Wrappers for Feature Subset Selection

Ron Kohavi
Data Mining and Visualization
Silicon Graphics, Inc.
2011 N. Shoreline Boulevard
Mountain View, CA 94043
ronnyk@sgi.com

George H. John
Global Business Intelligence Solutions
IBM Almaden Research Center
650 Harry Rd.
San Jose, CA 95120
gjohn@CS.Stanford.EDU

http://robotics.stanford.edu/~{ronnyk,gjohn}

May 20, 1997

Abstract

In the feature subset selection problem, a learning algorithm is faced with the problem of selecting a relevant subset of features upon which to focus its attention, while ignoring the rest. To achieve the best possible performance with a particular learning algorithm on a particular training set, a feature subset selection method should consider how the algorithm and the training set interact. We explore the relation between optimal feature subset selection and relevance. Our wrapper method searches for an optimal feature subset tailored to a particular algorithm and a domain. We study the strengths and weaknesses of the wrapper approach and show a series of improved designs. We compare the wrapper approach to induction without feature subset selection and to Relief, a filter approach to feature subset selection. Significant improvement in accuracy is achieved for some datasets for the two families of induction algorithms used: decision trees and Naive-Bayes.

1 Introduction

A universal problem that all intelligent agents must face is where to focus their attention. A problem-solving agent must decide which aspects of a problem are relevant, an expert-system designer must decide which features to use in rules, and so forth. Any learning agent must learn from experience, and discriminating between the relevant and irrelevant parts of its experience is a ubiquitous problem.

In supervised machine learning, an induction algorithm is typically presented with a set of training instances, where each instance is described by a vector of feature (or attribute) values and a class label. For example, in medical diagnosis problems the features might include the age, weight, and blood pressure of a patient, and the class label might indicate whether or not a physician determined that the patient was suffering from heart disease. The task of the induction algorithm, or the *inducer*, is to induce a *classifier* that will be useful in classifying future cases. The classifier is a mapping from the space of feature values to the set of class values.

In the feature subset selection problem, a learning algorithm is faced with the problem of selecting some subset of features upon which to focus its attention, while ignoring the rest. In the wrapper approach (John, Kohavi & Pfleger 1994), the feature subset selection algorithm exists as a wrapper around the induction algorithm. The feature subset selection algorithm conducts a search for a good subset using the induction algorithm itself as part of the function evaluating feature subsets. The idea behind the wrapper approach, shown in Figure 1, is simple: the induction algorithm is considered as a black box. The induction algorithm is run on the dataset, usually partitioned into internal training and holdout sets, with different sets of features removed from the data. The feature subset with the highest evaluation is chosen as the final set on which to run the induction algorithm. The resulting classifier is then evaluated on an independent test set that was *not* used during the search.

Since the typical goal of supervised learning algorithms is to maximize classification accuracy on an unseen test set, we have adopted this as our goal in guiding the feature subset selection. Instead of trying

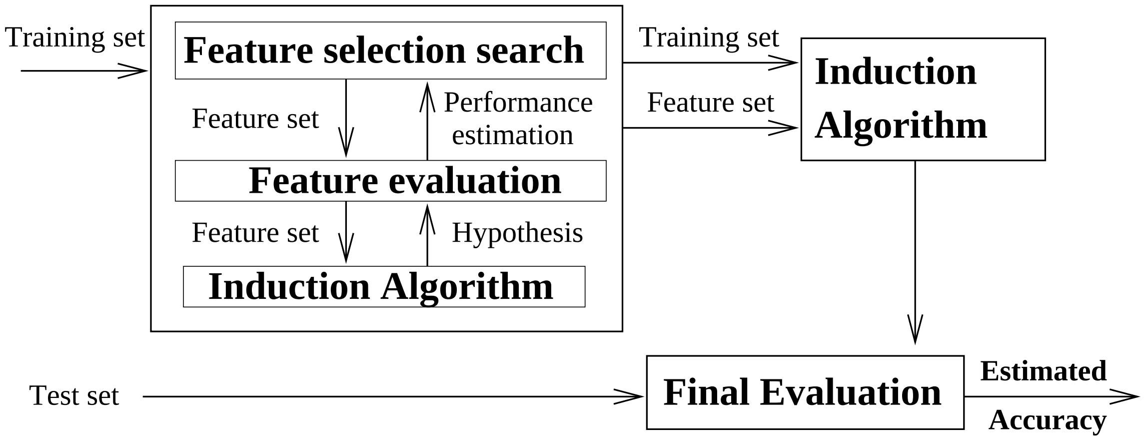

Figure 1: The wrapper approach to feature subset selection. The induction algorithm is used as a "black box" by the subset selection algorithm.

to maximize accuracy, we might instead have tried to identify which features were relevant, and use only those features during learning. One might think that these two goals were equivalent, but we show several examples of problems where they differ.

This paper is organized as follows. In Section 2, we review the feature subset selection problem, investigate the notion of relevance, define the task of finding optimal features, and describe the filter and wrapper approaches. In Section 3, we investigate the search engine used to search for feature subsets and show that greedy search (hill-climbing) is inferior to best-first search. In Section 4, we modify the connectivity of the search space to improve the running time. Section 5 contains a comparison of the best methods found. In Section 6, we discuss one potential problem in the approach, overfitting, and suggest a theoretical model that generalizes the feature subset selection problem in Section 7. Related work is given in Section 8, future work is discussed in Section 9, and we conclude with a summary in Section 10.

2 Feature Subset Selection

If variable elimination has not been sorted out after two decades of work assisted by high-speed computing, then perhaps the time has come to move on to other problems.
—R. L. Plackett, discussion in Miller (1984)

In this section, we look at the problem of finding a good feature subset and its relation to the set of relevant features. We show problems with existing definitions of relevance, and show how partitioning relevant features into two families, weak and strong, helps us understand the issue better. We examine two general approaches to feature subset selection: the filter approach and the wrapper approach, and we then investigate each in detail.

2.1 The Problem

Practical machine learning algorithms, including top-down induction of decision tree algorithms such as ID3 (Quinlan 1986), C4.5 (Quinlan 1993), and CART (Breiman, Friedman, Olshen & Stone 1984), and instance-based algorithms, such as IBL (Dasarathy 1990, Aha, Kibler & Albert 1991), are known to degrade in performance (prediction accuracy) when faced with many features that are not necessary for predicting the desired output. Algorithms such as Naive-Bayes (Langley, Iba & Thompson 1992, Duda & Hart 1973, Good 1965) are robust with respect to irrelevant features (*i.e.*, their performance degrades very slowly as more irrelevant features are added) but their performance may degrade quickly if correlated (even if relevant) features are added.

For example, running C4.5 with the default parameter setting on the Monk1 problem (Thrun *et al.* 1991), which has three irrelevant features, generates a tree with 15 interior nodes, five of which test irrelevant features. The generated tree has an error rate of 24.3%, which is reduced to 11.1% if only the three relevant features are given. John (1997) shows similar examples where adding relevant or irrelevant features to the credit-approval and Pima diabetes datasets degrades the performance of C4.5. Aha (1992) noted that "IB3's storage requirement increases exponentially with the number of irrelevant attributes" (IB3 is a nearest-neighbor algorithm that attempts to save only important prototypes). Performance likewise degrades rapidly with irrelevant features.

The problem of feature subset selection is that of finding a subset of the original features of a dataset, such that an induction algorithm that is run on data containing only these features generates a classifier with the highest possible accuracy. Note that feature subset selection chooses a set of features from existing features, and does not construct new ones; there is no feature extraction or construction (Kittler 1986, Rendell & Seshu 1990).

From a purely theoretical standpoint, the question of which features to use is not of much interest. A Bayes rule, or a Bayes classifier, is a rule that predicts the most probable class for a given instance, based on the full distribution D (assumed to be known). The accuracy of the Bayes rule is the highest possible accuracy, and it is mostly of theoretical interest. The optimal Bayes rule is monotonic, *i.e.*, adding features cannot decrease accuracy, and hence restricting a Bayes rule to a subset of features is never advised.

In practical learning scenarios, however, we are face with two problems: the learning algorithms are not given access to the underlying distribution, and most practical algorithms attempt to find a hypothesis by approximating NP-hard optimization problems. The first problem is closely related to the bias-variance tradeoff (Geman, Bienenstock & Doursat 1992, Kohavi & Wolpert 1996): one must tradeoff estimation of more parameters (bias reduction) with accurately estimating these parameters (variance reduction). This problem is independent of the computational power available to the learner. The second problem, that of finding a "best" (or approximately best) hypothesis, is usually intractable and thus poses an added computational burden. For example, decision tree induction algorithms usually attempt to find a small tree that fits the data well, yet finding the optimal binary decision tree is NP-hard (Hyafil & Rivest 1976, Hancock 1989). For neural-networks, the problem is even harder; the problem of loading a three-node neural network with a training set is NP-hard if the nodes compute linear threshold functions (Judd 1988, Blum & Rivest 1992).

Because of the above problems, we define an optimal feature subset with respect to a particular induction algorithm, taking into account its heuristics, biases, and tradeoffs. The problem of feature subset selection is then reduced to the problem of finding an optimal subset.

Definition 1
Given an inducer $\mathcal{I}$, and a dataset $\mathcal{D}$ with features $X_1, X_2, \ldots, X_n$, from a distribution D over the labeled instance space, an **optimal feature subset, $\mathbf{X_{opt}}$**, *is a subset of the features such that the accuracy of the induced classifier $\mathcal{C} = \mathcal{I}(\mathcal{D})$ is maximal.*

An optimal feature subset need not be unique because it may be possible to achieve the same accuracy using different sets of features (*e.g.*, when two features are perfectly correlated, one can be replaced by the other). By definition, to get the highest possible accuracy, the best subset that a feature subset selection algorithm can select is an optimal feature subset. The main problem with using this definition in practical learning scenarios is that one does not have access to the underlying distribution and must estimate the classifier's accuracy from the data.

2.2 Relevance of Features

One important question is the relation between optimal features and relevance. In this section, we present definitions of relevance that have been suggested in the literature.[1] We then show a single example where the definitions give unexpected answers, and we suggest that two degrees of relevance are needed: weak and strong.

[1] In general, the definitions given here are only applicable to discrete features, but can be extended to continuous features by changing $p(X = x)$ to $p(X \leq x)$.

Definition	Relevant	Irrelevant
Definition 2	X_1	X_2, X_3, X_4, X_5
Definition 3	None	All
Definition 4	All	None
Definition 5	X_1	X_2, X_3, X_4, X_5

Table 1: Feature relevance for the Correlated XOR problem under the four definitions.

2.2.1 Existing Definitions

Almuallim & Dietterich (1991, p. 548) define relevance under the assumptions that all features and the label are Boolean and that there is no noise.

Definition 2
A feature X_i is said to be **relevant** *to a concept $\mathcal{C}$ if X_i appears in every Boolean formula that represents $\mathcal{C}$ and* **irrelevant** *otherwise.*

Gennari, Langley & Fisher (1989, Section 5.5) allow noise and multi-valued features and define relevant features as those whose "values vary systematically with category membership." We formalize this definition as follows.

Definition 3
X_i is **relevant** *iff there exists some x_i and y for which $p(X_i = x_i) > 0$ such that*

$$p(Y = y \mid X_i = x_i) \neq p(Y = y) \ .$$

Under this definition, X_i is relevant if knowing its value can change the estimates for the class label Y, or in other words, if Y is conditionally dependent on X_i. Note that this definition fails to capture the relevance of features in the parity concept where all unlabeled instances are equiprobable, and it may therefore be changed as follows.

Let $S_i = \{X_1, \ldots, X_{i-1}, X_{i+1}, \ldots, X_m\}$, the set of all features except X_i. Denote by s_i a value-assignment to all features in S_i.

Definition 4
X_i is **relevant** *iff there exists some x_i, y, and s_i for which $p(X_i = x_i) > 0$ such that*

$$p(Y = y, S_i = s_i \mid X_i = x_i) \neq p(Y = y, S_i = s_i) \ .$$

Under the following definition, X_i is relevant if the probability of the label (given all features) can change when we eliminate knowledge about the value of X_i.

Definition 5
X_i is **relevant** *iff there exists some x_i, y, and s_i for which $p(X_i = x_i, S_i = s_i) > 0$ such that*

$$p(Y = y \mid X_i = x_i, S_i = s_i) \neq p(Y = y \mid S_i = s_i) \ .$$

The following example shows that all the definitions above give unexpected results.

Example 1 (Correlated XOR) Let features $X_1, \ldots, X_5$ be Boolean. The instance space is such that X_2 and X_3 are negations of X_4 and X_5, respectively, *i.e.*, $X_4 = \overline{X_2}$, $X_5 = \overline{X_3}$. There are only eight possible instances, and we assume they are equiprobable. The (deterministic) target concept is

$$Y = X_1 \oplus X_2 \qquad (\oplus \text{ denotes XOR}) \ .$$

Note that the target concept has an equivalent Boolean expression, namely, $Y = X_1 \oplus \overline{X_4}$. The features X_3 and X_5 are irrelevant in the strongest possible sense. X_1 is indispensable, and either but not both of

$\{X_2, X_4\}$ can be disposed of. Table 1 shows for each definition, which features are relevant, and which are not.

According to Definition 2, X_3 and X_5 are clearly irrelevant; both X_2 and X_4 are irrelevant because each can be replaced by the negation of the other. By Definition 3, all features are irrelevant because for any output value y and feature value x, there are two instances that agree with the values. By Definition 4, every feature is relevant because knowing its value changes the probability of four of the eight possible instances from 1/8 to zero. By Definition 5, X_3 and X_5 are clearly irrelevant, and both X_2 and X_4 are irrelevant because they do not add any information to S_4 and S_2, respectively. ∎

Although such simple negative correlations are unlikely to occur, domain constraints create a similar effect. When a nominal feature such as color is encoded as input to a neural network, it is customary to use a *local encoding*, where each value is represented by an indicator feature. For example, the local encoding of a four-valued nominal $\{a, b, c, d\}$ would be $\{0001, 0010, 0100, 1000\}$. Under such an encoding, any single indicator feature is redundant and can be determined by the rest. Thus most definitions of relevance will declare all indicator features to be irrelevant.

2.2.2 Strong and Weak Relevance

We now claim that two degrees of relevance are required: weak and strong. Relevance should be defined in terms of a Bayes classifier—the optimal classifier for a given problem. A feature X is **strongly relevant** if removal of X alone will result in performance deterioration of an optimal Bayes classifier. A feature X is **weakly relevant** if it is not strongly relevant and there exists a subset of features, S, such that the performance of a Bayes classifier on S is worse than the performance on $S \cup \{X\}$. A feature is **irrelevant** if it is not strongly or weakly relevant.

Definition 5 repeated below defines strong relevance. Strong relevance implies that the feature is indispensable in the sense that it cannot be removed without loss of prediction accuracy. Weak relevance implies that the feature can sometimes contribute to prediction accuracy.

Definition 5 (Strong relevance)
A feature X_i is **strongly relevant** *iff there exists some x_i, y, and s_i for which $p(X_i = x_i, S_i = s_i) > 0$ such that*

$$p(Y = y \mid X_i = x_i, S_i = s_i) \neq p(Y = y \mid S_i = s_i) .$$

Definition 6 (Weak relevance)
A feature X_i is **weakly relevant** *iff it is not strongly relevant, and there exists a subset of features S'_i of S_i for which there exists some x_i, y, and s'_i with $p(X_i = x_i, S'_i = s'_i) > 0$ such that*

$$p(Y = y \mid X_i = x_i, S'_i = s'_i) \neq p(Y = y \mid S'_i = s'_i) .$$

A feature is **relevant** if it is either weakly relevant or strongly relevant; otherwise, it is **irrelevant**.

In Example 1, feature X_1 is strongly relevant; features X_2 and X_4 are weakly relevant; and X_3 and X_5 are irrelevant.

2.3 Relevance and Optimality of Features

A Bayes classifier must use all strongly relevant features and possibly some weakly relevant features. Classifiers induced from data, however, are likely to be suboptimal, as they have no access to the underlying distribution; furthermore, they may be using restricted hypothesis spaces that cannot utilize all features (see example below). Practical induction algorithms that generate classifiers may benefit from the omission of features, including strongly relevant features. Relevance of a feature does not imply that it is in the optimal feature subset and, somewhat surprisingly, irrelevance does not imply that it should not be in the optimal feature subset (Example 3).

Figure 2: The feature filter approach, in which the features are filtered independently of the induction algorithm.

Example 2 (Relevance does not imply optimality) Let the universe of possible instances be $\{0,1\}^3$, that is, three Boolean features, say X_1, X_2, X_3. Let the distribution of instances be uniform, and assume the target concept is $f(X_1, X_2, X_3) = (X_1 \wedge X_2) \ \vee \ X_3$. Under any reasonable definition of relevance, all features are relevant to this target function.

If the hypothesis space is the space of monomials, *i.e.*, conjunctions of literals, the only optimal feature subset is $\{X_3\}$. The accuracy of the monomial X_3 is 87.5%, the highest accuracy achievable within this hypothesis space. Adding another feature to the monomial will decrease the accuracy. ∎

The example above shows that relevance (even strong relevance) does not imply that a feature is in an optimal feature subset. Another example is given in Section 3.2, where hiding features from ID3 improves performance even when we know they are strongly relevant for an artificial target concept (Monk3). Another question is whether an irrelevant feature can ever be in an optimal feature subset. The following example shows that this may be true.

Example 3 (Optimality does not imply relevance) Assume there exists a feature that always takes the value one. Under all the definitions of relevance described above, this feature is irrelevant. Now consider a limited Perceptron classifier (Rosenblatt 1958, Minsky & Papert 1988) that has an associated weight with each feature and then classifies instances based upon whether the linear combination is greater than zero (the threshold is fixed at zero). (Contrast this with a regular Perceptron that classifies instances depending on whether the linear combination is greater than some threshold, not necessarily zero.) Given this extra feature that is always set to one, the limited Perceptron is equivalent in representation power to the regular Perceptron. However, removal of all irrelevant features would remove that crucial feature.

In Section 4, we show an interesting problem with using any filter approach with Naive-Bayes. One of the artificial datasets (m-of-n-3-7-10) represents a symmetric target function, implying that all features should be ranked equally by any filtering method. However, Naive-Bayes improves if a single feature (any one of them) is removed.

We believe that cases such as those depicted in Example 3 are rare in practice and that irrelevant features should generally be removed. However, it is important to realize that relevance according to these definitions does not imply membership in the optimal feature subset, and that irrelevance does not imply that a feature cannot be in the optimal feature subset.

2.4 The Filter Approach

There are a number of different approaches to subset selection. In this section, we review existing approaches in machine learning. We refer the reader to Section 8 for related work in Statistics and Pattern Recognition. The reviewed methods for feature subset selection follow the *filter approach* and attempt to assess the merits of features from the data, ignoring the induction algorithm.

The filter approach, shown in Figure 2, selects features using a preprocessing step. The main disadvantage of the filter approach is that it totally ignores the effects of the selected feature subset on the performance of the induction algorithm. We now review some existing algorithms that fall into the filter approach.

2.4.1 The FOCUS Algorithm

The FOCUS algorithm (Almuallim & Dietterich 1991, Almuallim & Dietterich 1994), originally defined for noise-free Boolean domains, exhaustively examines all subsets of features, selecting the minimal subset of features that is sufficient to determine the label value for all instances in the training set. This preference for a small set of features is referred to as the MIN-FEATURES bias.

This bias has severe implications when applied blindly without regard for the resulting induced concept. For example, in a medical diagnosis task, a set of features describing a patient might include the patient's social security number (SSN). (We assume that features other than SSN are sufficient to determine the correct diagnosis.) When FOCUS searches for the minimum set of features, it will pick the SSN as the only feature needed to uniquely determine the label[2]. Given only the SSN, any induction algorithm is expected to generalize very poorly.

2.4.2 The Relief Algorithm

The Relief algorithm (Kira & Rendell 1992*a*, Kira & Rendell 1992*b*, Kononenko 1994) assigns a "relevance" weight to each feature, which is meant to denote the relevance of the feature to the target concept. Relief is a randomized algorithm. It samples instances randomly from the training set and updates the relevance values based on the difference between the selected instance and the two nearest instances of the same and opposite class (the "near-hit" and "near-miss"). The Relief algorithm attempts to find *all* relevant features:

> Relief does not help with redundant features. If most of the given features are relevant to the concept, it would select most of them even though only a fraction are necessary for concept description (Kira & Rendell 1992*a*, page 133).

In real domains, many features have high correlations with the label, and thus many are weakly relevant, and will not be removed by Relief. In the simple parity example used in (Kira & Rendell 1992*a*, Kira & Rendell 1992*b*), there were only strongly relevant and irrelevant features, so Relief found the strongly relevant features most of the time. The Relief algorithm was motivated by nearest-neighbors and it is good specifically for similar types of induction algorithms.

In preliminary experiments, we found significant variance in the relevance rankings given by Relief. Since Relief randomly samples instances and their neighbors from the training set, the answers it gives are unreliable without a very high number of samples. In our experiments, the required number of samples was on the order of two to three times the number of cases in the training set. We were worried by this variance, and implemented a deterministic version of Relief that uses all instances and all nearest-hits and nearest-misses of each instance. (For example, if there are two nearest instances equally close to the reference instance, we average both of their contributions instead of picking one.) This gives the results one would expect from Relief if run for an infinite amount of time, but requires only as much time as the standard Relief algorithm with the number of samples equal to the size of the training set. Since we are no longer worried by high variance, we call this deterministic variant Relieve*d*. We handle unknown values by setting the difference between two unknown values to 0 and the difference between an unknown and any other known value to one.

Relief as originally described can only run on binary classification problems, so we used the Relief-F method described by Kononenko (1994), which generalizes Relief to multiple classes. We combined Relief-F with our deterministic enhancement to yield the final algorithm Relieved-F. In our experiments, features with relevance rankings below 0 were removed.

2.4.3 Feature Filtering Using Decision Trees

Cardie (1993) used a decision tree algorithm to select a subset of features for a nearest-neighbor algorithm. Since a decision tree typically contains only a subset of the features, those that appeared in the final tree were selected for the nearest-neighbor. The decision tree thus serves as the filter for the nearest-neighbor algorithm.

Although the approach worked well for some datasets, it has some major shortcomings. Features that are good for decision trees are not necessarily useful for nearest-neighbor. As with Relief, one expects that the totally irrelevant features will be filtered out, and this is probably the major effect that led to some

[2] This is true even if SSN is encoded in 30 binary features as long as more than 30 other binary features are required to determine the diagnosis. Specifically, two real-valued attributes, each one with 16 bits of precision, will be inferior under this scheme.

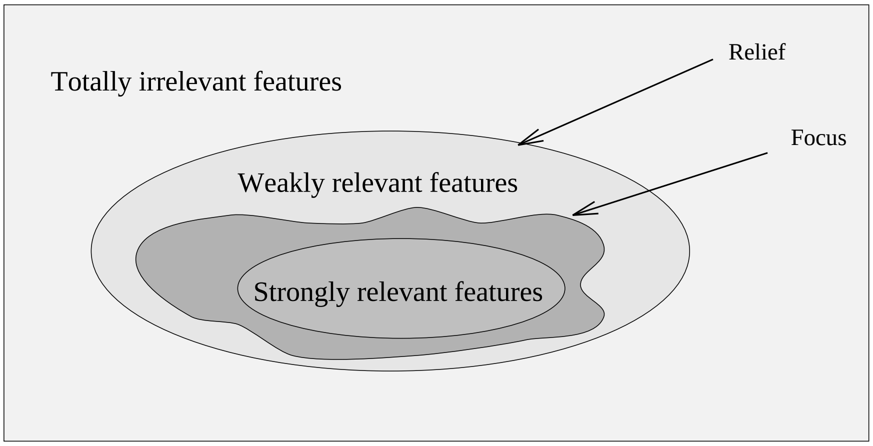

Figure 3: A view of feature set relevance.

improvements in the datasets studied. However, while a nearest-neighbor algorithm can take into account the effect of many relevant features, the current methods of building decision trees suffer from data fragmentation and only a few splits can be made before the number of instances is exhausted. If the tree is approximately balanced and the number of training instances that trickles down to each subtree is approximately the same, then a decision tree cannot test more than $O(\lg m)$ features in a path.

2.4.4 Summary of Filter Approaches

Figure 3 shows the set of features that FOCUS and Relief search for. While FOCUS is searching for a minimal set of features, Relief searches for all the relevant features (both weak and strong).

Filter approaches to feature subset selection do not take into account the biases of the induction algorithms and select feature subsets that are independent of the induction algorithms. In some cases, measures can be devised that are algorithm specific, and these may be computed efficiently. For example, measures such as Mallow's C_p (Mallows 1973) and PRESS (Prediction sum of squares) (Neter, Wasserman & Kutner 1990) have been devised specifically for linear regression. These measures and the relevance measure assigned by Relief would not be appropriate as feature subset selectors for algorithms such as Naive-Bayes because in some cases the performance of Naive-Bayes improves with the removal of relevant features.

The Corral dataset, which is an artificial dataset from (John et al. 1994) gives a possible scenario where filter approaches fail miserably. There are 32 instances in this Boolean domain. The target concept is

$$(A0 \wedge A1) \ \vee \ (B0 \wedge B1) \ .$$

The feature named "irrelevant" is uniformly random, and the feature "correlated" matches the class label 75% of the time. Greedy strategies for building decision trees pick the "correlated" feature as it seems best by all known selection criteria. After the "wrong" root split, the instances are fragmented and there are not enough instances at each subtree to describe the correct concept. Figure 4 shows the decision tree induced by C4.5. CART induces a similar decision tree with the "correlated" feature at the root. When this feature is removed, the correct tree is found. Because the "correlated" feature is highly correlated with the label, filter algorithms will generally select it. Wrapper approaches, on the other hand, may discover that the feature is hurting performance and will avoid selecting it.

These examples and the discussion of relevance versus optimality (Section 2.3) show that a feature selection scheme should take the induction algorithm into account, as is done in the wrapper approach.

2.5 The Wrapper Approach

In the wrapper approach, shown in Figure 1, the feature subset selection is done using the induction algorithm as a black box (*i.e.*, no knowledge of the algorithm is needed, just the interface). The feature subset

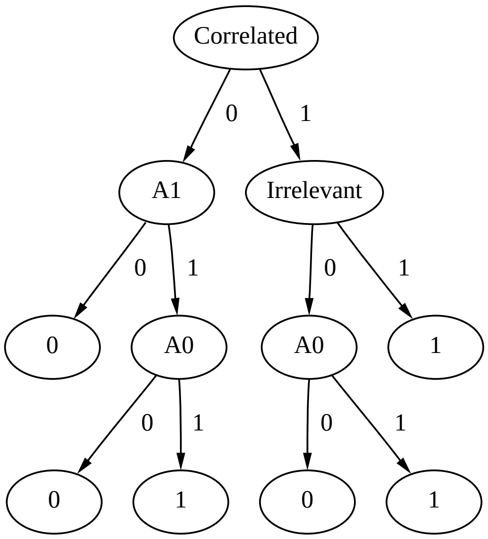

Figure 4: The tree induced by C4.5 for the "Corral" dataset, which fools top-down decision-tree algorithms into picking the "correlated" feature for the root, causing fragmentation, which in turns causes the irrelevant feature to be chosen.

selection algorithm conducts a search for a good subset using the induction algorithm itself as part of the evaluation function. The accuracy of the induced classifiers is estimated using accuracy estimation techniques (Kohavi 1995*b*). The problem we are investigating is that of state space search, and different search engines will be investigated in the next sections.

The wrapper approach conducts a search in the space of possible parameters. A search requires a state space, an initial state, a termination condition, and a search engine (Ginsberg 1993, Russell & Norvig 1995). The next section focuses on comparing search engines: hill-climbing and best-first search.

The search space organization that we chose is such that each state represents a feature subset. For n features, there are n bits in each state, and each bit indicates whether a feature is present (1) or absent (0). Operators determine the connectivity between the states, and we have chosen to use operators that add or delete a single feature from a state, corresponding to the search space commonly used in stepwise methods in Statistics. Figure 5 shows such the state space and operators for a four-feature problem. The size of the search space for n features is $O(2^n)$, so it is impractical to search the whole space exhaustively, unless n is small. We will shortly describe the different search engines that we compared.

The goal of the search is to find the state with the highest evaluation, using a heuristic function to guide it. Since we do not know the actual accuracy of the induced classifier, we use accuracy estimation as both the heuristic function and the evaluation function (See Section 7 for more details on the abstract problem). The evaluation function we use is five-fold cross-validation (Figure 6), repeated multiple times. The number of repetitions is determined on the fly by looking at the standard deviation of the accuracy estimate, assuming they are independent. If the standard deviation of the accuracy estimate is above 1% and five cross-validations have not been executed, we execute another cross-validation run. While this is only a heuristic, it seems to work well in practice and avoids multiple cross-validation runs for large datasets.

This heuristic has the nice property that it forces the accuracy estimation to execute cross-validation more times on small datasets than on large datasets. Because small datasets require less time to learn, the overall accuracy estimation time, which is the product of the induction algorithm running time and the cross-validation time, does not grow too fast. We thus have a conservation of "hardness" using this heuristic: small datasets will be cross-validated many times to overcome the high variance resulting from small amounts of data. For much larger datasets, one could switch to a holdout heuristic to save even more time (a factor of five), but we have not found this necessary for the datasets we used.

The term **forward selection** refers to a search that begins at the empty set of features; the term **backward elimination** refers to a search that begins at the full set of features (Devijver & Kittler 1982,

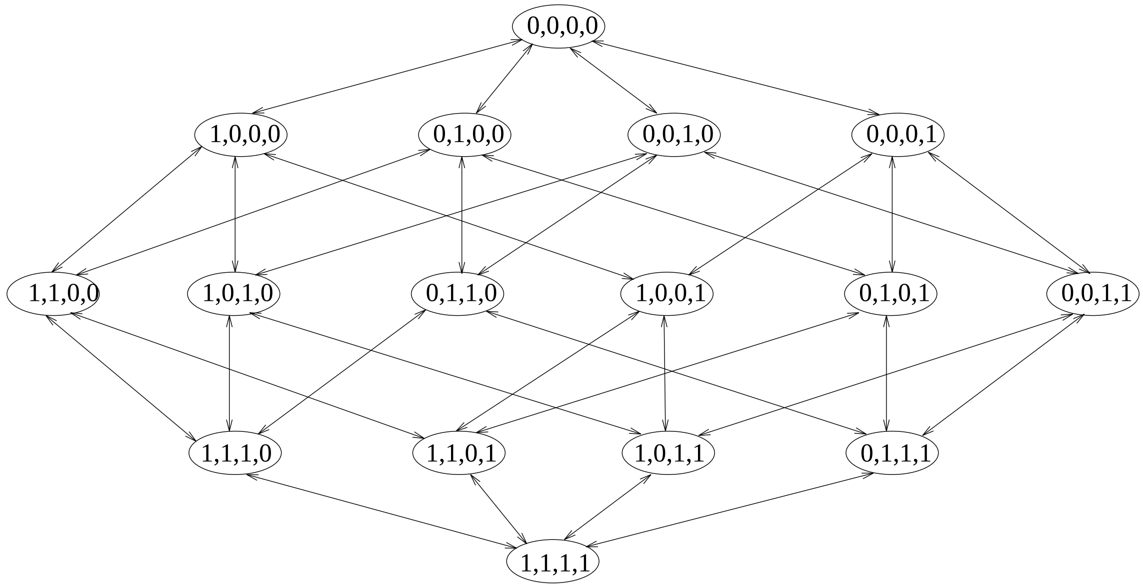

Figure 5: The state space search for feature subset selection. Each node is connected to nodes that have one feature deleted or added.

Miller 1990). The initial state we use in most of our experiments is the empty set of features, hence we are using a forward selection approach. The main reason for this choice is computational: building classifiers when there are few features in the data is much faster. Although in theory, going backward from the full set of features may capture interacting features more easily, the method is extremely expensive with only the add-feature and delete-feature operators. In Section 4, we will introduce compound operators that will make the backward elimination approach practical. The following summary shows the instantiation of the search problem:

State	A Boolean vector, one bit per feature
Initial state	The empty set of features (0,0,0..., 0)
Heuristic/evaluation	Five-fold cross-validation repeated multiple times with a small penalty (0.1%) for every feature.
Search algorithm	Hill-climbing or best-first search
Termination condition	Algorithm dependent (see below)

A **complexity penalty** was added to the evaluation function, penalizing feature subsets with many features so as to break ties in favor of smaller subsets. The penalty was set to 0.1%, which is very small compared to the standard deviation of the accuracy estimation, which is aimed to be below 1%. No attempts were made to set this value optimally for the specific datasets. It was simply added to pick the smaller of two feature subsets that have the same estimated accuracy.

3 The Search Engine

In this section we evaluate different search engines for the wrapper approach. We begin with a description of the experimental methodology used in the rest of the paper. We then describe the hill-climbing (greedy) search engine, and show that it terminates at local maxima too often. We then use a best-first search engine and show that it works much better.

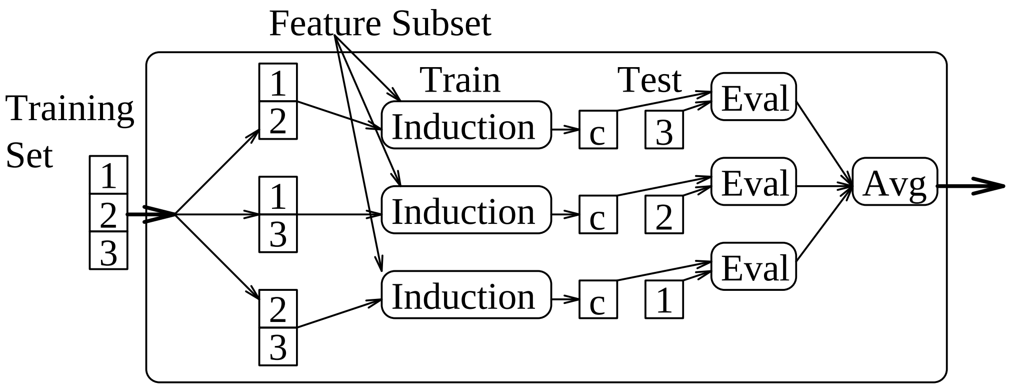

Figure 6: The cross-validation method for accuracy estimation. (3-fold cross-validation shown.)

Table 2: Summary of datasets. Datasets above the horizontal line are "real" and those below are artificial. CV indicates ten-fold cross-validation.

no.	Dataset	Features			no.	Train	Test	baseline
		all	nominal	continuous	classes	size	size	accuracy
1	breast cancer	10	0	10	2	699	CV	65.52
2	cleve	13	7	6	2	303	CV	54.46
3	crx	15	9	6	2	690	CV	55.51
4	DNA	180	180	0	3	2000	1186	51.91
5	horse-colic	22	15	7	2	368	CV	63.04
6	Pima	8	0	8	2	768	CV	65.10
7	sick-euthyroid	25	18	7	2	2108	1055	90.74
8	soybean-large	35	35	0	19	683	CV	13.47
9	Corral	6	6	0	2	32	128	56.25
10	m-of-n-3-7-10	10	10	0	2	300	1024	77.34
11	Monk1	6	6	0	2	124	432	50.00
12	Monk2-local	17	17	0	2	169	432	67.13
13	Monk2	6	6	0	2	169	432	67.13
14	Monk3	6	6	0	2	122	432	52.78

3.1 Experimental Methodology

We now describe the datasets we chose, the algorithms used, and the experimental methodology.

3.1.1 Datasets

Table 2 provides a summary of the characteristics of the datasets chosen. All datasets except for Corral were obtained from the University of California at Irvine repository (Merz & Murphy 1996), from which full documentation for all datasets can be obtained. Corral was introduced in John et al. (1994) and was defined above. The primary criteria were size (real datasets must have more than 300 instances), difficulty (the accuracy should not be too high after seeing only a small number of instances), age (old datasets at the UC Irvine repository, such as Chess, hypothyroid, and vote, were not considered because of their possible influence on the development of algorithms). A detailed description of the datasets and these considerations is given in Kohavi (1995c). Small datasets were tested using ten-fold cross-validation; artificial datasets and large datasets were split into training and testing sets (the artificial datasets have a well-defined training set, as does the DNA dataset from StatLog (Taylor, Michie & Spiegalhalter 1994)). The baseline accuracy is the accuracy (on the whole dataset) when predicting the majority class.

3.1.2 Algorithms

We use two families of induction algorithms as a basis for comparisons. These are the decision-tree and the Naive-Bayes induction algorithms. Both are well known in the machine learning community and represent two completely different approaches to learning, hence we hope that our results are of a general nature and will generalize to other induction algorithms. Decision trees have been well documented in Quinlan (1993), Breiman et al. (1984), Fayyad (1991), Buntine (1992), and Moret (1982); hence we will describe them briefly. The Naive-Bayes algorithm is explained below. The specific details are not essential for the rest of the paper.

The **C4.5** algorithm (Quinlan 1993) is a descendent of ID3 (Quinlan 1986), which builds decision trees top-down and prunes them. In our experiments we used release 7 of C4.5. The tree is constructed by finding the best single-feature test to conduct at the root node of the tree. After the test is chosen, the instances are split according to the test, and the subproblems are solved recursively. C4.5 uses gain ratio, a variant of mutual information, as the feature selection measure; other measures have been proposed, such as the Gini index (Breiman et al. 1984), C-separators (Fayyad & Irani 1992), distance-based measures (De Mántaras 1991), and Relief (Kononenko 1995). C4.5 prunes by using the upper bound of a confidence interval on the resubstitution error as the error estimate; since nodes with fewer instances have a wider confidence interval, they are removed if the difference in error between them and their parents is not significant.

We reserve the term **ID3** to a run of C4.5 that does not execute the pruning step and builds the full tree (*i.e.*, nodes are split unless they are pure or it is impossible to further split the node due to conflicting instances). The ID3 induction algorithm we used is really C4.5 with the parameters `-m1 -c100` that cause a full tree to be grown and only pruned if there is absolutely no increase in the resubstitution error rate. A relatively unknown post processing step in C4.5 replaces a node by one of its children if the accuracy of the child is considered better (Quinlan 1993, page 39). In one case (the Corral database described below), this had a significant impact on the resulting tree: although the root split was incorrect, it was replaced by one of the children.

The **Naive-Bayesian** classifier (Domingos & Pazzani 1996, Langley et al. 1992, Duda & Hart 1973, Good 1965, Anderson & Matessa 1992, Taylor et al. 1994) uses Bayes' rule to compute the probability of each class given the instance, assuming the features are conditionally independent given the label. Formally,

$$\begin{array}{ll}
p(Y = y \mid \vec{X} = \vec{x}) & \\
= p(\vec{X} = \vec{x} \mid Y = y) \cdot p(Y = y) / p(\vec{X} = \vec{x}) & \text{by Bayes rule} \\
\propto p(X_1 = x_1, \ldots, X_n = x_n \mid Y = y) \cdot p(Y = y) & p(\vec{X} = \vec{x}) \text{ is same for all label values.} \\
= \prod_{i=1}^{n} p(X_i = x_i \mid Y = y) \cdot p(Y = y) & \text{by independence}
\end{array}$$

The version of Naive-Bayes we use in our experiments was implemented in $\mathcal{MLC}$++ (Kohavi, Sommerfield & Dougherty 1996). The probabilities for nominal features are estimated from data using maximum likelihood estimation. Continuous features are discretized using a minimum-description length procedure described in Dougherty, Kohavi & Sahami (1995), and were thereafter treated as multi-valued nominals. Unknown values in a test instance (an instance that needs to be labeled) are ignored, *i.e.*, they do not participate in the product. In case of zero occurrences for a label value and a feature value, we use the $.5/m$ as the probability, where m is the number of instances. Other approaches are possible, such as using Laplace's law of succession or using a beta prior (Good 1965, Cestnik 1990). In these approaches, the probability for n successes after N trials is estimated at $(n + a)/(N + a + b)$, where a and b are the parameters of the beta function. The most common choice is to set a and b to one, and estimating the probability as $(n + 1)/(N + 2)$, which is Laplace's law of succession.

3.1.3 Results

When comparing a pair of algorithms, we will present accuracy results for each algorithm on each dataset. It is critical to understand that when we used ten-fold cross-validation for evaluation, this cross-validation is an independent outer loop, not the same as the inner, repeated five-fold cross-validation that is a part of the feature subset selection algorithms. Previously, some researchers have reported accuracy results from the inner cross-validation loop; such results are optimistically biased and are a subtle means of training on the test set.

Table 3: A hill-climbing search algorithm

1. Let $v \leftarrow$ initial state.
2. Expand v: apply all operators to v, giving v's children.
3. Apply the evaluation function f to each child w of v.
4. Let v' = the child w with highest evaluation $f(w)$.
5. If $f(v') > f(v)$ then $v \leftarrow v'$; goto 2.
6. Return v.

Our reported accuracies are the mean of the ten accuracies from ten-fold cross-validation. We also show the standard deviation of the mean. To determine whether the difference between two algorithms is significant or not, we report the p-values, which indicate the probability that one algorithm is better than the other, where the variance of the test is the average variance of the two algorithms and a normal distribution is assumed. A more powerful method would have been to conduct a paired t-test for each instance tested, or for each fold, but the overall picture would not change much.

Whenever we compare two or more algorithms, A_1 and A_2, we give the table of accuracies, and show two bar graphs. One bar graph shows the absolute difference, $A_2 - A_1$, in accuracies and the second bar graph shows the mean accuracy difference divided by the standard deviation, *i.e.*, $(A_2 - A_1)$/std-dev. When the length of the bars on the standard-deviation chart are higher than two, the results are significant at the 95% confidence level. Comparisons will generally be made such that A_2 is the algorithm proposed just prior to the comparison (the "new" algorithm) and A_1 is either a standard algorithm, such as C4.5, or the previous proposed algorithm. When the bar is above zero A_2, the proposed algorithm, outperforms A_1, which we are comparing with.

When we report CPU time results, these are in units of CPU seconds (or minutes or hours) on a Sun Sparc 10 for a single train-test sequence.

3.2 A Hill-climbing Search Engine

The simplest search technique is hill-climbing, also called greedy search or steepest ascent. Table 3 describes the algorithm, which expands the current node and moves to the child with the highest accuracy, terminating when no child improves over the current node.

Table 4 and Figures 7 and 8 show a comparison of ID3 and Naive-Bayes, both with and without feature subset selection. Table 5 and Figure 9 and 10 show the average number of features used for each algorithm (averaged over the ten folds when relevant). The following observations can be made:

- For the real datasets and ID3, this simple version of feature subset selection provides a regularization mechanism, which reduces the variance of the algorithm (Geman et al. 1992, Kohavi & Wolpert 1996). By hiding features from ID3, a smaller tree is grown. This type of regularization is different than pruning, which is another regularization method, because it is global: a feature is either present or absent, whereas pruning is a local operation. As shown in Table 5 and Figures 9 and 10, the number of features selected is small compared to the original set and compared to those selected by ID3. For ID3, the average accuracy increases from 84.53% to 86.67%, which is a 13.8% relative reduction in the error rate. The accuracy uniformly improves for all real datasets.

- For the artificial datasets and ID3, the story is different. All the artificial datasets, except Monk3 involve high-order interactions. In the Corral dataset, after the correlated feature is chosen, no single addition of a feature will lead to an improvement, so the hill-climbing process stops too early; similar scenarios happen with the other artificial datasets, where adding a single feature at a time does not help. In some cases, such as m-of-n-3-7-10, Monk2-local, and Monk2, zero features were chosen, causing the prediction to be the majority class independent of the attribute values.

Table 4: A comparison of ID3 and Naive-Bayes with a feature subset selection wrapper (hill-climbing search). The "-FSS" suffix indicates an algorithm is run with feature subset selection. The first p-val column indicates the probability that feature subset selection (FSS) improves ID3 and the second column indicates the probability that FSS improves Naive-Bayes.

	Dataset	ID3	ID3-FSS	p-val	Naive-Bayes	NB-FSS	p-val
1	breast cancer	94.57± 0.9	94.71± 0.5	0.58	97.00± 0.5	96.57± 0.6	0.22
2	cleve	72.35± 2.3	78.24± 2.0	1.00	82.88± 2.3	79.56± 3.9	0.15
3	crx	81.16± 1.4	85.65± 1.6	1.00	87.10± 0.8	85.36± 1.6	0.08
4	DNA	90.64± 0.9	94.27± 0.7	1.00	93.34± 0.7	94.52± 0.7	0.96
5	horse-colic	81.52± 2.0	83.15± 1.1	0.84	79.86± 2.5	83.15± 2.0	0.93
6	Pima	68.73± 2.5	69.52± 2.2	0.63	75.90± 1.8	74.34± 2.0	0.21
7	sick-euthyroid	96.68± 0.6	97.06± 0.5	0.76	95.64± 0.6	97.35± 0.5	1.00
8	soybean-large	90.62± 0.9	90.77± 1.1	0.56	91.80± 1.2	92.38± 1.1	0.69
9	Corral	100.00± 0.0	75.00± 3.8	0.00	90.62± 2.6	75.00± 3.8	0.00
10	m-of-n-3-7-10	91.60± 0.9	77.34± 1.3	0.00	86.43± 1.1	77.34± 1.3	0.00
11	Monk1	82.41± 1.8	75.00± 2.1	0.00	71.30± 2.2	75.00± 2.1	0.96
12	Monk2-local	82.41± 1.8	67.13± 2.3	0.00	60.65± 2.3	67.13± 2.3	1.00
13	Monk2	69.68± 2.2	67.13± 2.3	0.13	61.57± 2.3	67.13± 2.3	0.99
14	Monk3	90.28± 1.4	97.22± 0.8	1.00	97.22± 0.8	97.22± 0.8	0.50
	Average real:	84.53	86.67		87.94	87.90	
	Average artif.	86.06	76.47		77.96	76.47	

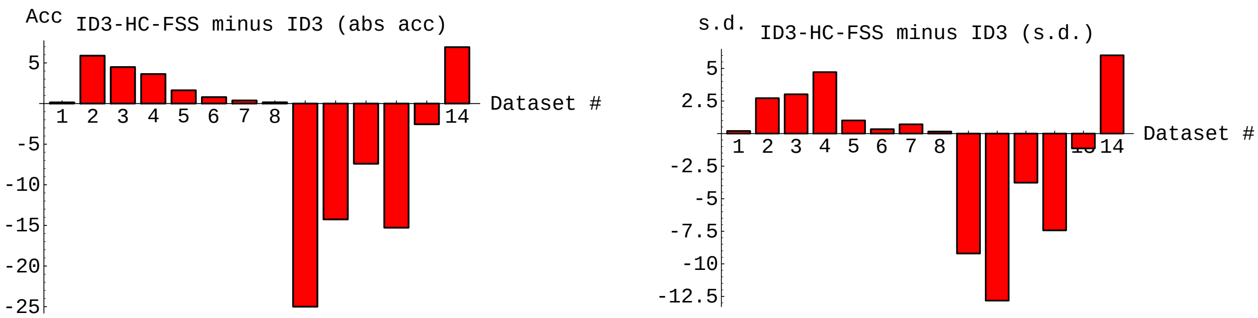

Figure 7: ID3: Absolute difference (FSS minus ID3) in accuracy (left) and in std-devs (right).

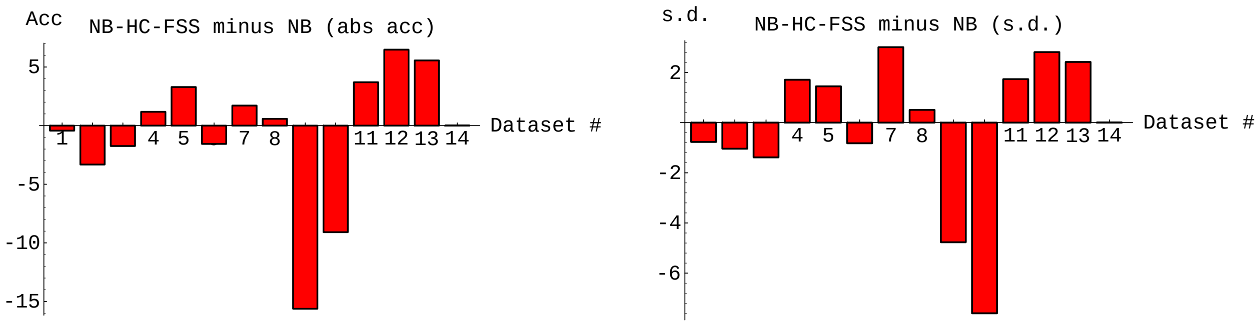

Figure 8: Naive-Bayes: Absolute difference in accuracy (left) and in std-devs (right).

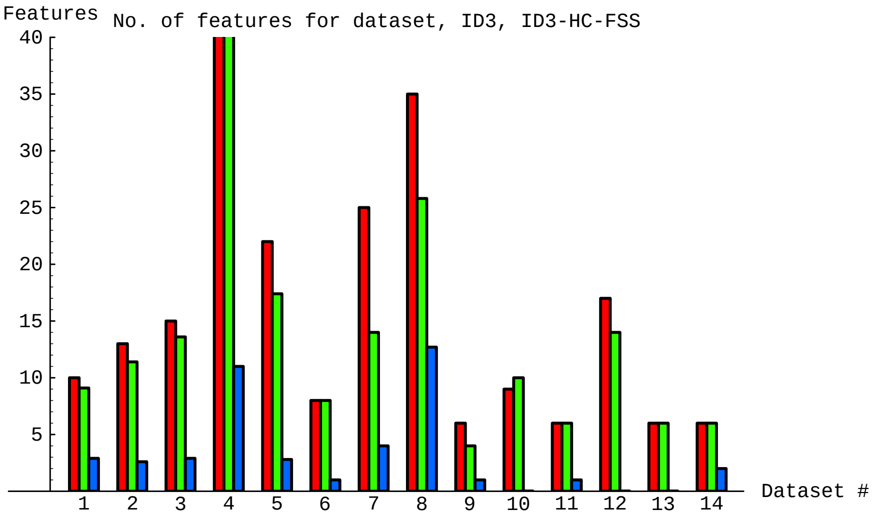

Figure 9: ID3: Number of features in original dataset (left), used by ID3 (middle), and selected by hill-climbing feature subset selection (right). The DNA dataset has 180 features (partially shown).

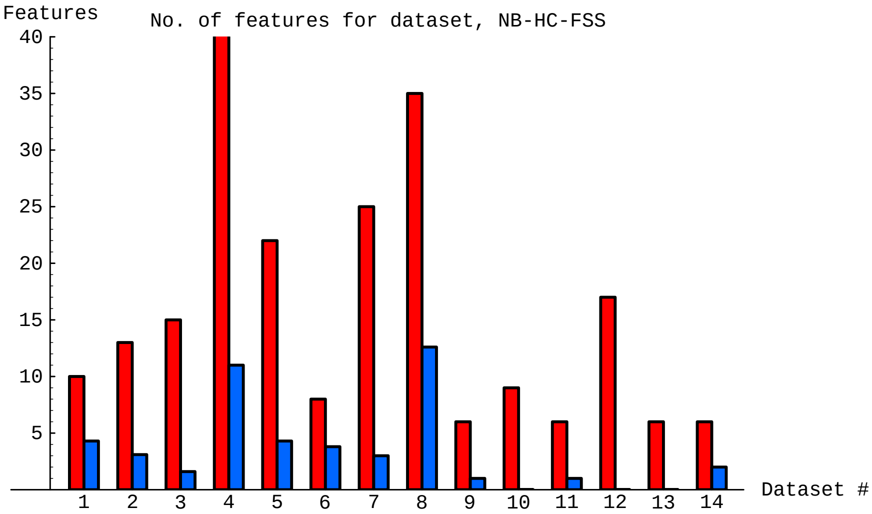

Figure 10: Naive-Bayes: Number of features in original dataset (left) and selected by hill-climbing feature subset selection (right).

Table 5: The number of features in the dataset, the number used by ID3 (since it does some feature subset selection), the number selected by feature subset selection (FSS) for ID3, and the number selected by FSS for Naive-Bayes. Numbers without a decimal point are for single runs, number with a decimal point are averages for the ten-fold cross-validation.

	Dataset	Number of Features			
		Original Dataset	ID3	ID3-FSS	NB-FSS
1	breast cancer	10	9.1	2.9	4.3
2	cleve	13	11.4	2.6	3.1
3	crx	15	13.6	2.9	1.6
4	DNA	180	72	11	11
5	horse-colic	22	17.4	2.8	4.3
6	Pima	8	8.0	1.0	3.8
7	sick-euthyroid	25	14	4	3
8	soybean-large	35	25.8	12.7	12.6
9	Corral	6	4	1	1
10	m-of-n-3-7-10	10	10	0	0
11	Monk1	6	6	1	1
12	Monk2-local	17	14	0	0
13	Monk2	6	6	0	0
14	Monk3	6	6	2	2

The concept for Monk3 is

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

and the training set contains 5% mislabeled instances. The feature subset selection algorithm quickly finds body-shape and jacket-color, which together yield the second conjunction in the above expression, which has accuracy 97.2%. With more features, a larger tree is built which is inferior. This is another example of the optimal feature subset being different than the subset of relevant features.

- For the real datasets and Naive-Bayes, the average accuracy is about same, but very few features are used.
- For the artificial datasets and Naive-Bayes, the average accuracy degrades because of Corral and m-of-n-3-7-10 (the relative error increases by 6.7%). Both of these require a better search than hill climbing can provide. An interesting observation is the fact that the performance on the Monk2 and Monk2-local datasets improves simply by hiding all features, forcing Naive-Bayes to predict the majority class. The independence assumption is so inappropriate for this dataset that it is better to predict the majority class.
- For the DNA dataset, both algorithms selected only 11 features out of 180. While the selected set differed, nine features were the same, indicating that these nine are crucial for both types of inducers.

The results, especially on the artificial datasets where we know what the relevant features are, indicate that the feature subset selection is getting stuck at local maxima too often. The next section deals with improving the search engine.

3.3 A best-first Search Engine

Best-first search (Russell & Norvig 1995, Ginsberg 1993) is a more robust method than hill-climbing. The idea is to select the most promising node we have generated so far that has not already been expanded. Table 6 describes the algorithm, which varies slightly from the standard version because there is no explicit

Table 6: The best-first search algorithm

1. Put the initial state on the OPEN list, CLOSED list $\leftarrow \emptyset$, BEST $\leftarrow$ initial state.
2. Let $v = \arg\max_{w \in \text{OPEN}} f(w)$ (get the state from OPEN with maximal $f(w)$).
3. Remove v from OPEN, add v to CLOSED.
4. If $f(v) - \epsilon > f(\text{BEST})$, then BEST $\leftarrow v$.
5. Expand v: apply all operators to v, giving v's children.
6. For each child not in the CLOSED or OPEN list, evaluate and add to the OPEN list.
7. If BEST changed in the last k expansions, goto 2.
8. Return BEST.

goal condition in our problem. Best-first search usually terminates upon reaching the goal. Our problem is an optimization problem, so the search can be stopped at any point and the best solution found so far can be returned (theoretically improving over time), thus making it an anytime algorithm (Boddy & Dean 1989). In practice, we must stop the run at some stage, and we use what we call a **stale search:** if we have not found an improved node in the last k expansions, we terminate the search. An improved node is defined as a node with an accuracy estimation at least ϵ higher than the best one found so far. In the following experiments, k was set to five and epsilon was 0.1%.

While best-first search is a more thorough search technique, it is not obvious that it is better for feature subset selection. Because of the bias-variance tradeoff (Geman et al. 1992, Kohavi & Wolpert 1996), it is possible that a more thorough search will increase variance and thus reduce accuracy. Quinlan (1995) and Murthy & Salzberg (1995) showed examples where increasing the search effort degraded the overall performance.

Table 7 and Figures 11 and 12 show a comparison of ID3 and Naive-Bayes with hill-climbing feature subset selection and best-first search feature subset selection. Table 8 shows the average number of features used for each algorithm (averaged over the ten folds when relevant). The following observations can be made:

- For the real datasets and both algorithms (ID3 and Naive-Bayes), there is almost no difference between hill climbing and best-first search. Best-first search usually finds a larger feature subset, but the accuracies are approximately the same. The only statistically significant difference is for Naive-Bayes and soybean, where there was a significant improvement with a p-value of 0.95.
- For the artificial datasets, there is a very large improvement for ID3. Performance drastically improves on three datasets (Corral, Monk1, Monk2-local), remains the same on two (m-of-n-3-7-10, Monk3), and degrades on only one (Monk2). Analyzing the selected features, the optimal feature subset was found for Corral, Monk1, Monk2-local, and Monk3 (only two features out of the three relevant ones were selected for Monk3 because this correctly led to better prediction accuracy). The improvement over ID3 without FSS (Table 4) is less dramatic but still positive: the absolute difference in accuracy is 2.49%, which translates into a relative error reduction of 17.8%.

  The search was unable to find the seven relevant features in m-of-n-3-7-10. Because of the complexity penalty of 0.1% for extra features, only subsets of two features were tried, and such subsets never improved over the majority prediction (ignoring all features) before the search was considered stale (five non-improving node expansions). The local maximum where the search stops in this dataset is too large for the current setting of best-first search to overcome. A specific experiment was conducted to determine how long it would take best-first search to find the correct feature subset. The stale limit (originally set to five) was increased until a node better than the node using zero features (predicting

Table 7: A comparison of a hill-climbing search and a best-first search. The first p-val column indicates the probability that best-first search feature subset selection (BFS-FSS) improves hill-climbing feature subset selection (HC-FSS) for ID3 and the second column is analogous but for Naive-Bayes.

	Dataset	ID3		p-val	Naive-Bayes		p-val
		HC-FSS	BFS-FSS		HC-FSS	BFS-FSS	
1	breast cancer	94.71± 0.5	94.57± 0.7	0.41	96.57± 0.6	96.00± 0.6	0.17
2	cleve	78.24± 2.0	79.52± 2.3	0.73	79.56± 3.9	80.23± 3.9	0.57
3	crx	85.65± 1.6	85.22± 1.6	0.39	85.36± 1.6	86.23± 1.0	0.75
4	DNA	94.27± 0.7	94.27± 0.7	0.50	94.52± 0.7	94.60± 0.7	0.55
5	horse-colic	83.15± 1.1	82.07± 1.5	0.21	83.15± 2.0	83.42± 2.0	0.55
6	Pima	69.52± 2.2	68.73± 2.2	0.36	74.34± 2.0	75.12± 1.5	0.67
7	sick-euthyroid	97.06± 0.5	97.06± 0.5	0.50	97.35± 0.5	97.35± 0.5	0.50
8	soybean-large	90.77± 1.1	91.65± 1.0	0.81	92.38± 1.1	93.70± 0.4	0.95
9	Corral	75.00± 3.8	100.00± 0.0	1.00	75.00± 3.8	90.62± 2.6	1.00
10	m-of-n-3-7-10	77.34± 1.3	77.34± 1.3	0.50	77.34± 1.3	77.34± 1.3	0.50
11	Monk1	75.00± 2.1	97.22± 0.8	1.00	75.00± 2.1	72.22± 2.2	0.10
12	Monk2-local	67.13± 2.3	95.60± 1.0	1.00	67.13± 2.3	67.13± 2.3	0.50
13	Monk2	67.13± 2.3	63.89± 2.3	0.08	67.13± 2.3	67.13± 2.3	0.50
14	Monk3	97.22± 0.8	97.22± 0.8	0.50	97.22± 0.8	97.22± 0.8	0.50
	Average real:	86.67	86.64		87.90	88.33	
	Average artif.	76.47	88.55		76.47	78.61	

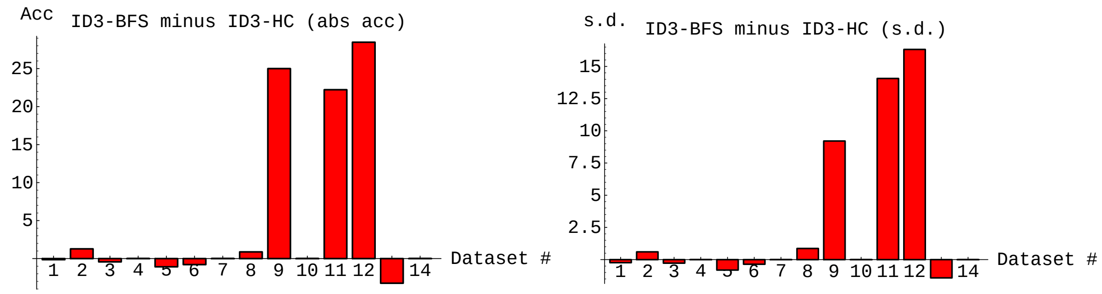

Figure 11: ID3: Absolute difference (best-first search FSS minus hill-climbing FSS) in accuracy (left) and in std-devs (right).

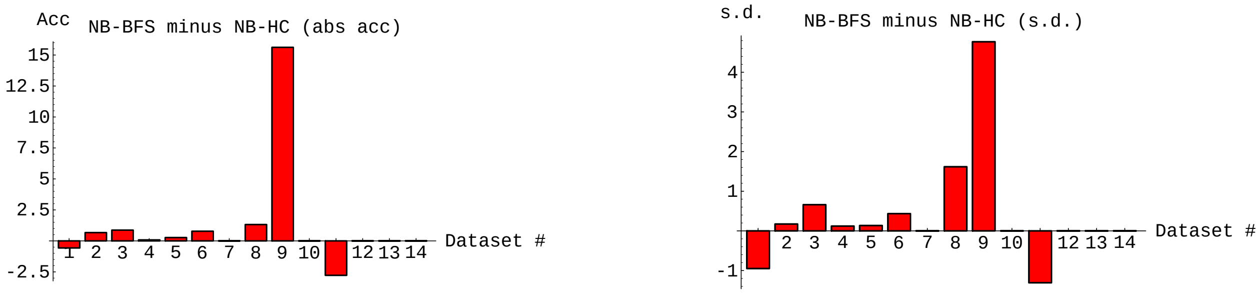

Figure 12: Naive-Bayes: Absolute difference in accuracy (left) and in std-devs (right).

Table 8: The number of features in the dataset, the number used by ID3 (since it does some feature subset selection), the number selected by hill-climbing FSS for ID3, best-first search FSS for ID3, and analogously for Naive-Bayes.

	Dataset	Number of Features					
		Original dataset	ID3	ID3-FSS		NB-FSS	
				HC	BFS	HC	BFS
1	breast cancer	10	9.1	2.9	3.6	4.3	5.2
2	cleve	13	11.4	2.6	3.4	3.1	3.6
3	crx	15	13.6	2.9	3.6	1.6	5.9
4	DNA	180	72	11	11	11	14
5	horse-colic	22	17.4	2.8	3.4	4.3	5.1
6	Pima	8	8.0	1.0	2.3	3.8	4.0
7	sick-euthyroid	25	14	4	4	3	3
8	soybean-large	35	25.8	12.7	13.7	12.6	13.8
9	Corral	6	4	1	4	1	5
10	m-of-n-3-7-10	10	10	0	0	0	0
11	Monk1	6	6	1	3	1	4
12	Monk2-local	17	14	0	6	0	0
13	Monk2	6	6	0	3	0	0
14	Monk3	6	6	2	2	2	2

the majority label value) was found. The first stale setting that overcame the local maximum was 29 (any number above would do). At this setting, a node with three features from the seven is found that is more accurate than majority. Nine more node expansions lead to the correct feature subset. Overall, 193 nodes were evaluated out of the 1024 possibilities. The total running time to find the correct feature subset was 33 CPU minutes, and the prediction accuracy was 100%.

In the Monk2 dataset, a set of three features was chosen, and accuracy significantly degraded compared to hill-climbing, which selected the empty feature subset. This is the only case where performance degraded significantly because best-first search was used (p-value of 0.08). The Monk2 concept in this encoding is unsuitable for decision trees, as a correct tree (built from the full space) contains 439 nodes and 296 leaves. Because the standard training set contains only 169 instances, it is impossible to build the correct tree using the standard recursive partitioning techniques.

- For the artificial datasets, there was a significant improvement for Naive-Bayes only for Corral (p-value of 1.00), and performance significantly degraded for Monk1 (p-value of 0.10). The rest of the datasets were unaffected.

  The chosen feature subset for Corral contained features A_0, A_1, B_0, B_1, and the "correlated" feature. It is known that only the first four are needed, yet because of the limited representation power of the Naive-Bayes, performance using the "correlated" feature is better than performance using only the first four features. If Naive-Bayes is given access only to the first four features, the accuracy degrades from 90.62% to 87.50%. This dataset is one example where the optimal feature subset for different induction algorithms is known to be different. Decision trees are hurt by the addition of the "correlated" feature (performance degrades), yet Naive-Bayes improves with this feature.

  The Monk1 dataset degrades in performance because the features head-shape, body-shape, is-smiling, and jacket-color were chosen, yet performance is better if only jacket-color is used. Note that both head-shape and body-shape are part of the target concept, yet the representation power of Naive-Bayes is again limited and cannot utilize this information well. As with the Monk2 dataset for ID3, this may be an example of the search overfitting in the sense that some subset seems to slightly improve the accuracy estimation, but not the accuracy on the independent test set (see Section 6 for further discussion on issues of overfitting).

  The datasets m-of-n-3-7-10, Monk2-local, Monk2, and Monk3, all had the same accuracy with best-first search as with hill-climbing. The performance of Naive-Bayes on the Monk3 dataset cannot be improved by using a different feature subset. As with ID3, the search was unable to find a good feature subset for m-of-n-3-7-10 (the correct feature subset allows improving the accuracy to 87.5%). For the Monk2 and Monk2-local datasets, the optimal feature subset is indeed the empty set! Naive-Bayes on the set of relevant features yields inferior performance to a majority inducer, which is how Naive-Bayes behaves on the empty set of features.

While best-first search generally gives better performance than hill-climbing, high-level interactions occurring in m-of-n-3-7-10 cannot be caught with a search that starts at the empty feature subset unless the stale parameter is drastically increased. An alternative approach to forward selection tested here is backward elimination, which suffers less from feature interaction because it starts with the full set of features; however, the running time would make the approach infeasible in practice, especially if there are many features.

The running times for the best-first search starting from the empty set of features range from about 5-10 minutes of CPU time for small problems such as Monk1, Monk2, Monk3, and Corral, to 15 hours for DNA. In the next section, we attempt to reorder the search space dynamically to allow the search to reach better nodes faster and make the backward feature subset selection feasible.

4 The State Space: Compound Operators

If we try to gild the lily by using both options together...
—Quinlan (1993)

In the previous section, we looked at two search engines. In this section, we look at the topology of the state space and dynamically modify it based on accuracy estimation results. As previously described, the state space is commonly organized such that each node represents a feature subset, and each operator represents the addition or deletion of a feature. The main problem with this organization is that the search must expand (*i.e.*, generate successors of) every node on the path from the initial feature subset to the best feature subset. This section introduces a new way to change the search space topology by creating dynamic operators that directly connect a node to nodes considered promising given the evaluation of its children. These operators better utilize the information available in the evaluated children.

The motivation for compound operators comes from Figure 13, which partitions the feature subsets into strongly relevant, weakly relevant, and irrelevant features. In practice, an optimal feature subset is likely to contain only relevant features (strongly and weakly relevant features). A backward elimination search starting from the full set of features (as depicted in Figure 13) and that removes one feature at a time after expanding all children reachable using one operator, will have to expand all the children of each node before removing a single feature. If there are i irrelevant features and f features, $(i \cdot f)$ nodes must be evaluated. Similar reasoning applies to forward selection search starting from the empty set of features. In domains where feature subset selection might be most useful, there are many features but such a search may be prohibitively expensive.

Compound operators are operators that are dynamically created *after* the standard set of children (created by the add and delete operators) has been evaluated. They are used for a single node expansion and then discarded. Intuitively, there is more information in the evaluation of the children than just the identification of the node with the maximum evaluation. Compound operators combine operators that led to the best children into a single dynamic operator. Figure 14 depicts a possible set of compound operators for forward selection. The root node containing no features was expanded by applying four add operators, each one adding a single feature. The operators that led to $0, 1, 0, 0$ and $0, 0, 1, 0$ were combined into the first compound operator (shown in a dashed line going left) because they led to the two nodes with the highest evaluation (evaluation not shown). If the first compound operator led to a node with an improved estimate, the second compound operator (shown in a dashed line going right) is created that combines the best three original operators, *etc.*

Formally, if we rank the operators by the estimated accuracy of the children, then we can define the **compound operator** c_i to be the combination of the best $i+1$ operators. For example, the first compound operator will combine the best two operators. If the best two operators each added a feature, then the first compound operator will add both; if one operator added and one operator deleted, then we try to do both in one operation. The compound operators are applied to the parent, thus creating children nodes that are

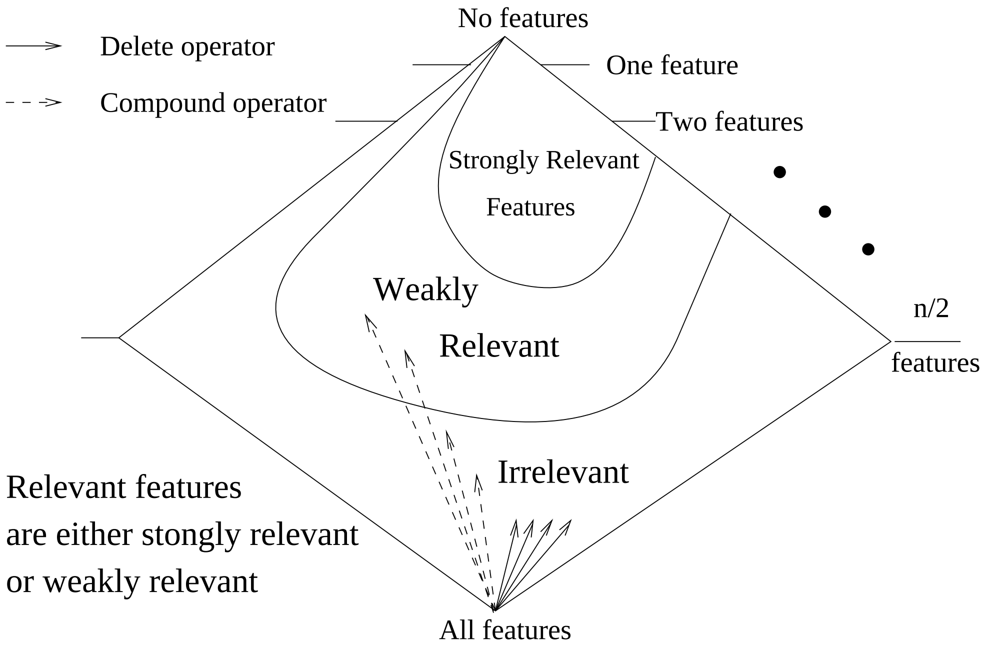

Figure 13: The feature subset state space divided into irrelevant, weakly relevant, and strongly relevant feature subsets. The dotted arrows indicate compound operators.

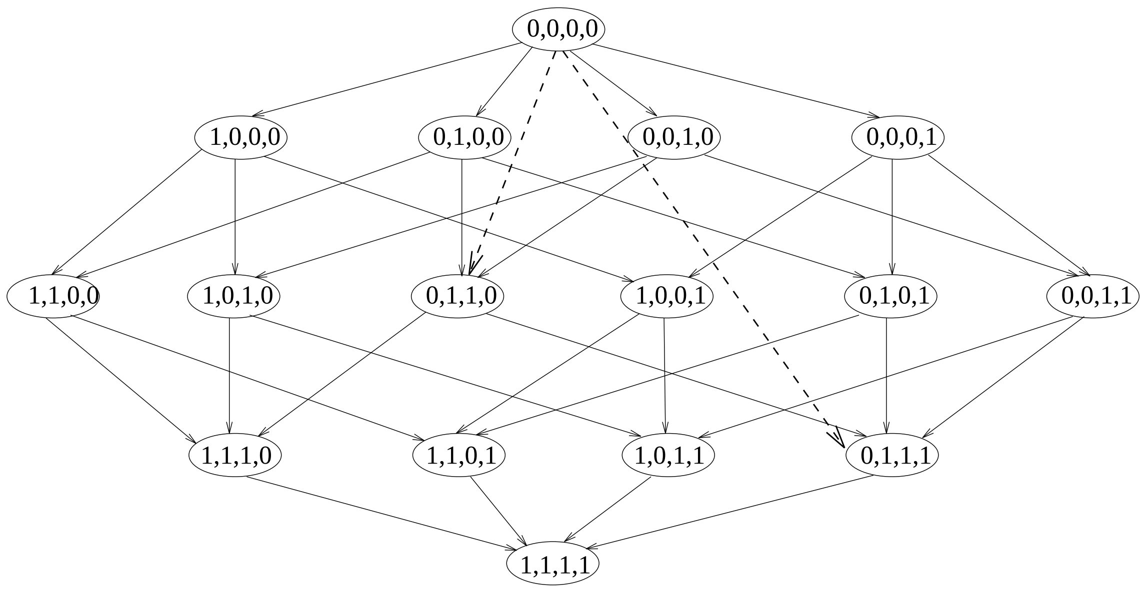

Figure 14: The state space search with dotted arrows indicating compound operators. From the root's children, the nodes (0,1,0,0) and (0,0,1,0) had the highest evaluation values, followed by (0,0,0,1).

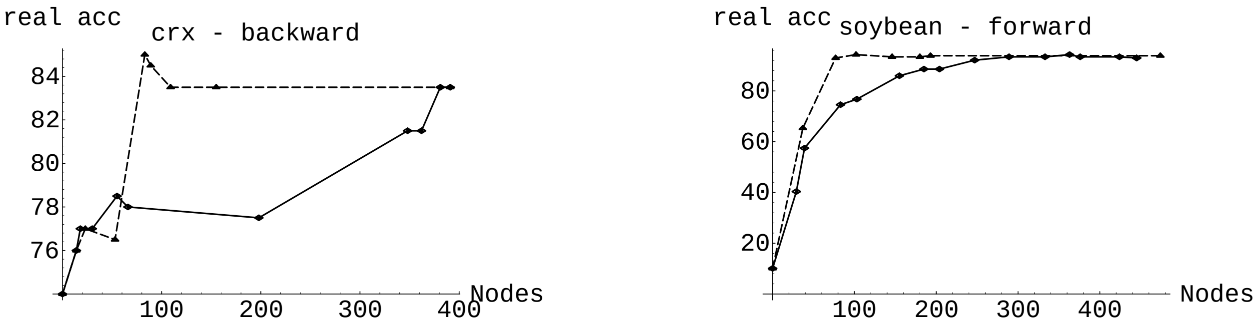

Figure 15: Comparison of compound (dotted line) and non-compound (solid line) searches. The accuracy (y-axis) is that of the best node (as determined by the algorithm) on an independent test set after a given number of node evaluations (x-axis). The running time is proportional to the number of nodes evaluated.

farther away in the state space. Each compound node is evaluated and the generation of compound operators continues as long as the estimated accuracy of the compound nodes improves.

Compound operators generalize a few existing approaches. Kohavi (1994) suggested that the search might start from the set of strongly relevant features. If one starts from the full set of features, removal of any single strongly relevant feature will cause a degradation in performance, while removal of any irrelevant or weakly relevant feature will not. Since the last compound operator, representing the combination of all delete operators, connects the full feature subset to the empty set of features, the compound operators from the full feature subset plot a path through the strongly relevant feature sets. The path is explored by removing one feature at a time until estimated accuracy deteriorates, thus generalizing the original proposal. Caruana & Freitag (1994) implemented SLASH, a version of feature subset selection that eliminates the features not used in the derived decision tree. If there are no features that improve the performance when deleted, then (ignoring orderings due to ties) one of the compound operators will lead to the same node that SLASH would take the search to. While the SLASH approach is only applicable to backward elimination, compound operators are also applicable to forward selection.

Figure 15 shows two searches with and without compound operators. Compound operators improve the search by finding nodes with higher accuracy faster; however, whenever it is easy to overfit (*e.g.*, for small datasets), they cause overfitting earlier (see Section 6). Experimental accuracies using compound operators are similar to those without them and the runs are usually faster. More significant time differences are achieved when the decision trees are pruned. Detailed results for that case are shown later in the paper (Table 11).

The main advantage of compound operators is that they make backward feature subset selection computationally feasible. Table 9 and Figures 16 and 17 show the results of running the best-first search algorithm with compound operators but starting with the full set of features (backward elimination) compared with best-first search forward selection without compound operators. Accuracy results for forward selection with and without compound operators did not significantly differ on any dataset. Table 10 shows the number of features used for each of the different methods. When one starts from the full set of features, feature interactions are easier for the search to identify. The following observations can be made:

- Except for m-of-n-3-7-10, the accuracy results for backward FSS with ID3 generally degraded. The main improvement was for m-of-n-3-7-10, where the correct seven bits were correctly identified, resulting in 100% accuracy. The feature subsets were generally larger, and apparently even best-first search cannot overcome some local maxima with our stale parameter setting. For example, the run on DNA stopped with 36 features, but pruning more features would improve the performance because the forward search found a subset of 11 features that was significantly better (the accuracy estimation for the 11 feature subset was higher than the one for the 36 feature subset, and because the same folds are used, if the best-first search were to get to this 11-feature node, it would prefer it over the final node selected in the backward search). In the next section, we use the backward search with C4.5. Because C4.5 prunes, the backward search is then more efficient with the best-first search algorithm.

Table 9: A comparison of a forward best-first search without compound operators and backward best-first search with compound operators. The p-val columns indicates the probability that backward is better than forward.

	Dataset	ID3		p-val	Naive-Bayes		p-val
		BFS-FSS forward	BFS-FSS back		BFS-FSS forward	BFS-FSS back	
1	breast cancer	94.57± 0.7	93.85± 0.5	0.11	96.00± 0.6	96.00± 0.6	0.50
2	cleve	79.52± 2.3	75.89± 3.7	0.12	80.23± 3.9	82.56± 2.5	0.76
3	crx	85.22± 1.6	83.33± 1.5	0.10	86.23± 1.0	84.78± 0.8	0.05
4	DNA	94.27± 0.7	91.23± 0.8	0.00	94.60± 0.7	96.12± 0.6	0.99
5	horse-colic	82.07± 1.5	82.61± 1.7	0.63	83.42± 2.0	82.33± 1.3	0.26
6	Pima	68.73± 2.2	67.44± 1.4	0.24	75.12± 1.5	76.03± 1.6	0.72
7	sick-euthyroid	97.06± 0.5	97.06± 0.5	0.50	97.35± 0.5	97.35± 0.5	0.50
8	soybean-large	91.65± 1.0	91.35± 1.0	0.38	93.70± 0.4	94.29± 0.9	0.81
9	Corral	100.00± 0.0	100.00± 0.0	0.50	90.62± 2.6	90.62± 2.6	0.50
10	m-of-n-3-7-10	77.34± 1.3	100.00± 0.0	1.00	77.34± 1.3	87.50± 1.0	1.00
11	Monk1	97.22± 0.8	97.22± 0.8	0.50	72.22± 2.2	72.22± 2.2	0.50
12	Monk2-local	95.60± 1.0	95.60± 1.0	0.50	67.13± 2.3	67.13± 2.3	0.50
13	Monk2	63.89± 2.3	64.35± 2.3	0.58	67.13± 2.3	67.13± 2.3	0.50
14	Monk3	97.22± 0.8	97.22± 0.8	0.50	97.22± 0.8	97.22± 0.8	0.50
	Average real:	86.64	85.35		88.33	88.68	
	Average artif.	88.55	92.40		78.61	80.30	

- For Naive-Bayes, backward FSS performs slightly better in terms of accuracy. Only on crx did the accuracy degrade significantly (p-val=0.05), while on m-of-n-3-7-10 and DNA it significantly improved (p-val=1.00 and 0.99 respectively). In fact, for the DNA dataset, no other known algorithm outperformed Naive-Bayes on the selected feature subset. Taylor et al. (1994, page 159) compared 23 algorithms on this dataset (with the same training and test sets), and the best was RBF (radial basis functions) using 720 centers with an accuracy of 95.9%. The Naive-Bayes algorithm with backward elimination had an accuracy of 96.12%.

- The m-of-n-3-7-10 dataset with Naive-Bayes is a very interesting case. The feature subset selection finds six out of the seven relevant features, and the seventh selected feature is an irrelevant one. Although m-of-n can be represented using a hyperplane, and although in a Boolean domain the surface represented by Naive-Bayes is always a hyperplane, it turns out that Naive-Bayes is unable to learn this target concept. The table below was constructed by giving Naive-Bayes all possible instances and their correct classification for the 3-of-7 concept, and testing it on the same instances. We can see that Naive-Bayes is unable to learn 3-of-7, but what is intriguing is that fact that hiding one bit (feature) improves the accuracy.

Features given	Naive-Bayes accuracy	Perceptron accuracy
7 (all)	83.59	100.00
6	88.28	88.28
5	82.03	82.03

The explanation for this result is as follows. There are $\binom{7}{0} + \binom{7}{1} + \binom{7}{2} = 29$ instances out of $2^7 = 128$ that have label 0. There are $\binom{7}{1} + \binom{7}{2} \cdot 2 = 49$ ones in these 29 instances, so each of the seven features has $49/7 = 7$ ones. We thus get the following:

$$p(Y = 0 \mid X_i = 1) = 7/29$$
$$p(Y = 0 \mid X_i = 0) = 22/29$$

Figure 16: ID3: Absolute difference (best-first search FSS backward with compound operators minus forward) in accuracy (left) and in std-devs (right).

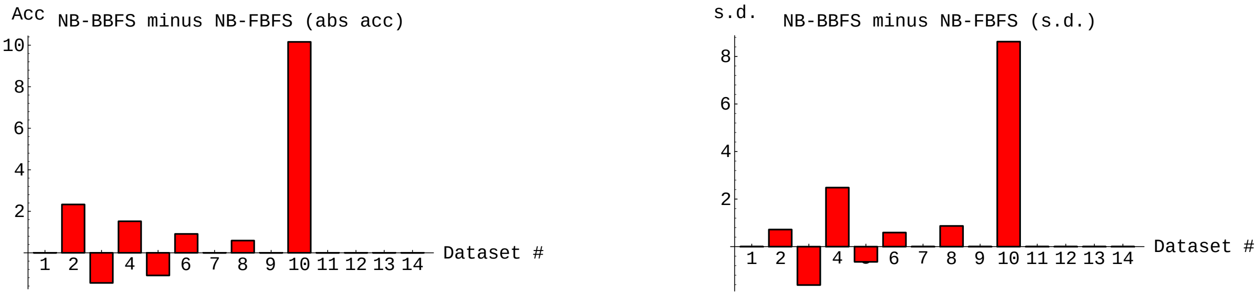

Figure 17: Naive-Bayes: Absolute difference in accuracy (left) and in std-devs (right).

Similarly, $\sum_{i=3}^{7} \binom{7}{i} * i = 399$, thus each of the seven features has $399/7 = 57$ ones, giving the following:

$$p(Y = 1 \mid X_i = 1) = 57/99$$
$$p(Y = 1 \mid X_i = 0) = 42/99$$

If there are only two ones in an instance, the probabilities computed by Naive-Bayes are:

$$p(Y = 0) \propto 29/128 \cdot (7/29)^2 \cdot (22/29)^5 = 0.00331674$$
$$p(Y = 1) \propto 99/128 \cdot (57/99)^2 \cdot (42/99)^5 = 0.00352351$$

giving the label "one" a small advantage, and making the wrong prediction. Thus there are $\binom{7}{2} = 21$ mistakes out of the 128 possible instances, which is exactly 83.59% accuracy.

With only six features, the best thing to do is to predict a label of one when there two "on" bits, which is what the Naive-Bayes does (the calculation is omitted). This will correctly capture all instances that originally had three bits, but will continue to be wrong for those instances that had only two bits. However, out of the 21 instances that had two bits on, six will now have only one bit on because there were 42 bits total, and each of the seven bits had a one six times. Thus Naive-Bayes will now make only $21 - 6 = 15$ mistakes, which yields an accuracy of 88.28%.

This example shows that although the hypothesis space for Naive-Bayes in Boolean domains is a space of hyperplanes, it is unable to correctly identify this target concept, while a Perceptron can. More interesting, however, is the fact that any approach to feature subset selection based on relevance that is independent of the induction algorithm and that ranks each feature independently (conditioned on the label) must give the same rank to each one of the seven relevant features (due to symmetry), and thus such an approach will never pick a subset of six features as the wrapper approach does. The wrapper approach indeed finds the optimal subset for this target concept.

Table 10: The number of features in the dataset, the number used by ID3 (since it does some feature subset selection), the number selected by best-first search FSS for ID3 forward without compound and backwards with compound, and analogously for Naive-Bayes.

	Dataset	Number of Features					
		Original dataset	ID3	ID3-FSS		NB-FSS	
				Forward	Backward	Forward	Backward
1	breast cancer	10	9.1	3.6	5.3	5.2	5.9
2	cleve	13	11.4	3.4	4.6	3.6	7.9
3	crx	15	13.6	3.6	7.7	5.9	9.1
4	DNA	180	72	11	36	14	48
5	horse-colic	22	17.4	3.4	7.2	5.1	6.1
6	Pima	8	8.0	2.3	5.7	4.0	4.4
7	sick-euthyroid	25	14	4	4	3	3
8	soybean-large	35	25.8	13.7	17.7	13.8	16.7
9	Corral	6	4	4	4	5	5
10	m-of-n-3-7-10	10	10	0	7	0	7
11	Monk1	6	6	3	3	4	4
12	Monk2-local	17	14	6	6	0	5
13	Monk2	6	6	3	3	0	0
14	Monk3	6	6	2	2	2	2

Running times for the backward feature subset selection were about five times longer than the forward, which is not bad considering the fact that we started with the full set of features (also see the next section where compound operators help more when C4.5 is used).

5 Global Comparison

We have used ID3 and Naive-Bayes as our basic inducers for feature subset selection because they do no pruning and, therefore, the effect of feature subset selection can be seen more clearly. We have seen improvements in both algorithms, but an important remaining question is how the wrapper algorithm developed in Sections 3 and 4 compares to the filter approach, and how the feature subset selection versions of these algorithms compare to the original versions. Although we have presented arguments in favor of the wrapper approach in Section 2, we had to develop a high-performance wrapper algorithm for the empirical comparisons, and this was the purpose of the preceding sections. When used with C4.5, the hill-climbing wrapper often gets stuck in local minima, and the best-first search wrapper took too long, so the work in the previous sections was necessary for the experiments in this section.

With compound operators, running the wrapper with C4.5 tends to be even faster than running the wrapper with ID3 because the compound operators tend to quickly remove the features pruned by C4.5. Features that do not appear in the tree are removed because the accuracy estimate does not change and, with the small complexity penalty for every feature, the evaluation function improves. The compound operators can remove all such features after a single node expansion. Without pruning, many more features are used in the tree and they cause slight random variations in the accuracy estimates. It hence makes more sense to run the feature subset selection search backwards, which is what we have done. Figures 18 and 19 show how the number of features used changes as the search progresses, *i.e.*, as more nodes are evaluated. Notice how before each node expansion, the compound operators are applied and combine the operators leading to the best children, thus drastically decreasing the number of nodes. Without compound operators, the number of features could only decrease or increase by one at every node expansion. For example, in the DNA dataset with C4.5, "only" 3555 nodes were evaluated and a subset of 12 features was selected; without compound operators, the algorithm would have to expand $(180 - 12) \cdot 180 = 30,240$ nodes just to get to this feature subset.

Backward FSS with C4.5 is still very slow, but generally faster than backward FSS with ID3. Table 11

Table 11: The CPU time for different versions of the wrapper approach. Time is for a single fold when cross-validation was done in an outer loop to estimate accuracy. All tests used compound operators, except for ID3-FSS-Forward. The "time" command overflowed for ID3-FSS-back on DNA under Sun's Solaris operating system. The command gave a negative number for execution time!

	CPU time (seconds)			
	ID3-FSS	ID3-FSS	C4.5-FSS	NB-FSS
Dataset	Forward	Backward	Backward	Backward
breast cancer	439	741	1,167	51
cleve	746	2,105	816	123
crx	936	4,076	1,658	206
DNA	42,908	overflow	165,621	88,334
horse-colic	1,067	2,875	1,434	462
Pima	963	2,178	719	57
sick-euthyroid	3,764	12,166	7,386	504
soybean-large	8,544	4,196	3,931	2,033
Corral	165	26	47	4
m-of-n-3-7-10	213	179	223	55
Monk1	128	57	75	15
Monk2-local	1,466	574	644	139
Monk2	247	90	81	18
Monk3	111	55	46	9

shows the running time for different versions of the algorithms; compared to the original algorithm, they are about two to three orders of magnitude slower. For example, running C4.5 on the DNA dataset takes about 1.5 minutes. The wrapper model has to run C4.5 five times for every node that is evaluated in the state space and in DNA there are hundreds of nodes.

We shall investigate two hypotheses: first, that using a filter method will sometimes improve the accuracy of ID3 and Naive-Bayes on real datasets but will be fairly erratic (often hurting performance), and second, that improvements from using the wrapper approach will surpass the gains from the filter and will be more consistent. As a representative of the filter methods, we chose the Relieved-F algorithm (Section 2.4.2), which seemed to have the most desirable properties among the filter algorithms discussed. For the reasons outlined in the preceding paragraphs, we use the backward best-first-search wrapper with compound operators as a representative of wrapper algorithms. The experimental methodology used to run and compare algorithms is the same as described in Section 3.1.

Since C4.5 is a modern algorithm that performs well on a variety of real databases, we might expect it to be difficult to improve upon its performance using feature selection. Table 12 shows that this is the case: overall, the accuracy on real datasets actually decreased when using Relieved-F, but the accuracy slightly increased using the wrapper (a 5.5% relative reduction in error). Note however that Relieved-F did perform well on some artificial databases, all of which (except for Corral) contain only strongly relevant and totally irrelevant attributes. On three artificial datasets, Relieved-F was significantly better than plain C4.5 at the 99% confidence level. On the real datasets, where relevance is ill-determined, Relieved-F often did worse than plain C4.5: on one dataset its performance was significantly worse at the 99% confidence level, and in no case was its performance better at even the 90% confidence level. The wrapper algorithm did significantly better than plain C4.5 on two real databases and two artificial databases, and was never significantly worse. Note that the most significant improvement on a real database was on the one real dataset with many features: DNA. Relieved-F was outperformed by the wrapper significantly on two real datasets, but it outperformed the wrapper on the m-of-n-3-7-10 dataset.

On the Corral dataset, the wrapper selected the correct features {A1, A2, B1, B2} as the best node early in the search, but later settled on only the features A1 and A2, which gave better cross-validation accuracy. The training set is very small (32 instances), so the problem was that even though the wrapper gave the ideal feature set to C4.5, it built the correct tree (100% accurate) but then pruned it back because according

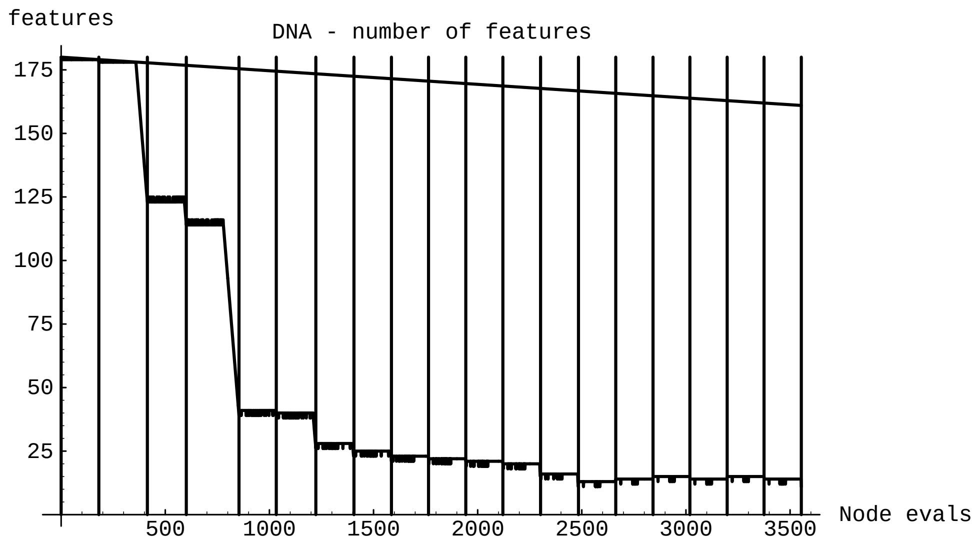

Figure 18: DNA: Number of features evaluated as the search progresses (C4.5, best-first search, backward). The vertical lines signify a node expansion, where the children of the best node are expanded. The slanted line on the top shows how ordinary backward selection would progress.

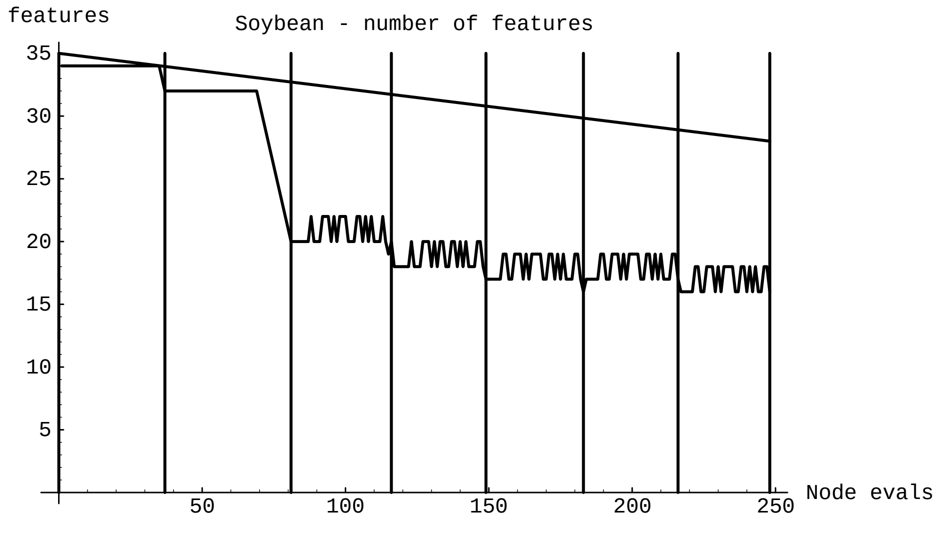

Figure 19: Soybean: Number of features evaluated as the search progresses (C4.5, best-first search, backward).

Table 12: A comparison of C4.5 with no feature selection, with the Relieved-F filter (RLF), and with the wrapper using backward best-first search with compound operators (BFS). The p-val columns indicates the probability that the top algorithm is improving over the lower algorithm.

Dataset	C4.5	C4.5-RLF	C4.5-BFS	C4.5-RLF vs C4.5	C4.5-BFS vs C4.5	C4.5-BFS vs C4.5-RLF
breast cancer	95.42± 0.7	94.42± 1.1	95.28± 0.6	0.14	0.41	0.83
cleve	72.30± 2.2	74.95± 3.1	77.88± 3.2	0.84	**0.98**	0.82
crx	85.94± 1.4	84.06± 1.2	85.80± 1.3	0.07	0.46	0.91
DNA	92.66± 0.8	92.75± 0.8	94.44± 0.7	0.54	**0.99**	**0.99**
horse-colic	85.05± 1.2	85.88± 1.0	84.77± 1.3	0.77	0.41	0.17
Pima	71.60± 1.9	64.18± 2.3	70.18± 1.3	**0.00**	0.19	**1.00**
sick-euthyroid	97.73± 0.5	97.73± 0.5	97.91± 0.4	0.50	0.65	0.65
soybean-large	91.35± 1.6	91.35± 1.6	91.93± 1.3	0.50	0.65	0.65
Corral	81.25± 3.5	81.25± 3.5	81.25± 3.5	0.50	0.50	0.50
m-of-*n*-3-7-10	85.55± 1.1	91.41± 0.9	85.16± 1.1	**1.00**	0.36	**0.00**
Monk1	75.69± 2.1	88.89± 1.5	88.89± 1.5	**1.00**	**1.00**	0.50
Monk2-local	70.37± 2.2	88.43± 1.5	88.43± 1.5	**1.00**	**1.00**	0.50
Monk2	65.05± 2.3	67.13± 2.3	67.13± 2.3	0.82	0.82	0.50
Monk3	97.22± 0.8	97.22± 0.8	97.22± 0.8	0.50	0.50	0.50
Average real:	86.51	85.67	87.27			
Average artif.	79.19	85.72	84.68			

to its pruning criterion the training set data was insufficient to warrant such a large tree.

Perhaps surprisingly, the Naive-Bayes algorithm turned out to be more difficult to improve using feature selection (Table 13). Both the filter and wrapper approaches significantly degraded performance on the breast cancer and crx databases. In both cases the wrapper approach chose feature subsets with high estimated accuracy that turned out to be poor performers on the real test data. The filter caused significantly worse performance in one other dataset, Pima diabetes, and never significantly improved on plain Naive-Bayes, even on the artificial datasets. This is partly due to the fact that the severely restricted hypothesis space of Naive-Bayes prevents it from doing well on the artificial problems (except for Monk3) for reasons discussed in Section 2.3, and partly because Naive-Bayes' accuracy is hurt more by conditional dependence between features than the presence of irrelevant features.

In contrast, the wrapper approach significantly improved performance on five databases over the plain Naive-Bayes accuracy. In the Monk2 dataset it did so by discarding all features! Because the conditional independence assumption is violated, one actually obtains better performance with Naive-Bayes by throwing out all features and using only the marginal probability distribution over the classes (*i.e.*, always predict the majority class). The wrapper approach significantly improved over the filter in six cases, and was never significantly outperformed by the filter approach.

Table 14 shows similar results with ID3. In this case, the filter approach significantly degraded performance on one real dataset but significantly improved all of the artificial datasets except for Monk2, as did the wrapper approach. The Monk2 concept is exactly-2-of-6, so all features are relevant. Relieved-F judged two features to be irrelevant (due to poor statistics from the small training set) and the wrapper's internal cross-validation gave an overly pessimistic estimate to the node representing the subset of all features, which was optimal. Note that our "ID3" is actually the C4.5 algorithm with command line arguments specifying no pruning. As it happens, command line arguments cannot turn off a tree postprocessing step that can swap a parent decision node with its child, and this swapping results in 100% accuracy on Corral with "plain ID3." The wrapper significantly outperformed the filter on two of the real datasets, but performed significantly worse than the filter on the Monk3 dataset. In Monk3, the feature subset search did test the node with 100% test-set accuracy, but the internal cross-validation estimated its accuracy to be lower than the node with 97.22% test-set accuracy.

We have focused only on accuracy above, so other criteria merit some consideration. First, the wrapper

Table 13: A comparison of Naive-Bayes (NB) with no feature selection, with the Relieved-F filter (RLF), and with the wrapper using backward best-first search with compound operators (BFS). The p-val columns indicates the probability that the top algorithm is improving over the lower algorithm.

Dataset	NB	NB-RLF	NB-BFS	NB-RLF vs NB	NB-BFS vs NB	NB-BFS vs NB-RLF
breast cancer	97.00± 0.5	95.14± 1.3	96.00± 0.6	**0.03**	**0.04**	0.80
cleve	82.88± 2.3	82.53± 2.4	82.56± 2.5	0.44	0.45	0.50
crx	87.10± 0.8	85.51± 0.8	84.78± 0.8	**0.02**	**0.00**	0.18
DNA	93.34± 0.7	93.25± 0.7	96.12± 0.6	0.45	**1.00**	**1.00**
horse-colic	79.86± 2.6	80.95± 2.3	82.33± 1.3	0.67	0.89	0.77
Pima	75.90± 1.8	64.57± 2.4	76.03± 1.6	**0.00**	0.53	**1.00**
sick-euthyroid	95.64± 0.6	95.64± 0.6	97.35± 0.5	0.50	**1.00**	**1.00**
soybean-large	91.80± 1.2	91.65± 1.2	94.29± 0.9	0.45	**0.99**	**0.99**
Corral	90.62± 2.6	90.62± 2.6	90.62± 2.6	0.50	0.50	0.50
m-of-n-3-7-10	86.43± 1.1	85.94± 1.1	87.50± 1.0	0.33	0.85	0.93
Monk1	71.30± 2.2	72.22± 2.2	72.22± 2.2	0.66	0.66	0.50
Monk2-local	60.65± 2.4	63.43± 2.3	67.13± 2.3	0.88	**1.00**	**0.95**
Monk2	61.57± 2.3	63.43± 2.3	67.13± 2.3	0.79	**0.99**	**0.95**
Monk3	97.22± 0.8	97.22± 0.8	97.22± 0.8	0.50	0.50	0.50
Average real:	87.94	86.16	88.68			
Average artif.	77.96	78.81	80.30			

Table 14: A comparison of ID3 with no feature selection, with the Relieved-F filter (RLF), and with the wrapper using backward best-first search with compound operators (BFS). The p-val columns indicates the probability that the top algorithm is improving over the lower algorithm.

Dataset	ID3	ID3-RLF	ID3-BFS	ID3-RLF vs ID3	ID3-BFS vs ID3	ID3-BFS vs ID3-RLF
breast cancer	94.57± 0.9	93.57± 1.5	93.85± 0.5	0.21	0.16	0.60
cleve	72.35± 2.3	72.96± 2.1	75.89± 3.7	0.61	0.87	0.83
crx	81.16± 1.4	78.70± 1.4	83.33± 1.5	**0.04**	0.93	**1.00**
DNA	90.64± 0.9	91.57± 0.8	91.23± 0.8	0.86	0.76	0.34
horse-colic	81.52± 2.0	81.52± 1.3	82.61± 1.7	0.50	0.72	0.76
Pima	68.73± 2.5	63.91± 2.1	67.44± 1.4	0.02	0.26	**0.98**
sick-euthyroid	96.68± 0.6	96.78± 0.5	97.06± 0.5	0.57	0.75	0.71
soybean-large	90.62± 0.9	90.19± 0.9	91.35± 1.0	0.32	0.78	0.89
Corral	100.00± 0.0	100.00± 0.0	100.00± 0.0	0.50	0.50	0.50
m-of-n-3-7-10	91.60± 0.9	100.00± 0.0	100.00± 0.0	**1.00**	**1.00**	0.50
Monk1	82.41± 1.8	97.22± 0.8	97.22± 0.8	**1.00**	**1.00**	0.50
Monk2-local	82.41± 1.8	95.60± 1.0	95.60± 1.0	**1.00**	**1.00**	0.50
Monk2	69.68± 2.2	63.90± 2.3	64.35± 2.3	**0.01**	**0.01**	0.58
Monk3	90.28± 1.4	100.00± 0.0	97.22± 0.8	**1.00**	**1.00**	0.00
Average real:	84.53	83.65	85.34			
Average artif.	86.06	92.79	92.39			

Table 15: The number of features in each dataset, the number selected by Relieved-F, the number used by the plain versions of the algorithms, and the number used by the wrapped versions using backward best-first search with compound operators (BFS).

Dataset	All	RLF	C4.5	C4.5-BFS	NB-BFS	ID3	ID3-BFS
breast cancer	10	5.7	7.0	3.9	5.9	9.1	5.3
cleve	13	10.5	9.1	5.3	7.9	11.4	4.6
crx	15	11.5	9.9	7.7	9.1	13.6	7.7
DNA	180	178	46	12	48	72	36
horse-colic	22	18.2	5.5	4.3	6.1	17.4	7.2
Pima	8	1.2	8.0	4.8	4.4	8.0	5.7
sick-euthyroid	25	24	4	3	3	14	4
soybean-large	35	34.8	22.0	17.1	16.7	25.8	17.7
Corral	6	5	4	2	5	4	4
m-of-n-3-7-10	10	7	9	6	7	10	7
Monk1	6	3	5	3	4	6	3
Monk2-local	17	8	12	6	5	14	6
Monk2	6	4	6	0	0	6	3
Monk3	6	3	2	2	2	6	2
Average Reduction		30%	37%	40%	28%	6%	19%

method extends directly to minimizing misclassification cost. Most Irvine datasets do not include cost information and so accuracy is a natural performance metric, but one can trivially use a cost function instead of accuracy as the evaluation function for the wrapper. For filter approaches, adapting to misclassification costs is a research topic. Second, we should compare the number of features selected by the filter and wrapper. Table 15 shows the number of features in each dataset, the number selected by the Relieved-F filter (note that since the filter is independent of the induction algorithm, it prescribes the same set of features whether using ID3, C4.5, or Naive-Bayes), and the number selected by the plain versions of the algorithms and their wrapper-enhanced versions. (Plain Naive-Bayes always uses all features, so it does not have its own column.) The average reduction column shows the average percentage decrease in number of features between each column and its natural benchmark (*e.g.*, RLF and C4.5 are compared to the original dataset, C4.5-BFS is compared to plain C4.5 *etc.*).

It is also interesting to compare results between the original C4.5 algorithm and the wrapped versions of ID3, C4.5, and Naive-Bayes. Table 16 shows accuracy results for C4.5, ID3 with best-first forward feature subset selection, C4.5 with best-first backward FSS with compound operators, and Naive-Bayes with backward compound-operator FSS. The following observations can be made:

- For real datasets, ID3-FSS and C4.5 perform approximately the same, but ID3-FSS uses fewer features. For the artificial datasets, ID3-FSS significantly outperforms C4.5 on three datasets (Corral, Monk1, Monk2-local), and is significantly inferior in one (m-of-n-3-7-10).

- C4.5-FSS significantly outperforms C4.5 on two real datasets (cleve and DNA), two artificial datasets (Monk1 and Monk2-local), and is never significantly outperformed by C4.5. The relative error is reduced by 5.6% for real datasets and by 26.4% for the artificial datasets.

- What is perhaps most interesting is how C4.5 and Naive-Bayes with feature subset selection compare. While there are datasets for which either one is better than the other, on the real datasets, C4.5 is significantly better only for the horse-colic dataset, but Naive-Bayes is significantly better for cleve, DNA, Pima, and soybean-large. The relative error of Naive-Bayes is smaller by 16.1%. For the artificial datasets, the two are about equal: C4.5 is significantly better on two datasets (Monk1, Monk2-local), and Naive-Bayes is better on two (Corral, m-of-n-3-7-10).

In summary, feature subset selection using the wrapper approach significantly improves ID3, C4.5 and Naive-Bayes on some of the datasets tested. On the real datasets, the wrapper approach is clearly superior

Table 16: A comparison of C4.5 with ID3-FSS, C4.5-FSS, and Naive-Bayes-FSS. The p-val columns indicates the probability that the column before it is improving over C4.5

Dataset	C4.5 original	ID3-FSS Frwd-BFS	p-val	C4.5-FSS Back-BFS	p-val	NB-FSS Back-BFS	p-val
breast cancer	95.42± 0.7	94.57± 0.7	0.11	95.28± 0.6	0.41	96.00± 0.6	0.81
cleve	72.30± 2.2	79.52± 2.3	**1.00**	77.88± 3.2	**0.98**	82.56± 2.5	**1.00**
crx	85.94± 1.4	85.22± 1.6	0.31	85.80± 1.3	0.46	84.78± 0.8	0.15
DNA	92.66± 0.8	94.27± 0.7	**0.99**	94.44± 0.7	**0.99**	96.12± 0.6	**1.00**
horse-colic	85.05± 1.2	82.07± 1.5	**0.01**	84.77± 1.3	0.41	82.33± 1.3	**0.01**
Pima	71.60± 1.9	68.73± 2.2	0.08	70.18± 1.3	0.20	76.03± 1.6	**0.99**
sick-euthyroid	97.73± 0.5	97.06± 0.5	0.09	97.91± 0.4	0.66	97.35± 0.5	0.21
soybean-large	91.35± 1.6	91.65± 1.0	0.59	91.93± 1.3	0.65	94.29± 0.9	**0.99**
Corral	81.25± 3.5	100.00± 0.0	**1.00**	81.25± 3.5	0.50	90.62± 2.6	**1.00**
m-of-n-3-7-10	85.55± 1.1	77.34± 1.3	**0.00**	85.16± 1.1	0.36	87.50± 1.0	**0.97**
Monk1	75.69± 2.1	97.22± 0.8	**1.00**	88.89± 1.5	**1.00**	72.22± 2.2	**0.05**
Monk2-local	70.37± 2.2	95.60± 1.0	**1.00**	88.43± 1.5	**1.00**	67.13± 2.3	0.07
Monk2	65.05± 2.3	63.89± 2.3	0.31	67.13± 2.3	0.82	67.13± 2.3	0.82
Monk3	97.22± 0.8	97.22± 0.8	0.50	97.22± 0.8	0.50	97.22± 0.8	0.50
Average real:	86.51	86.64		87.27		88.68	
Average artif.	79.19	88.55		84.68		80.30	

to the filter method. Perhaps the most surprising result is how well Naive-Bayes performs on real datasets once discretization and feature subset selection are performed. Some explanations for the apparently high accuracy of Naive-Bayes even when the independence assumptions are violated, are explained in Domingos & Pazzani (1996). However, we can see that in some real-world domains such as DNA, the feature selection step is important to improve performance.

6 Overfitting

Still, it is an error to argue in front of your data. You find yourself insensibly twisting them round to fit your theories.
—Sherlock Holmes / The Adventure of Wisteria Lodge.

An induction algorithm **overfits** the dataset if it models the training data too well and its predictions are poor. An example of an over-specialized hypothesis, or classifier, is a lookup table on all the features. Overfitting is closely related to the bias-variance tradeoff (Kohavi & Wolpert 1996, Geman et al. 1992, Breiman et al. 1984): if the algorithm fits the data too well, the variance term is large, and hence the overall error is increased.

Most accuracy estimation methods, including cross-validation, evaluate the predictive power of a given hypothesis over a feature subset by setting aside instances (holdout sets) that are not shown to the induction algorithm and using them to assess the predictive ability of the induced hypothesis. A search algorithm that explores a large portion of the space and that is guided by the accuracy estimates can choose a bad feature subset: a subset with a high accuracy estimate but poor predictive power.

Overuse of the accuracy estimates in feature subset selection may cause overfitting in the feature-subset space. Because there are so many feature subsets, it is likely that one of them leads to a hypothesis that has high predictive accuracy for the holdout sets. A good example of overfitting can be shown using a *no-information* dataset (Rand) where the features and the label are completely random. The top graph in Figure 20 shows the estimated accuracy versus the true accuracy for the best node the search has found after expanding k nodes. One can see that especially for the small sample of size 100, the estimate is extremely poor (26% optimistic), indicative of overfitting. The bottom graphs in the figure show overfitting in small real-world datasets.

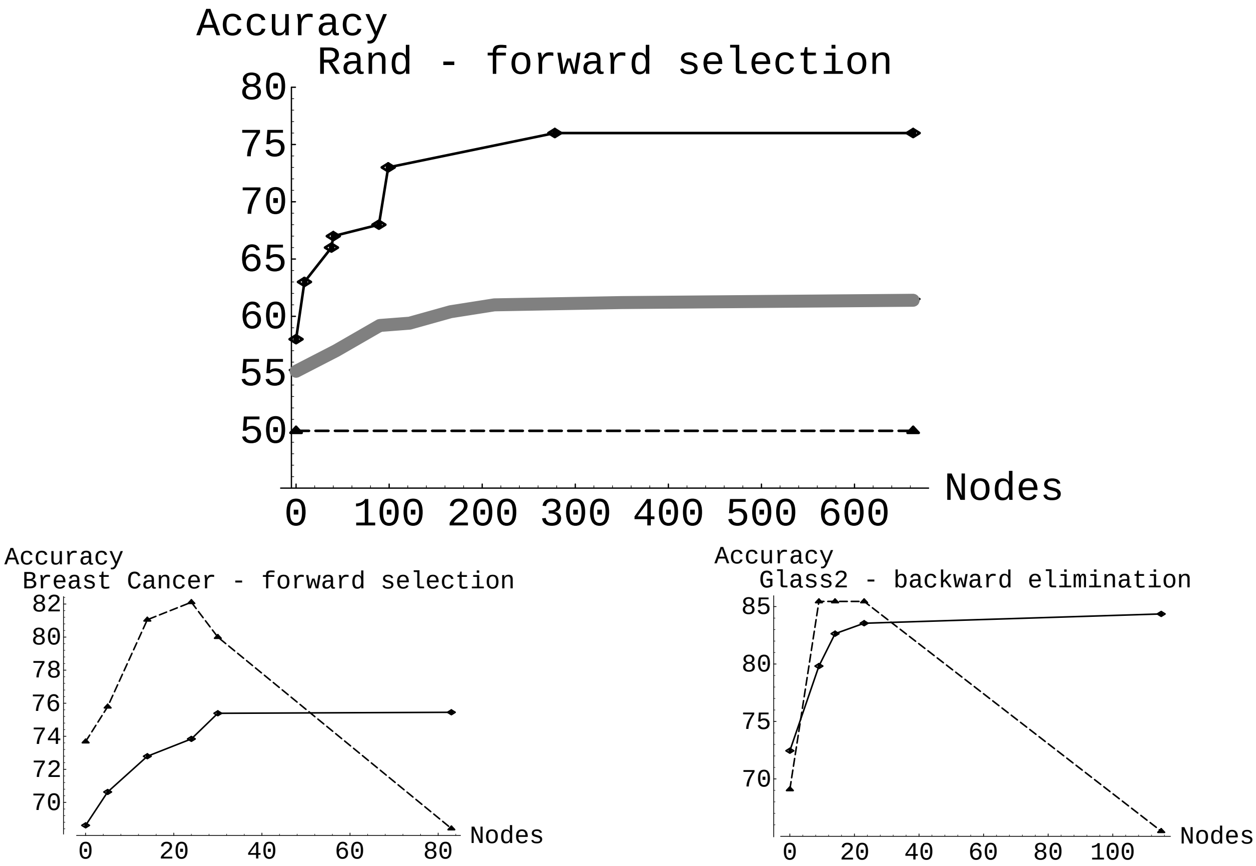

Figure 20: Overfitting in feature subset selection. The top graph shows the estimated and true accuracies for a random dataset and ID3. The solid line represents the estimated accuracy for a training set of 100 instances, the thick grey line for a training set of 500 instances, and the dotted line shows the real accuracy. The bottom graphs graphs show the accuracy for real-world datasets. The solid line is the estimated accuracy, and the dotted line is the accuracy on an independent test set.

Recently, a few machine learning researchers have reported the cross-validation estimates that were used to guide the search as a final estimate of performance, thus reporting overly optimistic results. Instead, experiments using cross-validation to guide the search must report the accuracy of the selected feature subset on a *separate* test set or on holdout sets generated by an external loop of cross-validation that were never used during the feature subset selection process.

The problem of overfitting in feature subset space has been previously raised in the machine learning community by Wolpert (1992*a*) and Schaffer (1993), and the subject has received much attention in the statistics community (cf. Miller (1990)). Although the theoretical problem exists, our experiments indicate that overfitting is mainly a problem when the number of instances is small. Kohavi & Sommerfield (1995) reported that out of 70 searches for feature subsets with datasets containing over 250 instances, ten searches were optimistically biased by more than two standard deviations and one was pessimistically biased by more than two standard deviations; 3.5 searches (5% of 70) are expected to be biased. While the problem clearly exists, it was not very severe on the datasets examined (all datasets contained more than 250 instances in the training set). Moreover, even if the estimates are biased, the algorithm may still choose the correct feature subsets because it is the relative accuracy that matters most.

7 Subset Selection as Search with Probabilistic Estimates

We now look at the problem of feature subset selection as search with probabilistic estimates, which generalizes standard search with deterministic state evaluations. The wrapper approach, using accuracy estimation for node evaluation as the heuristic function, complicates the common state-space search paradigm. The fact that the accuracy estimation is a random variable implies that there is uncertainty in the returned estimate. One way to decrease the variance is to run the accuracy estimation (*e.g.*, k-fold cross-validation) more than once and average the results, as we have done. Increasing the number of runs shrinks the confidence interval for the mean, but requires more time. The tradeoff between more accurate estimates and more extensive exploration of the search space is referred to as the exploration versus exploitation problem (Kaelbling 1993). We can either exploit our knowledge and shrink the confidence intervals of the explored nodes to make sure we select the right one, or we can explore new nodes in the hope of finding better nodes. The tradeoff leads to the following abstract search problem.

Definition 7 (Search with Probabilistic Estimates)
Let $\mathcal{S}$ be a state space with operators between states. Let $f : \mathcal{S} \mapsto \mathbb{R}$ be an unbiased probabilistic evaluation function that maps a state to a real number, indicating how good the state is. The number returned by $f(s)$ comes from a distribution $D(s)$ with mean $f^*(s)$, which is the actual (unknown) value of the state. The goal is to find the state s with the maximal value of $f^*(s)$.

The mapping of this definition to the feature subset selection problem is as follows. The states are the subsets, and the operators are the common ones (add feature, delete feature, create compound node). The evaluation function is the accuracy estimation. Although some accuracy estimation techniques, such as cross-validation, are biased, they can be viewed as unbiased estimators for a different quantity; for example, k-fold cross-validation is unbiased for datasets of size $m - m/k$. Furthermore, for model selection, this pessimism is of minor importance because the bias may cancel out. We now describe work that falls under this general framework of search with probabilistic estimators.

Greiner (1992) described a method for conducting a hill-climbing search when the evaluation function is probabilistic. The algorithm stops at a node that is a local optimum with high probability, based on the Chernoff bound. Yan & Mukai (1992) analyzed an algorithm based on simulated annealing and showed that it will find the global optimum if given enough time.

Maron & Moore (1994), in an approach similar to Greiner's, attempted to shrink the confidence interval of the accuracy for a given set of models, until one model can be proven to be optimal with high probability. The evaluation function is a single step in leave-one-out cross-validation, *i.e.*, the algorithm is trained on randomly chosen $n-1$ instances and tested on the one that is left. The induction algorithm used is instance-based learning, which leads to an extremely fast evaluation because training is not necessary. A step of leave-one-out is merely a test of whether an instance is classified correctly by its nearest-neighbor. Note, however, that $f(s)$ always returns either a zero or a one. The instance is either correctly classified, or not. This step must be repeated many times to get a reasonable confidence bound.

The general idea is to *race* competing models, until one is a clear winner. Models drop out of the race when the confidence interval of the accuracy does not overlap with the confidence interval of the accuracy of the best model (this is analogous to imposing a higher and lower bound on the estimation function in the B^* algorithm (Berliner 1981)). The race ends when there is a winner, or when all n steps in the leave-one-out cross-validation have been executed. The confidence interval is defined according to Hoeffding's formula (Hoeffding 1963):

$$p\left(\left|f^*(s) - \widehat{f}(s)\right| > \epsilon\right) < 2e^{-2m\epsilon^2/B^2}$$

where $\widehat{f}(s)$ is the average of m evaluations and B bounds the possible spread of point values. Given a confidence level, one can determine ϵ, and hence a confidence interval for $f^*(s)$, from the above formula. The paper (Maron & Moore 1994), however, does not discuss any search heuristic, and assumes that a fixed set of models is given by some external source.

Moore & Lee (1994) describe an algorithm for feature subset selection that has both ingredients of the abstract problem: it has a search heuristic, and it uses the probabilistic estimates in a non-trivial manner. The algorithm does a forward selection and backward elimination, but instead of estimating the accuracy of each added (deleted) feature using leave-one-out cross-validation, all the features that can be added (deleted) are raced in parallel until there is a clear winner.

Schemata search (Moore & Lee 1994) is another search variant that allows taking into account interactions between features. Instead of starting with the empty or full set of features, the search begins with all features marked as "unknown." Each time a feature is chosen and raced between being "in" or "out." All combinations of "unknown" features are used in equal probability, thus a feature that should be "in" will win the race, even if correlated with another feature. Although this method uses the probabilistic estimates in a Bayesian setting, the basic search strategy is simple hill-climbing.

Fong (1995) gives bounds for the sample complexity (the number of samples one needs to collect before termination) in the k-armed bandit problem. His γ-IE approach allows trading off exploitation and exploration, thus generalizing Kaelbling's interval estimation strategy (Kaelbling 1993). However, in all cases the worst-case bound remains the same and the optimal tradeoff between exploration and exploitation was empirically determined to be domain dependent.

When using the wrapper method, it is important to explore a sufficient portion of the search space. By using search algorithms that take advantage of the probabilistic nature of accuracy estimates, it is possible to explore a larger portion of the space if the evaluation time for a state can be reduced based on statistical estimates. Future work on the abstract problem presented above might improve the applicability of the wrapper method to even larger state spaces.

8 Related Work

The pattern recognition literature (Devijver & Kittler 1982, Kittler 1986, Ben-Bassat 1982), statistics literature (Draper & Smith 1981, Miller 1984, Miller 1990, Neter et al. 1990), and recent machine learning papers (Almuallim & Dietterich 1991, Almuallim & Dietterich 1994, Kira & Rendell 1992*a*, Kira & Rendell 1992*b*, Kononenko 1994) consist of many measures for feature subset selection that are all based on the data alone.

Most measures in the pattern recognition and statistics literature are monotonic, *i.e.*, for a sequence of nested feature subsets $F_1 \supseteq F_2 \supseteq \cdots \supseteq F_k$, the measure f obeys $f(F_1) \geq f(F_2) \geq \cdots \geq f(F_k)$. Notable selection measures that satisfy the monotonicity assumption are residual sum of squares (RSS), adjusted R-square, minimum mean residual, Mallow's C_p (Mallows 1973), discriminant functions, and distance measures, such as the Bhattacharyya distance and divergence. The PRESS measure (Prediction sum of squares), however, does not obey monotonicity. For monotonic functions, branch and bound techniques can be used to prune the search space. Furnival & Wilson (1974) show how to compute the residual sum of squares (RSS) for all possible regressions of k features in less than six (!) floating-point operations per regression; furthermore, the technique can be combined with branch and bound algorithms as described in their paper.[3] Narendra & Fukunaga (1977) apparently rediscovered the branch-and-bound technique, which was later improved in Yu & Yuan (1993). Most machine learning induction algorithms do not obey monotonic restrictions, and so this type of dynamic programming cannot be used. Even when branch and bound can be used, the search is usually exponential, and when there are more than 30 or 40 features, heuristic methods need to be used.

Searching in the space of feature subsets has been studied for many years. Sequential backward elimination, sometimes called sequential backward selection, was introduced by Marill & Green (1963). Kittler (1978) generalized the different variants including forward methods, stepwise methods, and "plus ℓ–take away r." Cover & Campenhout (1977) showed that even for multivariate normally distributed features, no hill-climbing procedure that uses a monotonic measure and that selects one feature at a time can find the best feature subset of a desired size; even a 2-1 algorithm that adds the best pair and removes the worst single feature can fail. More recent papers attempt to use AI techniques, such as beam search and bidirectional search (Siedlecki & Sklansky 1988), best-first search (Xu, Yan & Chang 1989), and genetic algorithms (Vafai & De Jong 1992, Vafai & De Jong 1993). All the algorithms described above use a deterministic evaluation function, although in some cases they can easily be extended to probabilistic estimates, such as cross-validation that we use. Recently, Bala, Jong, Haung, Vafaie & Wechsler (1995) used the wrapper

[3] The Forest Service must have been really interested in this problem. Furnival was at the School of Forestry at Yale University, and Wilson was from the USDA Forest Service! One would think that they should have been working on tree pruning and not on linear regression.

approach with holdout for accuracy estimation and a genetic algorithm to search the space. Langley (1994) reviewed feature subset selection methods in machine learning and contrasted the wrapper and filter approaches. Atkeson (1991) used leave-one-out cross-validation to search a multidimensional real-valued space which includes feature weights in addition to other parameters for local learning.

The theory of rough sets defines notions of relevance that are closely related to the ones defined here (Pawlak 1991). The set of strongly relevant features form the **core** and any set of features that allow a Bayes classifier to achieve the highest possible accuracy forms a **reduct.** A reduct can only contain strongly relevant and weakly relevant features. Pawlak (1991) shows that the core is the intersection of all the reducts and that every reduct consists only of the core features and weakly relevant features. Pawlak (1993) wrote that one of the most important and fundamental notions to the rough sets philosophy is the need to discover redundancy and dependencies between features, and there has been a lot of work on feature subset selection coming from the rough sets community (cf. Modrzejewski (1993) and Ziarko (1991)). While the goal of finding a good feature subset is the same, Kohavi & Frasca (1994) have claimed that relevance does not necessarily imply usefulness for induction tasks (see also Section 2.3).

While we concentrated on selection of relevant features in this paper, an alternative method is to weigh features, giving each one a degree of relevance. Theoretical results have been shown for multiplicative learning algorithms, which work well for linear combinations of features (*e.g.*, Perceptrons) (Littlestone & Warmuth 1994).

Skalak (1994) uses the wrapper approach for feature subset selection and for decreasing the number of prototypes stored in instance-based methods. He shows that very few prototypes sometimes suffice. This is an example of choosing relevant training instances as opposed to relevant features.

Turney (1996) defines a feature to be **primary** if there is one feature value such that the probability of a class changes when conditioned on this value.[4] A primary feature is thus informative about the class when considered all by itself. He then defines a **contextual feature** as a non-primary relevant feature. A feature is contextual only if it helps in the context of all others. Contextual features are harder to find because they involve interactions. These definitions are orthogonal to ours: a feature may be primary and either strongly or weakly relevant, or contextual and either strongly or weakly relevant.

Since the introduction of the wrapper approach (John et al. 1994), we have seen it used in a few papers. Langley & Sage (1994*a*) used the wrapper approach to select features for Naive-Bayes (but without discretization) and Langley & Sage (1994*b*) used it to select features for a nearest-neighbor algorithm. Pazzani (1995) used the wrapper approach to select features and join features (create super-features that compound others) for Naive-Bayes and showed that it indeed finds correct combinations when features interact. Singh & Provan (1995) and Provan & Singh (1995) used the wrapper approach to select features for Bayesian networks and showed significant improvements over the original K2 algorithm. Street, Mangasarian & Wolberg (1995) use the wrapper in the context of a linear programming generalizer. All the algorithms mentioned above use a hill-climbing search engine.

The idea of wrapping around induction algorithms appeared several times in the literature without the explicit name "wrapper approach." The closest formulation is the *Search of the Bias Space* approach described in Provost & Buchanan (1995) and which dates back to Provost (1992).

Aha & Bankert (1994) used the wrapper for identifying feature subsets in a cloud classification problem with 204 features and 1633 instances; they concluded that their empirical results strongly support the claim that the wrapper strategy is superior to filter methods. Aha & Bankert (1995) compare forward and backward feature subset selection using the wrapper approach and a beam-search engine and conclude that forward selection is better. In other work, we have applied the wrapper approach to parameter tuning as well (specifically, setting the parameters of C4.5 for maximal performance) in Kohavi & John (1995). Mladenić (1995) independently extended the use of wrappers from feature subset selection to parameter tuning. Doak (1992) has developed a method similar to the wrapper approach independently, and compared many search engines for feature subset selection; however, he was not aware of the fact that one should use an independent test set for the final estimation and used the accuracy estimation used to guide the search (see Section 6).[5]

[4] A longer discussion of contextual features may be found in Turney (1993), although the definitions originally given were found to be flawed as mentioned in Turney (1996).

[5] The results in both papers by Aha and Bankert, those of Mladenić, and those of Doak must be interpreted cautiously because they were using the cross-validation accuracy used during the search as the final estimated performance as opposed to an independent test set or an external loop of cross-validation as we have done.

9 Future Work

Many variations and extensions of the current work are possible. We have examined hill-climbing and best-first search engines. Other approaches could be examined, such as simulated annealing approaches that evaluate the better nodes more times (Laarhoven & Aarts 1987). Looking at the search, we have seen that one general area of the search space is explored heavily when it is found to be good. It might be worthwhile to introduce some diversity into the search, following the genetic algorithm and genetic programming approaches (Holland 1992, Goldberg 1989, Koza 1992). The problem has been abstracted as search with probabilistic estimates (Section 7), but we have not done experiments in an attempt to understand the tradeoff between the quality of the estimates and the search size, *i.e.*, exploration versus exploitation experiments.

The search for a good subset is conducted in a very large space. We have started the search from the empty set of features and from the full set of features, but one can start from some other initial node. One possibility is to estimate which features are strongly relevant, and start the search from this subset, although compound operators seem to be a partial answer to this problem. Another possibility is to start at random points and conduct a series of hill-climbing searches. We could also start with the set of features suggested by Relieved-F, or at least ensure that this set is explored by the wrapper at some point during the search.

The wrapper approach is very slow. For larger datasets, it is possible to use cheaper accuracy estimation methods, such as holdout, or decrease the number of folds. Furthermore, some inducers allow incremental operations on the classifiers (add and delete instances), leading to the possibility of doing incremental cross-validation as suggested in Kohavi (1995*a*), thus drastically reducing the running time. Although C4.5 does not support incremental operations, Utgoff (1994) has shown that this is possible and has implemented a fast version of leave-one-out for decision trees (Utgoff 1995). The wrapper approach is also very easy to parallelize. In a node expansion, all children can be evaluated in parallel, which will cut the running time by a factor equal to the number of attributes assuming enough processors are available (*e.g.*, 180 for DNA).

In theory, every possible feature subset identifies a different model, so the problem can be viewed as that of model selection (Linhart & Zucchini 1986) in Statistics. If there are only a few models, as is the case when one chooses between three induction algorithms, one can estimate the accuracy of each one and select the one with the highest accuracy (Schaffer 1993) or perhaps even find some underlying theory to help predict the best one for a given dataset (Brazdil, Gama & Henery (1994)). For all but the smallest problems, the space of possible feature subsets is too large for brute-force enumeration of all possibilities, and we must resort to heuristic search.

Recently, aggregation techniques, sometimes called stacking, have been advocated by many people in machine learning, neural networks, and Statistics (Wolpert 1992*b*, Breiman 1996, Freund & Schapire 1995, Schapire 1990, Freund 1990, Perrone 1993, Krogh & Vedelsby 1995, Buntine 1992, Kwok & Carter 1990). It is possible to build many models, each one with a different parameter setting or with a different feature subset, and let them vote on the class. Aggregation techniques reduce the variance of the models by aggregating them, but they make it extremely hard to interpret the resulting classifier.

10 Summary

We have described the feature subset selection problem in supervised learning, which involves identifying the relevant or useful features in a dataset and giving only that subset to the learning algorithm. We have investigated the relevance and irrelevance of features, and concluded that weak and strong relevance are needed to capture our intuition better. We have then shown that these definitions are mainly useful with respect to an optimal rule, *i.e.*, Bayes rule, but that in practice one should look for optimal features with respect to the specific learning algorithm and training set at hand. Such optimal features do not necessarily correspond to relevant features (either weak or strong) as shown in Section 2.3. The optimal features depend on the specific biases and heuristics of the learning algorithm, and hence the wrapper approach naturally fits with this definition. Feature relevance helped motivate compound operators, which work well in practice and are currently the only practical way to conduct backward searches for feature subsets using the wrapper approach when the datasets have many features.

The wrapper approach requires a search space, operators, a search engine, and an evaluation function. For the evaluation function, we used cross-validation as our accuracy estimation technique, based on the results in Kohavi (1995*b*). We have used the common search space with add and delete operators as the basis for comparing two search engines: hill-climbing and best-first search. We have then defined compound operators that use more information in the children of an expanded node, not just the maximum value. These compound operators make a backward search, starting from the full set of features, practical. Best-first search with compound operators seems to be a strong performer and improves ID3, C4.5, and Naive-Bayes, both in accuracy, and in comprehensibility, as measured by the number of features used.

We showed several problems with filter methods that attempt to define relevance independently of the learning algorithm. These problems include: inability to remove a feature in symmetric targets concepts such as m-of-n-3-7-10 where removal of one feature improves performance (Section 4), inability to include irrelevant features that may actually help performance (Example 3), and inability to remove correlated features that may hurt performance (Section 2.4.4). Not only have we given theoretical reasons why relevance should be defined relative to an algorithm, but we conducted experiments comparing the wrapper approach with Relieved-F, a filter approach to feature subset selection.

Our comparisons include two different families of induction algorithms: decision trees and Naive-Bayes. Significant performance improvement is achieved for both on some datasets. For the DNA dataset, which was extensively compared in the StatLog project, the wrapper approach using Naive-Bayes reduced the error rate from 6.1% to 3.9% (a relative error reduction of 36%), making it the best known induction algorithm for this problem. One of the more surprising results was how well Naive-Bayes performed overall: in the global comparison (Table 16), Naive-Bayes outperforms C4.5 (with and without feature selection) on the real datasets. On average, the performance using feature subset selection improved both algorithms.

Our experiments were done on real and artificial datasets. In some cases, the results varied dramatically between these two sets. One reason is that many of the real datasets were already preprocessed to include only relevant features (DNA being the only exception), while the artificial ones included irrelevant features on purpose. The artificial datasets were mostly noise-free (except monk3), while the real ones contained noise. Finally, the artificial problems contained high-order interactions, which make it harder for hill-climbing algorithms such as C4.5 to find the optimal feature subset. We expect that tougher problems containing interactions will occur more in unprocessed datasets coming from the real world.

We have also shown some problems with the wrapper approach, namely overfitting and the large amounts of CPU time required, and we defined the search problem as an abstract state space search with probabilistic estimates, a formulation that may capture other general problems and that might be studied independently to solve the existing problems. The time issue seems to be the most important, although with larger amounts of data, cross-validation can be replaced with holdout accuracy estimation for an immediate improvement in time by a factor of five. Overfitting is a problem of lesser importance and seems to occur mostly in small training sets; as more data is available for training, overfitting it by chance is much harder.

In supervised classification learning, the question of whether a feature in a dataset is relevant to a given prediction task is less useful than the question of whether a feature is relevant to the prediction task given a learning algorithm. If the goal is to optimize accuracy, one should ask whether a set of features is optimal for a task given the learning algorithm and the training set. Different algorithms have different biases and a feature that may help one algorithm may hurt another. Similarly, different training set sizes might imply that a different set of features is optimal. If only a small training set is given, it may be better to reduce the number of features and thus reduce the algorithm's variance; when more instances are given, more features can be chosen to reduce the algorithm's bias.

Acknowledgments

We would also like to thank Karl Pfleger for his help in formulating the wrapper idea. We would like thank our anonymous reviewers. One reviewer formulated Example 3, which is much better than our original example. Pat Langley, Nick Littlestone, Nils Nilsson, and Peter Turney gave helpful feedback on the ideas and presentation. Dan Sommerfield implemented large parts of the wrapper in $\mathcal{MLC}$++ (Kohavi et al. 1996), and all of the experiments were done using $\mathcal{MLC}$++. George John's work was supported under a National Science Foundation Graduate Research Fellowship. Most of the research for this paper was completed while the authors were at Stanford University.

References

Aha, D. W. (1992), "Tolerating noisy, irrelevant and novel attributes in instance-based learning algorithms", *International Journal of Man-Machine Studies* **36**(1), pp. 267–287.

Aha, D. W. & Bankert, R. L. (1994), Feature selection for case-based classification of cloud types: An empirical comparison, in *Working Notes of the AAAI-94 Workshop on Case-Based Reasoning*, pp. 106–112.

Aha, D. W. & Bankert, R. L. (1995), A comparative evaluation of sequential feature selection algorithms, in D. Fisher & H. Lenz, eds, *Proceedings of the Fifth International Workshop on Artificial Intelligence and Statistics*, Ft. Lauderdale, FL, pp. 1–7.

Aha, D. W., Kibler, D. & Albert, M. K. (1991), "Instance-based learning algorithms", *Machine Learning* **6**(1), pp. 37–66.

Almuallim, H. & Dietterich, T. G. (1991), Learning with many irrelevant features, in *Ninth National Conference on Artificial Intelligence*, MIT Press, pp. 547–552.

Almuallim, H. & Dietterich, T. G. (1994), "Learning boolean concepts in the presence of many irrelevant features", *Artificial Intelligence* **69**(1-2), pp. 279–306.

Anderson, J. R. & Matessa, M. (1992), "Explorations of an incremental, Bayesian algorithm for categorization", *Machine Learning* **9** pp. 275–308.

Atkeson, C. G. (1991), Using locally weighted regression for robot learning, in *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 958–963.

Bala, J., Jong, K. A. D., Haung, J., Vafaie, H. & Wechsler, H. (1995), Hybrid learning using genetic algorithms and decision trees for pattern classification, in C. S. Mellish, ed., *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, Morgan Kaufmann Publishers, Inc., pp. 719–724.

Ben-Bassat, M. (1982), Use of distance measures, information measures and error bounds in feature evaluation, in P. R. Krishnaiah & L. N. Kanal, eds, *Handbook of Statistics*, Vol. 2, North-Holland Publishing Company, pp. 773–791.

Berliner, H. (1981), The B* tree search algorithm: A best-first proof procedure, in B. Webber & N. Nilsson, eds, *Readings in Artificial Intelligence*, Morgan Kaufmann, pp. 79–87.

Blum, A. L. & Rivest, R. L. (1992), "Training a 3-node neural network is NP-complete", *Neural Networks* **5** pp. 117–127.

Boddy, M. & Dean, T. (1989), Solving time-dependent planning problems, in N. S. Sridharan, ed., *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, Vol. 2, Morgan Kaufmann Publishers, Inc., pp. 979–984.

Brazdil, P., Gama, J. & Henery, B. (1994), Characterizing the applicability of classification algorithms using meta-level learning, in F. Bergadano & L. D. Raedt, eds, *Proceedings of the European Conference on Machine Learning*.

Breiman, L. (1996), "Bagging predictors", *Machine Learning* **24** pp. 123–140.

Breiman, L., Friedman, J. H., Olshen, R. A. & Stone, C. J. (1984), *Classification and Regression Trees*, Wadsworth International Group.

Buntine, W. (1992), "Learning classification trees", *Statistics and Computing* **2**(2), June, pp. 63–73.

Cardie, C. (1993), Using decision trees to improve case-based learning, in *Proceedings of the Tenth International Conference on Machine Learning*, Morgan Kaufmann Publishers, Inc., pp. 25–32.

Caruana, R. & Freitag, D. (1994), Greedy attribute selection, in W. W. Cohen & H. Hirsh, eds, *Machine Learning: Proceedings of the Eleventh International Conference*, Morgan Kaufmann Publishers, Inc., pp. 28–36.

Cestnik, B. (1990), Estimating probabilities: A crucial task in machine learning, in L. C. Aiello, ed., *Proceedings of the ninth European Conference on Artificial Intelligence*, pp. 147–149.

Cover, T. M. & Campenhout, J. M. V. (1977), "On the possible orderings in the measurement selection problem", *IEEE Transactions on Systems, Man, and Cybernetics* **SMC-7**(9), pp. 657–661.

Dasarathy, B. V. (1990), *Nearest Neighbor (NN) Norms: NN Pattern Classification Techniques*, IEEE Computer Society Press, Los Alamitos, California.

De Mántaras, R. L. (1991), "A distance-based attribute selection measure for decision tree induction", *Machine Learning* **6** pp. 81–92.

Devijver, P. A. & Kittler, J. (1982), *Pattern Recognition: A Statistical Approach*, Prentice-Hall International.

Doak, J. (1992), An evaluation of feature selection methods and their application to computer security, Technical Report CSE-92-18, University of California at Davis.

Domingos, P. & Pazzani, M. (1996), Beyond independence: conditions for the optimality of the simple Bayesian classifier, in L. Saitta, ed., *Machine Learning: Proceedings of the Thirteenth International Conference*, Morgan Kaufmann, pp. 105–112.

Dougherty, J., Kohavi, R. & Sahami, M. (1995), Supervised and unsupervised discretization of continuous features, in A. Prieditis & S. Russell, eds, *Machine Learning: Proceedings of the Twelfth International Conference*, Morgan Kaufmann, pp. 194–202.

Draper, N. R. & Smith, H. (1981), *Applied Regression Analysis*, 2nd edition, John Wiley & Sons.

Duda, R. & Hart, P. (1973), *Pattern Classification and Scene Analysis*, Wiley.

Fayyad, U. M. (1991), On the induction of decision trees for multiple concept learning, PhD thesis, EECS Dept, Michigan University.

Fayyad, U. M. & Irani, K. B. (1992), The attribute selection problem in decision tree generation, in *Tenth National Conference on Artificial Intelligence*, MIT Press, pp. 104–110.

Fong, P. W. L. (1995), A quantitative study of hypothesis selection, in A. Prieditis & S. Russell, eds, *Machine Learning: Proceedings of the Twelfth International Conference*, Morgan Kaufmann Publishers, Inc., pp. 226–234.

Freund, Y. (1990), Boosting a weak learning algorithm by majority, in *Proceedings of the Third Annual Workshop on Computational Learning Theory*, pp. 202–216. To appear in Information and Computation.

Freund, Y. & Schapire, R. E. (1995), A decision-theoretic generalization of on-line learning and an application to boosting, in *Proceedings of the Second European Conference on Computational Learning Theory*, Springer-Verlag, pp. 23–37.

Furnival, G. M. & Wilson, R. W. (1974), "Regression by leaps and bounds", *Technometrics* **16**(4), pp. 499–511.

Geman, S., Bienenstock, E. & Doursat, R. (1992), "Neural networks and the bias/variance dilemma", *Neural Computation* **4** pp. 1–48.

Gennari, J. H., Langley, P. & Fisher, D. (1989), "Models of incremental concept formation", *Artificial Intelligence* **40** pp. 11–61.

Ginsberg, M. L. (1993), *Essentials of Artificial Intelligence*, Morgan Kaufmann.

Goldberg, D. E. (1989), *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley Publishing Company.

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

Greiner, R. (1992), Probabilistic hill climbing : Theory and applications, in J. Glasgow & R. Hadley, eds, *Proceedings of the Ninth Canadian Conference on Artificial Intelligence*, Morgan Kaufmann Publishers, Inc., pp. 60–67.

Hancock, T. R. (1989), On the difficulty of finding small consistent decision trees, Unpublished Manuscript, Harvard University.

Hoeffding, W. (1963), "Probability inequalities for sums of bounded random variables", *Journal of the American Statistical Association* **58** pp. 13–30.

Holland, J. H. (1992), *Adaptation in natural and artificial systems : an introductory analysis with applications to biology, control, and artificial intelligence*, MIT Press.

Hyafil, L. & Rivest, R. L. (1976), "Constructing optimal binary decision trees is NP-complete", *Information Processing Letters* **5**(1), pp. 15–17.

John, G. H. (1997), Enhancements to the Data Mining Process, PhD thesis, Stanford University, Computer Science department.

John, G., Kohavi, R. & Pfleger, K. (1994), Irrelevant features and the subset selection problem, in *Machine Learning: Proceedings of the Eleventh International Conference*, Morgan Kaufmann, pp. 121–129.

Judd, S. (1988), "On the complexity of loading shallow neural networks", *Journal of Complexity* **4** pp. 177–192.

Kaelbling, L. P. (1993), *Learning in Embedded Systems*, MIT Press.

Kira, K. & Rendell, L. A. (1992*a*), The feature selection problem: Traditional methods and a new algorithm, in *Tenth National Conference on Artificial Intelligence*, MIT Press, pp. 129–134.

Kira, K. & Rendell, L. A. (1992*b*), A practical approach to feature selection, in *Proceedings of the Ninth International Conference on Machine Learning*, Morgan Kaufmann.

Kittler, J. (1978), Une généralisation de quelques algorithms sous-optimaux de recherche d'ensembles d'attributs, in *Proc. Congrés Reconnaissance des Formes et Traitement des Images*.

Kittler, J. (1986), *Feature Selection and Extraction*, Academic Press, Inc, chapter 3, pp. 59–83.

Kohavi, R. (1994), Feature subset selection as search with probabilistic estimates, in *AAAI Fall Symposium on Relevance*, pp. 122–126.

Kohavi, R. (1995*a*), The power of decision tables, in N. Lavrac & S. Wrobel, eds, *Proceedings of the European Conference on Machine Learning*, Lecture Notes in Artificial Intelligence 914, Springer Verlag, Berlin, Heidelberg, New York, pp. 174–189.

Kohavi, R. (1995*b*), A study of cross-validation and bootstrap for accuracy estimation and model selection, in C. S. Mellish, ed., *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, Morgan Kaufmann, pp. 1137–1143.

Kohavi, R. (1995*c*), Wrappers for Performance Enhancement and Oblivious Decision Graphs, PhD thesis, Stanford University, Computer Science department. STAN-CS-TR-95-1560, ftp://starry.stanford.edu/pub/ronnyk/teza.ps.

Kohavi, R. & Frasca, B. (1994), Useful feature subsets and rough set reducts, in *Third International Workshop on Rough Sets and Soft Computing*, pp. 310–317. Also appeared in *Soft Computing* by Lin and Wildberger.

Kohavi, R. & John, G. (1995), Automatic parameter selection by minimizing estimated error, in A. Prieditis & S. Russell, eds, *Machine Learning: Proceedings of the Twelfth International Conference*, Morgan Kaufmann, pp. 304–312.

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

Kohavi, R. & Wolpert, D. H. (1996), Bias plus variance decomposition for zero-one loss functions, in L. Saitta, ed., *Machine Learning: Proceedings of the Thirteenth International Conference*, Morgan Kaufmann, pp. 275–283. Available at `http://robotics.stanford.edu/users/ronnyk`.

Kohavi, R., Sommerfield, D. & Dougherty, J. (1996), Data mining using $\mathcal{MLC}$++: A machine learning library in C++, in *Tools with Artificial Intelligence*, IEEE Computer Society Press, pp. 234–245. Received the best paper award. `http://www.sgi.com/Technology/mlc`.

Kononenko, I. (1994), Estimating attributes: Analysis and extensions of Relief, in F. Bergadano & L. D. Raedt, eds, *Proceedings of the European Conference on Machine Learning*.

Kononenko, I. (1995), On biases in estimating multi-valued attributes, in C. S. Mellish, ed., *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, Morgan Kaufmann Publishers, Inc., pp. 1034–1040.

Koza, J. (1992), *Genetic Programming : On the Programming of Computers by Means of Natural Selection*, MIT Press.

Krogh, A. & Vedelsby, J. (1995), Neural network ensembles, cross validation, and active learning, in *Advances in Neural Information Processing Systems*, Vol. 7, MIT Press.

Kwok, S. W. & Carter, C. (1990), Multiple decision trees, in R. D. Schachter, T. S. Levitt, L. N. Kanal & J. F. Lemmer, eds, *Uncertainty in Artificial Intelligence*, Elsevier Science Publishers, pp. 327–335.

Laarhoven, P. & Aarts, E. (1987), *Simulated annealing : Theory and Applications*, Kluwer Academic Publishers.

Langley, P. (1994), Selection of relevant features in machine learning, in *AAAI Fall Symposium on Relevance*, pp. 140–144.

Langley, P. & Sage, S. (1994*a*), Induction of selective bayesian classifiers, in *Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence*, Morgan Kaufmann, Seattle, WA, pp. 399–406.

Langley, P. & Sage, S. (1994*b*), Oblivious decision trees and abstract cases, in *Working Notes of the AAAI-94 Workshop on Case-Based Reasoning*, AAAI Press, Seattle, pp. 113–117.

Langley, P., Iba, W. & Thompson, K. (1992), An analysis of Bayesian classifiers, in *Proceedings of the tenth national conference on artificial intelligence*, AAAI Press and MIT Press, pp. 223–228.

Linhart, H. & Zucchini, W. (1986), *Model Selection*, John Wiley & Sons.

Littlestone, N. & Warmuth, M. K. (1994), "The weighted majority algorithm", *Information and Computation* **108**(2), pp. 212–261.

Mallows, C. L. (1973), "Some comments on c_p", *Technometrics* **15** pp. 661–675.

Marill, T. & Green, D. M. (1963), "On the effectiveness of receptors in recognition systems", *IEEE Transactions on Information Theory* **9** pp. 11–17.

Maron, O. & Moore, A. W. (1994), Hoeffding races: Accelerating model selection search for classification and function approximation, in *Advances in Neural Information Processing Systems*, Vol. 6, Morgan Kaufmann Publishers, Inc.

Merz, C. J. & Murphy, P. M. (1996), UCI repository of machine learning databases, `http://www.ics.uci.edu/~mlearn/MLRepository.html`.

Miller, A. J. (1984), "Selection of subsets of regression variables", *Royal Statistical Society A* **147** pp. 389–425.

Miller, A. J. (1990), *Subset Selection in Regression*, Chapman and Hall.

Minsky, M. L. & Papert, S. (1988), *Perceptrons : an Introduction to Computational Geometry*, MIT Press. Expanded ed.

Mladenić, D. (1995), Automated model selection, in *ECML workshop on Knowledge Level Modeling and Machine Learning*.

Modrzejewski, M. (1993), Feature selection using rough sets theory, in P. B. Brazdil, ed., *Proceedings of the European Conference on Machine Learning*, Springer, pp. 213–226.

Moore, A. W. & Lee, M. S. (1994), Efficient algorithms for minimizing cross validation error, in W. W. Cohen & H. Hirsh, eds, *Machine Learning: Proceedings of the Eleventh International Conference*, Morgan Kaufmann Publishers, Inc.

Moret, B. M. E. (1982), "Decision trees and diagrams", *ACM Computing Surveys* **14**(4), pp. 593–623.

Murthy, S. & Salzberg, S. (1995), Lookahead and pathology in decision tree induction, in C. S. Mellish, ed., *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, Morgan Kaufmann, pp. 1025–1031.

Narendra, M. P. & Fukunaga, K. (1977), "A branch and bound algorithm for feature subset selection", *IEEE Transactions on Computers* **C-26**(9), September, pp. 917–922.

Neter, J., Wasserman, W. & Kutner, M. H. (1990), *Applied Linear Statistical Models*, 3rd edition, Irwin: Homewood, IL.

Pawlak, Z. (1991), *Rough Sets*, Kluwer Academic Publishers.

Pawlak, Z. (1993), "Rough sets: present state and the future", *Foundations of Computing and Decision Sciences* **18**(3-4), pp. 157–166.

Pazzani, M. J. (1995), Searching for dependencies in Bayesian classifiers, in D. Fisher & H. Lenz, eds, *Proceedings of the fifth International Workshop on Artificial Intelligence and Statistics*, Ft. Lauderdale, FL.

Perrone, M. (1993), Improving regression estimation: averaging methods for variance reduction with extensions to general convex measure optimization, PhD thesis, Brown University, Physics Dept.

Provan, G. M. & Singh, M. (1995), Learning Bayesian networks using feature selection, in D. Fisher & H. Lenz, eds, *Proceedings of the fifth International Workshop on Artificial Intelligence and Statistics*, Ft. Lauderdale, FL, pp. 450–456.

Provost, F. J. (1992), Policies for the Selection of Bias in Inductive Machine Learning, PhD thesis, University of Pittsburgh, Computer Science Department. Report No. 92-34.

Provost, F. J. & Buchanan, B. G. (1995), "Inductive policy: The pragmatics of bias selection", *Machine Learning* **20** pp. 35–61.

Quinlan, J. R. (1986), "Induction of decision trees", *Machine Learning* **1** pp. 81–106. Reprinted in Shavlik and Dietterich (eds.) *Readings in Machine Learning*.

Quinlan, J. R. (1993), *C4.5: Programs for Machine Learning*, Morgan Kaufmann, San Mateo, California.

Quinlan, J. R. (1995), Oversearching and layered search in empirical learning, in C. S. Mellish, ed., *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, Morgan Kaufmann, pp. 1019–1024.

Rendell, L. & Seshu, R. (1990), "Learning hard concepts through constructive induction: framework and rationale", *Computational Intelligence* **6**(4), November, pp. 247–270.

Rosenblatt, F. (1958), "The perceptron: A probabilistic model for information storage and organization in the brain", *Psychological Review* **65** pp. 386–408.

Russell, S. J. & Norvig, P. (1995), *Artificial Intelligence: A Modern Approach*, Prentice Hall, Englewood Cliffs, New Jersey 07632.

Schaffer, C. (1993), "Selecting a classification method by cross-validation", *Machine Learning* **13**(1), pp. 135–143.

Schapire, R. E. (1990), "The strength of weak learnability", *Machine Learning* **5**(2), pp. 197–227.

Siedlecki, W. & Sklansky, J. (1988), "On automatic feature selection", *International Journal of Pattern Recognition and Artificial Intelligence* **2**(2), pp. 197–220.

Singh, M. & Provan, G. M. (1995), A comparison of induction algorithms for selective and non-selective Bayesian classifiers, in *Machine Learning: Proceedings of the Twelfth International Conference*, pp. 497–505.

Skalak, D. B. (1994), Prototype and feature selection by sampling and random mutation hill climbing algorithms, in W. W. Cohen & H. Hirsh, eds, *Machine Learning: Proceedings of the Eleventh International Conference*, Morgan Kaufmann Publishers, Inc.

Street, W. N., Mangasarian, O. L. & Wolberg, W. H. (1995), An inductive learning approach to prognostic prediction, in *Machine Learning: Proceedings of the Twelfth International Conference*.

Taylor, C., Michie, D. & Spiegalhalter, D. (1994), *Machine Learning, Neural and Statistical Classification*, Paramount Publishing International.

Thrun *et al.* (1991), The Monk's problems: A performance comparison of different learning algorithms, Technical Report CMU-CS-91-197, Carnegie Mellon University.

Turney, P. D. (1993), Exploiting context when learning to classify, in P. B. Brazdil, ed., *Proceedings of the European Conference on Machine Learning (ECML)*, pp. 402–407.

Turney, P. D. (1996), The identification of context-sensitive features, a formal definition of context for concept learning, in M. Kubat & G. Widmer, eds, *Proceedings of the Workshop on Learning in Context-Sensitive Domains*, pp. 53–59. Also available as National Research Council of Canada Technical Report #39222.

Utgoff, P. E. (1994), An improved algorithm for incremental induction of decision trees, in *Machine Learning: Proceedings of the Eleventh International Conference*, Morgan Kaufmann, pp. 318–325.

Utgoff, P. E. (1995), Decision tree induction based on efficient tree restructuring, Technical Report 05-18, University of Massachusetts, Amherst.

Vafai, H. & De Jong, K. (1992), Genetic algorithms as a tool for feature selection in machine learning, in *Fourth International Conference on Tools with Artificial Intelligence*, IEEE Computer Society Press, pp. 200–203.

Vafai, H. & De Jong, K. (1993), Robust feature selection algorithms, in *Fifth International Conference on Tools with Artificial Intelligence*, IEEE Computer Society Press, pp. 356–363.

Wolpert, D. H. (1992*a*), "On the connection between in-sample testing and generalization error", *Complex Systems* **6** pp. 47–94.

Wolpert, D. H. (1992*b*), "Stacked generalization", *Neural Networks* **5** pp. 241–259.

Xu, L., Yan, P. & Chang, T. (1989), Best first strategy for feature selection, in *Ninth International Conference on Pattern Recognition*, IEEE Computer Society Press, pp. 706–708.

Yan, D. & Mukai, H. (1992), "Stochastic discrete optimization", *Siam J. Control and Optimization* **30**(3), pp. 594–612.

Yu, B. & Yuan, B. (1993), "A more efficient branch and bound algorithm for feature selection", *Pattern Recognition* **26**(6), pp. 883–889.

Ziarko, W. (1991), The discovery, analysis, and representation of data dependencies in databases, in G. Piatetsky-Shapiro & W. Frawley, eds, *Knowledge Discovery in Databases*, MIT Press.