1 THE WRAPPER APPROACH

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

George H. John
Data Mining
Epiphany Marketing Software
2141 Landings Drive
Mountain View, CA 94043
gjohn@cs.stanford.edu
robotics.stanford.edu/~gjohn

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. The wrapper method searches for an optimal feature subset tailored to a particular algorithm and a domain. We compare the wrapper approach to induction without feature subset selection and to Relief, a filter approach to feature subset selection. Improvement in accuracy is achieved for some datasets for the two families of induction algorithms used: decision trees and Naive-Bayes. In addition, the feature subsets selected by the wrapper are significantly smaller than the original subsets used by the learning algorithms, thus producing more comprehensible models.

1.1 INTRODUCTION

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 de-

Significant parts of this chapter are reprinted from *Artificial Intelligence Journal*, Vol. 97, Nos. 1-2, pp. 273–324, 1997 with kind permission from Elsevier Science – NL, Sara Burgerhartstraat 25, 1055 KV Amsterdam, The Netherlands.

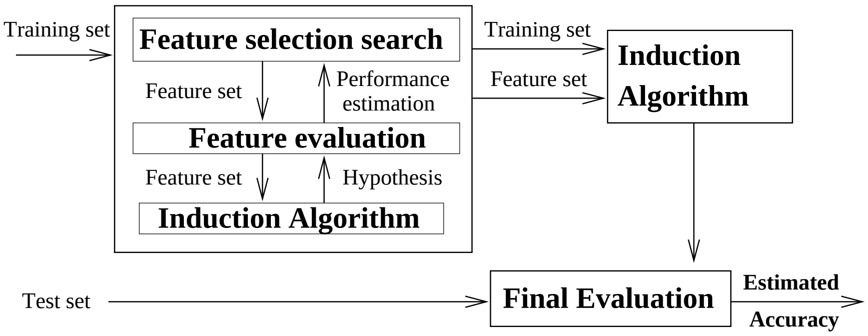

Figure 1.1 The wrapper approach to feature subset selection. The induction algorithm itself is used as a "black box" by the subset selection algorithm.

termined that the patient was suffering from heart disease. The task of the induction algorithm, or the *inducer*, is to induce from training data 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. The idea behind the wrapper approach (John, Kohavi and Pfleger, 1994; Kohavi and John, 1997), shown in Figure 1.1, is simple: the induction algorithm is used as a black box. It is repeatedly run on the dataset using various feature subsets. Some method is used to evaluate its performance on each subset, and 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 use accuracy as our metric in guiding the feature subset selection.

Practical machine learning algorithms—decision tree algorithms such as C4.5 (Quinlan, 1993) and CART (Breiman et al., 1984), and instance-based algorithms such as IBL (Aha et al., 1991)—are known to degrade in prediction accuracy when trained on data containing superfluous features. Algorithms such as Naive-Bayes (Duda and Hart, 1973; Good, 1965; Domingos and Pazzani, 1997) are robust with respect to irrelevant features, but their performance degrades when correlated (even if relevant) features are added. John (1997) shows examples where adding a single irrelevant feature to a credit-approval or diabetes dataset degrades the performance of C4.5 by over 5%. 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.

From a purely theoretical standpoint, the question of which features to use may not be of much interest. A Bayes rule predicts the most probable class for a given instance, based on the full joint probability distribution over the features and the class. The Bayes rule is monotonic, *i.e.*, adding features cannot decrease accuracy, so subset selection is useless.

In practical learning scenarios, however, we are faced with two problems. First, the learning algorithms are not given access to the underlying distribution; rather, they are usually given a relatively small training set. Second, even quite similar algorithms may incorporate different heuristics to aid in quickly building models of the training data—finding the smallest model consistent with the data is NP-hard in many cases (Hyafil and Rivest, 1976; Blum and Rivest, 1992). We define an optimal feature subset with respect to a particular induction algorithm and dataset, taking into account the algorithm's biases and their interaction with the training sample:

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

Thus, by definition, to get the highest possible accuracy, the best subset that a feature subset selection algorithm can select is an optimal feature subset.

This chapter is organized as follows. In Section 1.2, we present definitions of relevance and distinguish between relevance and optimality. In Section 1.3 we describe filter methods of feature subset selection. Section 1.4 describes an algorithm based on the wrapper approach, comparing the algorithm empirically with a filter algorithm. Related and future work are discussed in Sections 1.5 and 1.6, and we present our conclusions in Section 1.7.

1.2 RELEVANCE OF FEATURES

In this section, we define two degrees of relevance: weak and strong. (These and several earlier definitions of relevance are reviewed in Kohavi and John (1997).) We define relevance in terms of a Bayes rule. A feature X is **strongly relevant** if removal of X alone will result in performance deterioration of an optimal Bayes rule. A feature X is **weakly relevant** if it is not strongly relevant, but in some contexts may contribute to prediction accuracy of an optimal Bayes rule. A feature is **relevant** if it is either weakly relevant or strongly relevant; otherwise, it is **irrelevant**.

Definition 2 (Strong relevance)
Let S be the set of all features, and let S_i be $S - \{X_i\}$. A feature X_i is **strongly relevant** *iff there exist values 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 3 (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*

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

Figure 1.2 The feature filter approach, in which the features are filtered independently of the induction algorithm.

For a Bayes rule, the optimal set of features must include all strongly relevant features and possibly some weakly relevant features. However, classifiers induced from real data do not have such nice theoretical properties. Relevance of a feature does not imply that it is in the optimal feature subset for a particular induction algorithm and, somewhat surprisingly, irrelevance does not imply that it should not be in the optimal feature subset (Kohavi and John, 1997).

Example 1 (Optimality does not imply relevance) A feature that always takes the value one is irrelevant by our definitions. But consider a limited Perceptron classifier (Rosenblatt, 1958). It has a weight associated with each feature, and it classifies instances based upon whether the linear combination of weights and feature values is greater than zero. By adding the feature that is always set to one, the limited Perceptron becomes equivalent in representational power to the regular Perceptron (which has an additional parameter allowing an arbitrary threshold, not just zero). However, removal of all irrelevant features would remove that crucial feature.

Although relevance and optimality are not equivalent concepts, the idea that they must be related empirically motivates a set of feature selection methods that measure the relevance of a feature from the data alone.

1.3 THE FILTER APPROACH

Whereas the wrapper approach attempts to identify the best feature subset to use with a particular algorithm, the *filter approach*, which is more common in statistics, attempts to assess the merits of features from the data alone. The filter approach, shown in Figure 1.2, selects features using a preprocessing step, based on the training data. In this section we describe three algorithms from the machine learning literature: FOCUS, Relief, and tree filters.

FOCUS (Almuallim and Dietterich, 1991; Almuallim and Dietterich, 1994), is a feature selection algorithm for noise-free Boolean domains. It exhaustively examines all subsets of features, selecting the minimal subset of features sufficient to determine the label value for all instances in the training set.

The Relief algorithm (Kira and Rendell, 1992; Kononenko, 1994) assigns a "relevance" weight to each feature, which is meant to denote the relevance of the feature to the target concept. The Relief algorithm attempts to find all relevant features. In real domains, many features have high correlations with the label, and thus many are weakly relevant, and will not be removed by Relief. Kohavi and John (1997) discuss our early experiments with Relief and

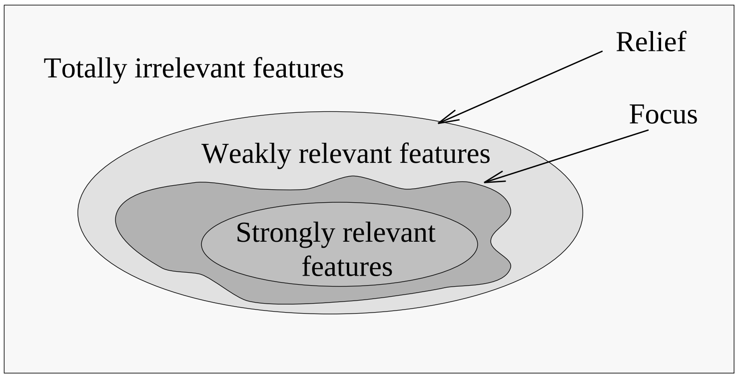

Figure 1.3 A view of feature relevance.

the variant we used, Relieved-F. Relief searches for all the relevant features (both weak and strong).

Tree filters (Cardie, 1993) use a decision tree algorithm to select a subset of features, typically for a nearest-neighbor algorithm. Although they work well for some datasets, they may select bad feature subsets because features that are good for decision trees are not necessarily useful for nearest-neighbor. Also, due to fragmentation, the tree may fail to include relevant features.

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 Prediction Sum of Squares (Neter et al., 1990) have been devised specifically for linear regression. However, these tailored measures would not work well with other algorithms such as Naive-Bayes.

Filter approaches can fail—the Corral artificial dataset from John et al., (1994) is one example. There are 32 instances with six Boolean features plus a Boolean class. The target concept is $(A0 \wedge A1) \vee (B0 \wedge B1)$. A feature named "irrelevant" is uniformly random, and another 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. Although the "correlated" feature is weakly relevant, it is harmful to decision trees. When this feature is removed, the optimal tree is found.

Even a filter that perfectly selected strongly and weakly relevant features would not always work well with Naive-Bayes for example, because in some cases the performance of Naive-Bayes improves with the removal of relevant features. These examples and the discussion of relevance versus optimality above suggest that a feature selection method should take the induction algorithm into account.

1.4 THE WRAPPER APPROACH

In the wrapper approach, shown in Figure 1.1, 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 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, 1995b). 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 and Norvig, 1995).

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 1.4 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.

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. The evaluation function we use is five-fold cross-validation, 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.

1.4.1 A best-first Search Engine

Best-first search (Russell and Norvig, 1995; Ginsberg, 1993) is a robust search method. The idea is to select the most promising node we have generated so far that has not already been expanded. The algorithm varies slightly from the standard version because there is no explicit 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 and Dean, 1989). In practice, we must

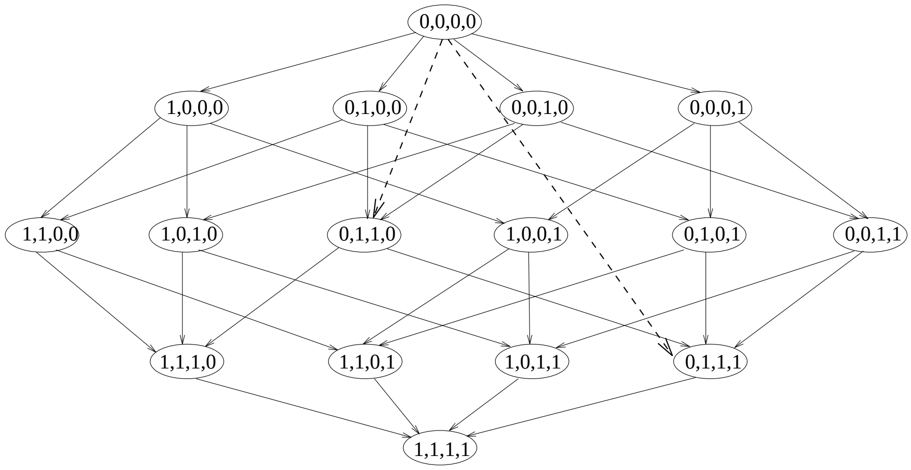

Figure 1.4 The state space search for feature subset selection. Each node is connected to nodes that have one feature deleted or added. The dotted arrows are an enhancement that will be explained in Section 1.4.2.

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 than, say hill-climginb, for feature subset selection. Because of the bias-variance tradeoff (Geman et al., 1992; Kohavi and Wolpert, 1996), it is possible that a more thorough search will increase variance and thus reduce accuracy.

1.4.2 The State Space: Compound Operators

In the previous section, we looked at the search engine. In this section, we look at the topology of the state space and dynamically modify it based on accuracy estimation results. 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.

The motivation for compound operators comes is that the feature subsets can be partitioned 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 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 1.4 depicts a possible set of compound operators for forward selection. 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 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 improve the search by finding nodes with higher accuracy faster. The main advantage of compound operators is that they make backward feature subset selection computationally feasible. 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 in a best-first backward feature subset selection, 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 is still very slow. Compared to the original algorithm, wrapper runs 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.

Table 1.1 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			

1.4.3 Experimental Comparison

We have presented arguments in favor of the wrapper approach above. In this section we show an experimental comparison.

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 1.1 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

Table 1.2 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			

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 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 1.2). 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), 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.

Results for ID3 are detailed in Kohavi and John (1997). 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.

We have focused only on accuracy above, so other criteria merit some consideration. First, the wrapper 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 1.3 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 that the wrapper reduces the number of features used significantly more than Relieved-F.

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 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 and Pazzani (1997). However, we can see that in some real-world domains such as DNA, the feature selection step is important to improve performance.

Table 1.3 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% vs. All	37% vs. All	46% vs. RLF	28% vs. RLF	6% vs. All	19% vs. ID3

1.4.4 Overfitting

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 and 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. For the small sample of 100 instances, the estimated accuracy rose to 76% (26% optimistic) after about 300 node evaluations, indicative of overfitting. Although the theoretical problem exists, our experiments with the wrapper approach indicate that overfitting is mainly a problem when the number of instances is small (Kohavi and Sommerfield, 1995). 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.

1.5 RELATED WORK

The pattern recognition and statistics literature offers many filter approaches for feature subset selection (Devijver and Kittler, 1982; Neter et al., 1990). Sequential backward elimination was introduced by Marill and Green (1963). Most machine learning induction algorithms do not obey the monotonic restrictions that underlie much of the early work in statistics and pattern recognition, and they are applied to databases with a large number of features, so they require special heuristic methods.

Some more recent work in feature selection in the machine learning community includes Langley (1994), which reviews feature subset selection methods in machine learning and contrasted the wrapper and filter approaches. A new filter approach by Koller and Sahami (1996) based on cross-entropy seems to work well in practice. Turney (1996) defines *primary* and *contextual* features, which are related to but different from our ideas of strong and weak relevance.

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 and Buchanan (1995). Moore and Lee (1994) describe an algorithm for feature subset selection that uses a search method motivated by genetic algorithms with leave-one-out cross-validation. Using a clever trick, instead of fully evaluating each node in their search space, they perform partial evaluations of all frontier nodes in parallel in a "race," until one node becomes a clear winner.

Since the introduction of the wrapper approach (John et al., 1994), several authors have experimented with it in various contexts. Langley and Sage (1994) used the wrapper approach to select features for Naive-Bayes. Pazzani (1995) used the wrapper approach to select features and join features (create superfeatures that compound others) for Naive-Bayes and showed that it indeed finds correct combinations when features interact. Singh and Provan (1995) used the wrapper approach to select features for Bayesian networks and showed significant improvements over the original K2 algorithm. Kohavi and John (1997) describe the wrapper with other search methods and as search using probabilistic estimates.

The wrapper idea has been used in several contexts in addition to feature selection. 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 and John (1995). Brunk, Kelly and Kohavi (1997) describe a commercial data mining system, MineSet, that includes the wrapper algorithm for feature subset selection and parameter tuning. 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, similar to the Generalized memory-based learning approach (Moore et al., 1992). Skalak (1994) uses the wrapper approach for selecting a prototype subset to use with nearest-neighbor, in addition to feature selection—an interesting example of choosing training instances as opposed to features.

One might consider the wrapper methods in a larger class of methods that involve running an induction algorithm multiple times to get better results. This class would include ensemble methods such as bagging, boosting, and stacking (cf. Dietterich, 1998). The question arises whether, for a given amount of computation, it would be better to use an ensemble method than a wrapper for feature subset selection. Generally, ensembles produce high accuracy models that are much more complex than a single model produced by one run of an inducer. In contrast, the model produced by running an inducer on the feature subset selected by the wrapper will usually be even simpler than the model produced by the inducer without subset selection. So, if interpretability of the final model is important, using the wrapper for subset selection might be a better choice.

1.6 FUTURE WORK

Many variations and extensions of the current work are possible. In this and previous papers, we have investigated hill-climbing and best-first search engines, starting with either the full or empty subset. Other search methods or initial states might lead to better candidate subsets. The algorithm described in this paper explores one general area of the search space heavily when it is found to be good. It might be worthwhile to introduce some diversity into the search by restarting at random points or perhaps even starting at Exploring the search space more fully could magnify the problems with overfitting, as discussed above. Strategies for evaluating promising states more fully, by doing extra cross-validation runs or possibly using another accuracy estimation method altoghether, might improve results significantly.

Both of these extensions—considering more nodes in the search space, and evaluating candidate nodes more fully—will obviously increase the running time of the wrapper, which can already be very slow, so another area for future work would be runtime performance enhancements. A method similar to Moore and Lee's races would allow the wrapper to waste less time evaluating unpromising nodes. For larger datasets, it is possible to use cheaper accuracy estimation methods, such as holdout, or decrease the number of folds. Even if we continue to do a full cross-validation on every candidate subset, some inducers allow incremental addition and deletion of instances, leading to the possibility of doing incremental cross-validation as suggested in Kohavi, 1995a (), thus drastically reducing the running time. 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 bounded by the number of attributes (assuming enough processors are available).

1.7 CONCLUSION

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 defined two degrees of relevance: weak and strong. We have then shown that these definitions are mainly useful with respect to a Bayes optimal 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). 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.

We compared Relieved-F, a filter algorithm, and our wrapper algorihtm on 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, 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 induction algorithm for this problem out of all the methods used in the StatLog experiments (Taylor et al., 1994). One of the more surprising results was how well Naive-Bayes performed overall: 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.

With both c4.5 and naive-Bayes, Relieved-F degraded the average accuracy on real datasets, whereas the wrapper improved accuracy. Both feature selection algorithms improved average accuracy on the artificial datasets. The wrapper algorithm's accuracy was significantly higher than Relieved-F's in eight cases, and significantly worse on only one. Relieved-F significantly improved upon the plain inducers in 3 cases but was significantly worse in four. The wrapper was significantly better than the plain inducers in nine cases, and worse on only two. The wrapper reduced the number of features significantly more than the filter. These results support or contention that subset selection can improve accuracy, and that a wrapper method should be preferable to a filter. However, we have also shown some problems with the wrapper approach, namely overfitting and the large amounts of CPU time required.

Acknowledgments

Karl Pfleger contributed to our first wrapper paper. This chapter incorporates a number of corrections and improvements suggested by readers of our earlier publications on the wrapper method, especially Pat Langley, Nick Littlestone, Nils Nilsson, and Peter Turney. Dan Sommerfield implemented large parts of the wrapper in $\mathcal{MLC}$++ (Kohavi et al., 1996), which was used for all of the experiments. 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., Kibler, D., and Albert, M. K. (1991). Instance-based learning algorithms. *Machine Learning*, 6(1):37–66.

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

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

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

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

Boddy, M. and Dean, T. (1989). Solving time-dependent planning problems. In Sridharan, N. S., editor, *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, volume 2, pages 979–984. Morgan Kaufmann Publishers, Inc.

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

Brunk, C., Kelly, J., and Kohavi, R. (1997). MineSet: an integrated system for data mining. In Heckerman, D., Mannila, H., Pregibon, D., and Uthurusamy, R., editors, *Proceedings of the third international conference on Knowledge Discovery and Data Mining*, pages 135–138. AAAI Press. `http://www.sgi.com/Products/software/MineSet`.

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

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

Domingos, P. and Pazzani, M. (1997). Beyond independence: Conditions for the optimality of the simple bayesian classifier. *Machine Learning*, 29(2/3):103–130.

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

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

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

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

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

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

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

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

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

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

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

Kohavi, R. and John, G. H. (1997). Wrappers for feature subset selection. *Artificial Intelligence*, 97(1-2):273–324.

Kohavi, R. and 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*, pages 192–197.

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

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

Koller, D. and Sahami, M. (1996). Toward optimal feature selection. In *Proceedings of the Thirteenth International Conference on Machine Learning*, pages 284–292. Morgan Kaufmann Publishers, Inc.

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

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

Langley, P. and Sage, S. (1994). Induction of selective Bayesian classifiers. In *Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence*, pages 399–406, Seattle, WA. Morgan Kaufmann.

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

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

Moore, A. W., Hill, D. J., and Johnson, M. P. (1992). An empirical investigation of brute force to choose features, smoothers and function approximators. In *Computational Learning Theory and Natural Learning Systems Conference*.

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

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

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

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

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

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

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

Singh, M. and 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*, pages 497–505.

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

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

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

Contributing Authors

George H. John is the Data Mining Guru at Epiphany Marketing Software, where he is developing third-generation data mining technology and applications for marekting. He earned a Ph.D. with Distinction from the Computer Science Department at Stanford University, where his research was supported by a National Science Foundation fellowship. His dissertation, *Enhancements to the Data Mining Process*, is available at `http://xenon.stanford.edu/~gjohn`
.