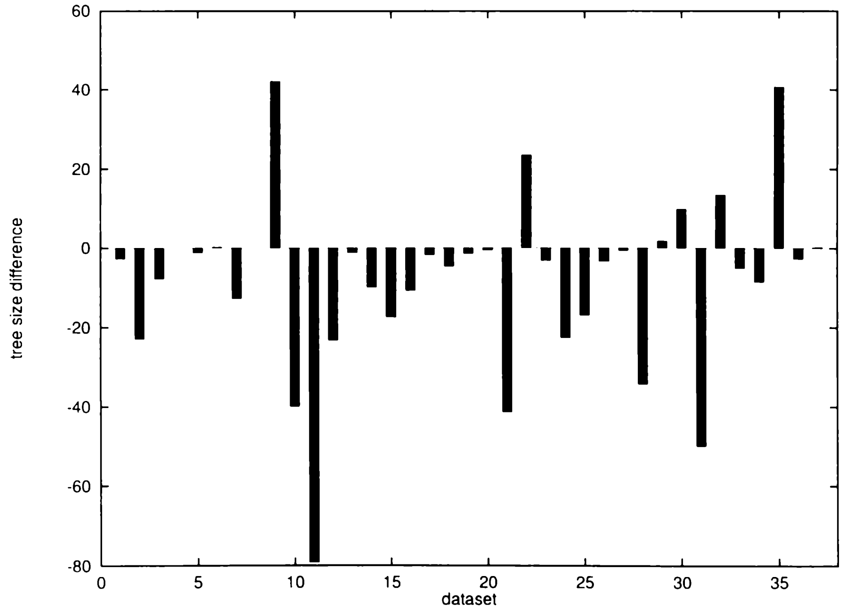


Figure F.2: Effect of feature selection on the size of the trees induced by C4.5 on 37 UCI domains. Bars below the zero line indicate that feature selection has reduced tree size.

Bibliography

[AB94] D. W. Aha and R. L. Blankert. Feature selection for case-based classification of cloud types. In *Working Notes of th AAAI-94 Workshop on Case-Based Reasoning*, pages 106–112, 1994.

[AD91] H. Almuallim and T. G. Dietterich. Learning with many irrelevant features. In *Proceedings of the Ninth National Conference on Artificial Intelligence*, pages 547–542. MIT Press, 1991.

[AD92] H. Almuallim and T. G. Dietterich. Efficient algorithms for identifying relevant features. In *Proceedings of the Ninth Canadian Conference on Artificial Intelligence*, pages 38–45. Morgan Kaufmann, 1992.

[Aha92] D. Aha. Tolerating noisy, irrelevant and novel attributes in instance-based learning algorithms. *International Journal of Man-Machine Studies*, 36:267–287, 1992.

[AKA91] D. W. Aha, D. Kibler, and M. K. Albert. Instance based learning algorithms. *Machine Learning*, 6:37–66, 1991.

[All74] D. Allen. The relationship between variable selection and data augmentation and a method for prediction. *Technometrics*, 16:125–127, 1974.

[BFOS84] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Wadsworth International Group, 1984.

[BI98] D. Bainbridge and S. Inglis. Musical image compression. In *Proceedings of the IEEE Data Compression Conference*, pages 209–218, 1998.

[Bre96a] L. Breiman. Bias, variance, and arcing classifiers. Technical Report 460, University of California, Berkeley, CA., 1996.

[Bre96b] L. Breiman. Techical note: Some properties of splitting criteria. *Machine Learning*, 24:41–47, 1996.

[Car95] C. Cardie. Using decision trees to improve cased-based learning. In *Proceedings of the First International Conference on Knowledge Discovery and Data Mining*. AAAI Press, 1995.

[Cat91] J. Catlett. On changing continuous attributes into ordered discrete attributes. In *Proceedings of the European Working Session on Learning*, pages 164–178, Berlin, 1991. Springer-Verlag.

[CB97] C. W. Codrington and C. E. Brodley. On the qualitative behavour of impurity-based splitting rules: The minima-free property. Technical report, Electrical and Computer Engineering, Purdue University, IN, 1997.

[CF94] R. Caruana and D. Freitag. Greedy attribute selection. In *Machine Learning: Proceedings of the Eleventh International Conference*. Morgan Kaufmann, 1994.

[CLW96] J. G. Cleary, S. Legg, and I. H. Witten. An MDL estimate of the significance of rules. In *Proceedings of ISIS: Information, Statistics, and Induction in Science*, 1996.

[CLW97] S. J. Cunningham, J. Littin, and I. H. Witten. Applications of machine learning in information retrieval. Technical Report 97/6, University of Waikato, 1997.

[CMSW92] R. H. Creecy, B. M. Masand, S. J. Smith, and D. L. Waltz. Trading mips and memory for knowledge engineering. *Communications of the ACM*, 35:48–64, 1992.

[CN89] P. Clark and T. Niblett. The CN2 induction algorithm. *Machine Learning*, 3:261–283, 1989.

[CS93] S. Cost and S. Salzberg. A weighted nearest neighbor algorithm for learning with symbolic features. *Machine Learning*, 10:57–78, 1993.

[CS96] K. J. Cherkauer and J. W. Shavlik. Growing simpler decision trees to facilitate knowledge discovery. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*. AAAI Press, 1996.

[CT95] J. G. Cleary and L. E. Trigg. K*: An instance-based learner using an entropic distance measure. In *Machine Learning: Proceedings of the Twelvth International Conference*. Morgan Kaufmann, 1995.

[Die88] T.G. Dietterich. Approximate statistical tests for comparing supervised classification learnign algorithms. *Neural Computation*, 10(7):1895–1924, 1988.

[DK82] P. A. Devijver and J. Kittler. *Pattern Recognition: A Statistical Approach*. Prentice/Hall International, 1982.

[DKS95] D. Dougherty, R. Kohavi, and M. Sahami. Supervised and unsupervised discretisation of continuous features. In *Machine Learning: Proceedings of the Twelvth International Conference*. Morgan Kaufmann, 1995.

[Dom97] P. Domingos. Context-sensitive feature selection for lazy learners. *Artificial Intelligence Review*, (11):227–253, 1997.

[DP96] P. Domingos and M. Pazzani. Beyond independence: Conditions for the optimality of the simple Bayesian classifier. In *Machine Learning: Proceedings of the Thirteenth International Conference on Machine Learning*. Morgan Kaufmann, 1996.

[DP97] P. Domingos and M. Pazzani. On the optimality of the simple Bayesian classifier under zero-one loss. *Machine Learning*, 29:103–130, 1997.

[FG96] N. Friedman and M. Goldszmidt. Building classifiers using Bayesian networks. In *Proceedings of the National Conference on Artificial Intelligence*, pages 1277–1284, 1996.

[FI93] U. M. Fayyad and K. B. Irani. Multi-interval discretisation of continuous-valued attributes for classification learning. In *Proceedings of the Thirteenth International Join Conference on Artificial Intelligence*. Morgan Kaufmann, 1993.

[FS96] Y. Freund and R. R. Schapire. Experiments with a new boosting algorithm. In *Machine Learning: Proceedings of the Thirteenth International Conference on Machine Learning*. Morgan Kaufmann, 1996.

[Gei75] S. Geisser. The predictive sample reuse method with applications. *Journal of the American Statistical Association*, 70(350):320–328, 1975.

[Ghi64] E. E. Ghiselli. *Theory of Psychological Measurement*. McGrawHill, New York, 1964.

[GL97] D. Gamberger and N. Lavrac. Conditions for Occam's Razor applicability and noise elimination. In *Proccedings of the Ninth European Conference on Machine Learning*, 1997.

[GLF89] J. H. Gennari, P. Langley, and D. Fisher. Models of incremental concept formation. *Artificial Intelligence*, (40):11–61, 1989.

[HDW94] G. Holmes, A. Donkin, and I.H. Witten. Weka: A machine learning workbench. In *Proceedings of the Second Australia and New Zealand Conference on Intelligent Information Systems*, 1994.

[HNM95] G. Holmes and C. G. Nevill-Manning. Feature selection via the discovery of simple classification rules. In *Proceedings of the Symposium on Intelligent Data Analysis*, Baden-Baden, Germany, 1995.

[Hog77] R. M. Hogarth. Methods for aggregating opinions. In H. Jungermann and G. de Zeeuw, editors, *Decision Making and Change in Human Affairs*. D. Reidel Publishing, Dordrecht-Holland, 1977.

[Hol75] J. H. Holland. *Adaption in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, MI, 1975.

[Hol93] R. C. Holte. Very simple classification rules perform well on most commonly used datasets. *Machine Learning*, 11:63–91, 1993.

[Hut93] A. Hutchinson. *Algorithmic Learning*. Clarendon Press, Oxford, 1993.

[JKP94] G. H. John, R. Kohavi, and P. Pfleger. Irrelevant features and the subset selection problem. In *Machine Learning: Proceedings of the Eleventh International Conference*. Morgan Kaufmann, 1994.

[JL96] G. H. John and P. Langley. Static versus dynamic sampling for data mining. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*. AAAI Press, 1996.

[KB91] I. Kononenko and I. Bratko. Information-based evaluation criterion for classifier's performance. *Machine Learning*, 6:67–80, 1991.

[KF94] R. Kohavi and B. Frasca. Useful feature subsets and rough sets reducts. In *Proceedings of the Third International Workshop on Rough Sets and Soft Computing*, 1994.

[Kit78] J. Kittler. Feature set search algorithms. In C. H. Chen, editor, *Pattern Recognition and Signal Processing*. Sijhoff an Noordhoff, the Netherlands, 1978.

[KJ96] R. Kohavi and G. John. Wrappers for feature subset selection. *Artificial Intelligence, special issue on relevance*, 97(1–2):273–324, 1996.

[KJL$^+$94] R. Kohavi, G. John, R. Long, D. Manley, and K. Pfleger. $\mathcal{MLC}$++: A machine learning library in C++. In *Tools with Artificial Intelligence*, pages 740–743. IEEE Computer Scociety Press, 1994.

[KLY97] R. Kohavi, P. Langley, and Y. Yun. The utility of feature weighting in nearest-neighbor algorithms. In *Proceedings of the Ninth European Conference on Machine Learning*, Prague, 1997. Springer-Verlag.

[Koh95a] R. Kohavi. The power of decision tables. In *European Conference on Machine Learning*, 1995.

[Koh95b] R. Kohavi. *Wrappers for Performance Enhancement and Oblivious Decision Graphs*. PhD thesis, Stanford University, 1995.

[Kon91] I. Kononenko. Semi-naive Bayesian classifier. In *Proceedings of the Sixth European Working Session on Learning*, pages 206–219, 1991.

[Kon94] I. Kononenko. Estimating attributes: Analysis and extensions of relief. In *Proceedings of the European Conference on Machine Learning*, 1994.

[Kon95] I. Kononenko. On biases in estimating multi-valued attributes. In *IJCAI95*, pages 1034–1040, 1995.

[KR92] K. Kira and L. A. Rendell. A practical approach to feature selection. In *Machine Learning: Proceedings of the Ninth International Conference*, 1992.

[KS95] R. Kohavi and D. Sommerfield. Feature subset selection using the wrapper method: Overfitting and dynamic search space topology. In *Proceedings of the First International Conference on Knowledge Discovery and Data Mining*. AAAI Press, 1995.

[KS96a] R. Kohavi and M. Sahami. Error-based and entropy-based discretisation of continuous features. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*. AAAI Press, 1996.

[KS96b] D. Koller and M. Sahami. Towards optimal feature selection. In *Machine Learning: Proceedings of the Thirteenth International Conference on Machine Learning*. Morgan Kaufmann, 1996.

[KS97] D. Koller and M. Sahami. Hierachically classifying documents using very few words. In *Machine Learning: Proceedings of the Fourteenth International Conference*. Morgan Kaufmann, 1997.

[Lan94] P. Langley. Selection of relevant features in machine learning. In *Proceedings of the AAAI Fall Symposium on Relevance*. AAAI Press, 1994.

[Lin81] G. H. Lincoff. *The Audubon Society Field Guide to North American Mushrooms*. Alfred A. Knopf, New York, 1981.

[LS94a] P. Langley and S. Sage. Induction of selective Bayesian classifiers. In *Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence*, Seattle, W.A, 1994. Morgan Kaufmann.

[LS94b] P. Langley and S. Sage. Oblivious decision trees and abstract cases. In *Working Notes of the AAAI-94 Workshop on Case-Based Reasoning*, Seattle, W.A, 1994. AAAI Press.

[LS94c] P. Langley and S. Sage. Scaling to domains with irrelevant features. In R. Greiner, editor, *Computational Learning Theory and Natural Learning Systems*, volume 4. MIT Press, 1994.

[LS95] P. Langley and H. A. Simon. Applications of machine learning and rule induction. *Comunications of the ACM*, 38(11):55–64, 1995.

[LS96] H. Liu and R. Setiono. A probabilistic approach to feature selection: A filter solution. In *Machine Learning: Proceedings of the Thirteenth International Conference on Machine Learning*. Morgan Kaufmann, 1996.

[MG63] T. Marill and D. M. Green. On the effectiveness of receptors in recognition systems. *IEEE Transactions on Information Theory*, 9:11–17, 1963.

[MHJ92] A. W. Moore, D. J. Hill, and M. P. Johnson. An empirical investigation of brute force to choose features, smoothers and function approximators. In S. Hanson, S. Judd, and T. Petsche, editors, *Computational Learning Theory and Natural Learning Systems*, volume 3. MIT Press, 1992.

[Mic83] R. S. Michalski. A theory and methodology of inductive learning. *Artificial Intelligence*, 20(2):111–161, 1983.

[Mil90] A. J. Miller. *Subset Selection in Regression*. Chapman and Hall, New York, 1990.

[Mit97] T. M. Mitchell. *Machine Learning*. McGraw-Hill, New York, 1997.

[ML94] A. W. Moore and M. S. Lee. Efficient algorithms for minimizing cross validation error. In *Machine Learning: Proceedings of the Eleventh International Conference*. Morgan Kaufmann, 1994.

[MM98] C. J. Merz and P. M. Murphy. UCI Repository of machine learning databases. http://www.ics.uci.edu/~mlearn/MLRepository.html, 1998.

[Mod93] M. Modrzejewski. Feature selection using rough sets theory. In *Proceedings of the European Conference on Machine Learning*, pages 213–226, 1993.

[NF77] P. M. Narendra and K. Fukunaga. A branch and bound algorithm for feature subset selection. *IEEE Transactions on Computers*, C-26(9), 1977.

[Par87] T. Parsons. *Voice and Speech Processing*. McGraw-Hill, 1987.

[Paw91] Z. Pawlak. *Rough Sets, Theoretical Aspects of Reasoning About Data*. Kluwer, 1991.

[Paz95] M. Pazzani. Searching for dependencies in Bayesian classifiers. In *Proceedings of the Fifth International Workshop on AI and Statistics*, 1995.

[Pea88] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, San Mateo, California, 1988.

[Pfa95] B. Pfahringer. Compression-based feature subset selection. In *Proceedings of the IJCAI-95 Workshop on Data Engineering for Inductive Learning*, pages 109–119, 1995.

[PFTV88] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling. *Numerical Recipes in C*. Cambridge University Press, Cambridge, 1988.

[PS96] G. M. Provan and M. Singh. Learning Bayesian networks using feature selection. In D. Fisher and H. Lenz, editors, *Learning from Data, Lecture Notes in Statistics*, pages 291–300. Springer-Verlag, New York, 1996.

[PSF91] G. Piatetsky-Shapiro and W. J. E. Frawley. *Knowledge Discovery in Databases*. MIT Press, Cambridge, Mass., 1991.

[Qui86] R. R. Quinlan. Induction of decision trees. *Machine Learning*, 1:81–106, 1986.

[Qui87] J. R. Quinlan. Simplifying decision trees. *International Journal of Man-Machine Studies*, 27:221–234, 1987.

[Qui89] J. R. Quinlan. Inferring decision trees using the minimum description length principle. *Information and Computation*, 80:227–248, 1989.

[Qui93] J.R. Quinlan. *C4.5: Programs for machine learning*. Morgan Kaufmann, Los Altos, California, 1993.

[Ris78] J. Rissanen. Modeling by shortest data description. *Automatica*, 14:465–471, 1978.

[RK91] E. Rich and K. Knight. *Artificial Intelligence*. McGraw-Hill, 1991.

[Sah96] M. Sahami. Learning limited dependence Bayesian classifiers. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*. AAAI Press, 1996.

[Sal91] S. Salzberg. A nearest hyperrectangle learning method. *Machine Learning*, 6:251–276, 1991.

[SB97] M. Scherf and W. Brauer. Feature selection by means of a feature weighting approach. Technical Report FKI-221-97, Technische Universität München, 1997.

[Sch93] C. Schaffer. Selecting a classification method by cross-validation. *Machine Learning*, 13:135–143, 1993.

[SFBL97] R. E. Schapire, Y. Freund, P. Bartlett, and W. S. Lee. Boosting the margin: A new explanation for the effectiveness of voting methods. In *Machine Learning: Proceedings of the Fourteenth International Conference*. Morgan Kaufmann, 1997.

[Sie56] S. Siegel. *Nonparametric Statistics*. McGraw-Hill, New York, 1956.

[Ska94] D. B. Skalak. Prototype and feature selection by sampling and random mutation hill climbing algorithms. In *Machine Learning: Proceedings of the Eleventh International Conference*. Morgan Kaufmann, 1994.

[SL95] R. Setiono and H. Liu. Chi2: Feature selection and discretization of numeric attributes. In *Proceedings of the Seventh IEEE International Conference on Tools with Artificial Intelligence*, 1995.

[SP96] M. Singh and G. M. Provan. Efficient learning of selective Bayesian network classifiers. In *Machine Learning: Proceedings of the Thirteenth International Conference on Machine Learning*. Morgan Kaufmann, 1996.

[SW48] C. E. Shannon and W. Weaver. *The Mathematical Theory of Comunication*. University of Illinois Press, Urbana, Ill, 1948.

[TBB$^+$91] S. B. Thrun, J. Bala, E. Bloedorn, I. Bratko, B. Cestnik, J. Cheng, K. De Jonng, S. Dzeroski, S.E Fahlman, D. Fisher, R. Hamann, K. Kaufman, S. Keller, I. Kononenko, J. Kreuziger, R. S. Michalski, T. Mitchell, P. Pachowicz, Y. Reich, H. Vafaie, W. Van de Welde, W. Wenzel, J. Wnek, and J. Zhang. The MONK's problems: A performance comparison of different learning algorithms. Technical Report CMU-CS-91-197, Carnegie Mellon University, 1991.

[Tho92] C. J. Thornton. *Techniques In Computational Learning*. Chapman and Hall, London, 1992.

[Tin95] K. M. Ting. *Common Issues in Instance Based and Naive Bayesian Classifiers*. PhD thesis, University of Sydney, NSW 2006, Australia, 1995.

[VJ95] H. Vafaie and K. De Jong. Genetic algorithms as a tool for restructuring feature space representations. In *Proceedings of the International Conference on Tools with A.I.* IEEE Computer Society Press, 1995.

[WA95] D. Wettschereck and D. W. Aha. Weighting features. In *First International Conference on Cased-Based Reasoning*, Portugal, 1995. Springer.

[WC87] A. K. C Wong and D. K. Y Chiu. Synthesizing statistical knowledge from incomplete mixed-mode data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 9(6), 1987.

[WL94] A. P. White and W. Z. Liu. Bias in information-based measures in decision tree induction. *Machine Learning*, 15:321–329, 1994.

[WW77] T. H. Wonnacott and R. J. Wonnacott. *Introductory Statistics*. Wiley, 1977.

[WW97] Y. Wang and I. H. Witten. Inducing model trees for continuous classes. In *Proceedings of Poster Papers, Ninth European Conference on Machine Learning*, 1997.

[Zaj62] R. B. Zajonic. A note on group judgements and group size. *Human Relations*, 15:177–180, 1962.