# Filters, Wrappers and a Boosting-Based Hybrid for Feature Selection

**Sanmay Das** SANMAY@EECS.HARVARD.EDU
Division of Engineering and Applied Sciences, Harvard University, Cambridge, MA 02138, USA

## Abstract

In this paper, we examine the advantages and disadvantages of filter and wrapper methods for feature selection and propose a new hybrid algorithm that uses boosting and incorporates some of the features of wrapper methods into a fast filter method for feature selection. Empirical results are reported on six real-world datasets from the UCI repository, showing that our hybrid algorithm is competitive with wrapper methods while being much faster, and scales well to datasets with thousands of features.

## 1. Introduction

A *supervised learning* algorithm receives a set of labeled training examples, each with a feature vector and a class. The presence of irrelevant or redundant features in the feature set can often hurt the accuracy of the induced classifier (John et al., 1994). *Feature selection*, the process of selecting a feature subset from the training examples and ignoring features not in this set during induction and classification, is an effective way to improve the performance and decrease the training time of a supervised learning algorithm. Feature selection typically improves classifier performance when the training set is small without significantly degrading performance on large training sets (Hall, 1999). It is also useful in making the induced concept more comprehensible to humans, since concepts that make use of many features are hard to understand. Feature selection is sometimes essential to the success of a learning algorithm. For example, Kushmerick (1999) points out that it is not feasible to use a nearest-neighbors algorithm on the Internet Advertisements dataset (described later) because of the overabundance of features. Feature selection can reduce the number of features to the extent that such an algorithm can be applied.

Algorithms used for selecting features prior to concept induction fall into two categories. *Wrapper methods* wrap the feature selection around the induction algorithm to be used, using cross-validation to predict the benefits of adding or removing a feature from the feature subset used. *Filter methods* are general preprocessing algorithms that do not rely on any knowledge of the algorithm to be used. There are strong arguments in favor of both methods.

This paper presents a careful analysis of arguments for both methods. It also introduces a new method of feature selection that is based on the concept of boosting from computational learning theory and combines the advantages of filter and wrapper methods. Like filters, it is very fast and general, while at the same time using knowledge of the learning algorithm to inform the search and provide a natural stopping criterion. We present empirical results using two different wrappers and three variants of our algorithm. The experiments use six datasets and three different learning algorithms, namely Naive Bayes (NB), ID3 with $\chi^2$ pruning (ID3), and k-Nearest Neighbors (k-NN).

## 2. Issues in Feature Selection

A feature that is part of the feature subset used by a learning algorithm is a good feature to use if it is either a good predictor of the class by itself, or a good predictor of the class when taken together with some other subset of features in the set. At the same time, it should not be redundant given the other features in the selected feature set. Langley (1994) notes that feature selection algorithms that search through the space of feature subsets must address four main issues: the starting point of the search, the organization of the search, the evaluation of feature subsets and the criterion used to terminate the search. Different algorithms address these issues differently.

It is intractable to look at all possible feature subsets, even if the size is specified. Feature selection algorithms usually proceed greedily. They can be classified into those that add features to an initially empty

set (*forward selection*) and those that remove features from an initially complete set (*backward elimination*). Hybrids both add and remove features as the algorithm progresses. A major problem of forward selection methods is that it is difficult for them to select sets of features that are good *copredictors* of the class if none of these predictors is a good predictor of the class by itself. On the other hand, forward selection is much faster than backward elimination and therefore scales better to large datasets. The major approaches to the problem of when the greedy search should terminate are specifying the size of the feature set to be selected, e.g (Koller & Sahami, 1996), or evaluating the goodness of each feature set in some manner and stopping when further search results in a decrease in goodness.

## 3. Filters vs. Wrappers

Wrapper methods, e.g (John et al., 1994; Langley & Sage, 1994; Caruana & Freitag, 1994, inter alia), search through the space of feature subsets using a learning algorithm to inform the search. They calculate the estimated accuracy of the learning algorithm for each feature that can be added to or removed from the feature subset. Accuracy is estimated using cross-validation on the training set. In forward selection, a wrapper estimates the accuracy of adding each unselected feature to the feature subset and chooses the best feature to add according to this criterion. These methods typically terminate when the estimated accuracy of adding any feature is less than the estimated accuracy of the feature set already selected. Filter methods, e.g (Hall, 1999; Koller & Sahami, 1996, inter alia), on the other hand, select a feature set for any learning algorithm to use when learning a concept from that training set. The biases of the feature selection algorithm and the learning algorithm do not interact. The search proceeds until a pre-specified number of features is selected or some thresholding criterion is met.

A strong argument for wrapper methods is that the estimated accuracy of the learning algorithm is the best available heuristic for measuring the values of features. Different learning algorithms may perform better with different feature sets, even if they are using the same training set. This argument is supported by experiments on the CorrAL dataset (Kohavi, 1995), which consists of 6 Boolean features, labeled $A_0$, $A_1$, $B_0$, $B_1$, $C$ and $I$. The correct concept is $(A_0 \wedge A_1) \vee (B_0 \wedge B_1)$. $I$ is a feature irrelevant to the class, and $C$ a feature 75% correlated with the class label. The decision-tree learning algorithm ID3 (Quinlan, 1986) will typically split on $C$ initially, and may not be able to recover the original concept in the subtrees (Kohavi, 1995). This results in poor performance on the test set. On the other hand, the Naive Bayes classifier performs better when the correlated feature is included because NB is a linear separator unable to represent disjunctions of conjunctions.

The difference in the behavior of the two algorithms suggests that there cannot be a single concept of useful features across different learning algorithms, and the use of the algorithm itself to decide which features to select is important to the success of wrapper methods. However, since cross-validation using the induction algorithm must be performed for each unselected feature every time a feature is added, wrappers are computationally very expensive and do not scale well to large datasets. Further, Hall (1999) cites Kohavi's (1995) finding that cross-validation accuracies have high variance on small training sets as evidence that wrapper methods tend to overfit on small training sets.

The primary advantage of filter methods is their speed and ability to scale to large datasets. The process of feature selection is often most useful in situations in which wrappers may overfit, i.e with small training sets. Given these advantages, we decided to examine four real-world datasets to see how consistently feature sets selected by different wrappers (Naive Bayes and ID3) performed with three different learning algorithms. We hypothesized that the feature sets selected by NB and ID3 wrappers would perform similarly to each other on these datasets for each of the learning algorithms because most real-world datasets are not similar to the CorrAL dataset, which is essentially a thought experiment in finding a dataset for which inducing the simplest correct concept is difficult. Many learning algorithms have trouble with copredictors, disjunctive concepts and redundant and irrelevant features, all of which are features of CorrAL. At the same time, different learning algorithms perform better with different feature sets. Our theory was that, for most real-world datasets, a feature set that allows one algorithm to induce a high-accuracy concept should also allow a different algorithm to induce a high-accuracy concept, even if the feature set selected is not optimal for that algorithm. Further, the accuracy of an algorithm on the dataset should be relatively similar across feature sets selected in different manners. The experimental results presented in section 6 support our hypothesis.

## 4. Bridging the Gap: A More Informed Filter Method

Because of the speed and success of filter methods on many datasets (Koller & Sahami, 1996; Hall, 1999) and our focus on real-world datasets with potentially large numbers of features and small training sets, we decided to develop a filter-based forward selection algorithm that takes a first step towards bridging the gap between filter and wrapper methods for feature selection by incorporating some of the advantages of wrapper methods. Forward selection is appropriate because on most real-world datasets the number of features that is optimal for an induction algorithm to use is a small proportion of the total number of available features. In such situations, forward selection is far less time-consuming than backward elimination.

However, it is difficult for forward selection methods to consistently select copredictors as parts of the feature set. Wrappers using backward search (John et al., 1994) as well as Koller and Sahami's (1996) information theoretic algorithm should succeed in this because they work by eliminating features rather than selecting them. Hall points out that correlation-based feature selection often performs worse than wrappers in situations where there are features that are highly predictive of a small part of the feature set (Hall, 1999). There are clearly cases where some algorithms do better with more features than other algorithms. For example, Naive Bayes performs better on the CorrAL dataset when it uses the redundant but correlated feature. It would be useful for a feature selector to use some knowledge of the learning algorithm to inform the search for the feature subset without exposing one to the problems associated with wrapper methods, namely their massive computational expense, and their tendency to overfit with small training sets.

We begin by describing a pure filter method designed to overcome the second problem stated above — selecting features that are highly predictive of a small part of the instance space. A drawback of our algorithm is that the size of the feature set to select needs to be prespecified. Since different learning algorithms perform better with different-sized feature sets, it would be better to automatically determine the number of features to use. As an improvement, we allow the algorithm to keep adding features as long as the training accuracy of the induction algorithm to be used increases on the training set. Finally, we change the feature selection algorithm to use the actual learning algorithm to guide the search. The algorithm and its variants are described in detail below, and experimental results are presented in Section 6.

### 4.1 Boosted Decision Stumps for Feature Selection

Boosting is a general method for improving the accuracy of a weak learning algorithm. We use the AdaBoost algorithm[1]. The algorithm is run for a number of rounds, say $T$. The set of training examples is treated as a weighted distribution $D_i$ on the $i^{th}$ round, with each example initially given a weight of $\frac{1}{n}$ where $n$ is the number of training examples. The weak learner is trained using the distribution $D_i$. Suppose the error of the hypothesis is $\epsilon_i$. Then let $\alpha_i = \frac{1}{2}\ln(\frac{1-\epsilon_i}{\epsilon_i})$ and set $D_{i+1}(j) = \frac{D_i(j)}{Z_i}e^{-\alpha_i}$ if the hypothesis predicted the class of the $j^{th}$ example correctly, and $D_{i+1}(j) = \frac{D_i(j)}{Z_i}e^{\alpha_i}$ otherwise. Here $Z_i$ is a normalization factor chosen so that $D_{i+1}$ is a distribution. The final hypothesis is the weighted sum of the hypotheses at each round $i$ with the weight given to each of the hypotheses being $\alpha_i$.

We disregard the problem of classifying examples using boosting by using boosting only for feature selection. Our algorithm uses boosted decision stumps as the weak learners. It uses the information gain criterion for deciding which feature to choose. The number of features to be selected is a parameter, say $k$, to the feature selection algorithm. The boosting algorithm is run for $k$ rounds, and at each round all features that have previously been selected are ignored. Hence at each round it looks for the previously unselected feature with the highest information gain on the weighted distribution of training examples. The search for features is greedy. If a feature is selected in any round, it becomes part of the set that will be returned. We refer to this algorithm as BDSFS (Boosted Decision Stump Feature Selection).

Boosted decision stumps have many desirable properties as a method of feature selection. If $k$ features are to be selected, we do not want to select $k$ features that are all meaningful in themselves, but redundant given the presence of a subset of the others. Boosting assigns higher weights to examples that have often been misclassified in the previous rounds. A feature that correctly predicts the class label of examples that the previously selected features often classify wrongly will have a higher information gain. This should eventually help guide the search for features in finding features that are highly predictive of a small region of the instance space get selected, because if that part of the space is often misclassified by other features, those examples will keep increasing in weight. The features

[1]This brief overview of AdaBoost is based entirely on the presentation of Freund and Schapire (1999).

selected by this algorithm are likely to perform well together on the dataset. While this heuristic is not theoretically optimal, it performs well (see Section 6 for experimental results).

### 4.2 Extending BDSFS

The necessity for pre-specification of the number of features to be selected is an undesirable property of BDSFS. As a first step, we extended the feature selection algorithm to use the learning algorithm to provide a stopping criterion. Features are added to the feature set as before, by selecting the feature with maximum weighted information gain. The learning algorithm is trained using this feature in addition to the existing feature set. If training accuracy does not increase, the search stops without adding the last feature to the feature set. This variant of BDSFS, which we shall refer to as BDSFS-2, turns out to be a very good feature selector on the datasets we examined (see Section 6).

We decided to use training accuracy instead of cross-validation in the interests of efficiency. This is justifiable because we are only using it as a stopping criterion, not as a means of evaluating the worth of features. Also, training accuracy is a good measure of whether the set of features selected provides sufficient discriminatory power for the learning algorithm[2].

The second major change we made was to modify the feature selection algorithm to use the learning algorithm in a stronger way to guide the search. We changed the reweighting process so that the weak hypotheses used in each round of the boosting process were the concepts that the learning algorithm would learn from the unweighted training set when using just the features in the feature set thus far. Selection of the next feature to be added is still performed on the basis of weighted information gain, so the algorithm is not subject to the inefficiency of wrappers. Using the learning algorithm itself to identify examples that need to be given more weight is helpful in guiding the search for features to add to the set. The stopping criterion is the same as in BDSFS-2. We call this method Boosting Based Hybrid Feature Selection (BBHFS) because it incorporates many of the desirable features of filters and wrappers discussed in Section 3. Again, the algorithm is very fast and its performance is good in terms of classification accuracy on test data.

[2]Preliminary experiments using cross-validation instead of training accuracy as the stopping criterion did not reveal a significant difference in performance.

*Table 1.* Summary of Datasets Used

| Dataset | Features | Trials | Train | Test |
|---|---|---|---|---|
| Vote | 16 | 100 | 218 | 217 |
| Chess | 32 | 50 | 2131 | 1065 |
| Mushroom | 21 | 50 | 1015 | 7109 |
| DNA | 57 | 100 | 69 | 37 |
| Lymphography | 18 | 100 | 92 | 50 |
| Ads | 1558 | 50 | 500 | 2500 |

## 5. Datasets and Experimental Methodology

### 5.1 Datasets

In this paper we report experiments only on two class problems. Extending boosted decision stumps to handle multi-class datasets is problematic (Drucker, 1997) and this is a potential limitation of our algorithm. We used the Chess Endgames, Congressional Voting, Mushroom, DNA, Lymphography and Internet Advertisement databases from the UCI machine learning repository (Murphy & Aha, 2000). Both the Congressional Voting Records and Mushroom databases have a single highly predictive feature. The Mushroom dataset is large in that it has more than 8000 examples, while the Voting dataset has only 435. On the Chess Endgames dataset, ID3 learns a very accurate concept that uses many of the features. The DNA dataset has 57 features and only 106 examples. The Lymphography dataset, originally a four class problem, was converted to a two class problem by eliminating all 6 examples from two of the classes. The Internet Advertisements (Ads) dataset has 3279 examples and 1558 features, 3 of which are continuous. We discretized these features into 10 bins naively by choosing 10 splits of equal length between the minimum and the maximum values so that the boosted decision stumps algorithm could use discrete features.

One attribute was eliminated from the Mushroom database because of missing values. Table 1 summarizes the number of features in each dataset and the number of trials, training examples and test examples used in most experiments. Any deviations from this are noted with the description of the experiments.

### 5.2 Algorithms, Software and Experimental Methodology

We used Ray Mooney's publicly available library of ML software written in Common Lisp (Mooney, 2000). Modifications were made to enable optional feature selection. The k-Nearest Neighbors algorithm was set to

run with $k = 1$. Naive Bayes uses the Laplace correction instead of 0 for conditionals. ID3 uses $\chi^2$ pruning of the induced decision tree. These three learning algorithms work very differently from each other, and are thus an ideal cross-section of learning algorithms to use for comparison of feature selection techniques.

Each trial was run by independent, random selection of disjoint training and test sets from the entire dataset. We ran 50 trials each for the larger datasets (Mushroom, Chess and Ads) and 100 trials for the smaller ones (DNA, Vote and Lymphography). Wrappers used forward selection, which Hall reports is as good as backward elimination on most datasets (Hall, 1999). Ten-fold cross validation was used in the wrappers for accuracy estimation. Classification accuracy on the test set is reported. Plus-minus figures and error bars reflect the standard deviation of the entire set of trials.

## 6. Experimental Results

In order to empirically test many of the hypotheses about the benefits and disadvantages of filter and wrapper methods, we performed several experiments on real-world datasets. We wanted to compare the performance of filters and wrappers on large and small training sets and evaluate the benefits of using a hybrid algorithm. We begin by describing the results obtained by NB and ID3 wrappers, then discuss the performance of BDSFS and its variants.

### 6.1 Comparing Wrappers

To verify our hypothesis that feature sets that are useful for one algorithm are also useful for others, we decided to examine the performance of the three learning algorithms using the feature sets induced by a Naive Bayes wrapper and an ID3 wrapper. Tables 2, 3 and 4 report on the accuracies achieved by NB, ID3 and k-NN using no feature selection, the NB wrapper and the ID3 wrapper. The results confirm our hypothesis. There is little difference between the accuracies induced by NB and ID3 using feature sets selected by NB and ID3 wrappers. Both NB and ID3 wrappers also perform well as feature selectors for k-NN. While the feature sets selected may not be optimal for the learning algorithm to be used, there is clearly a strong similarity across the feature sets selected by the different algorithms. If filter methods select similar subsets, they could be a feasible alternative to wrapper methods on most real-world datasets.

### 6.2 BDSFS, BDSFS-2 and BBHFS

BDSFS produces feature sets that yield results comparable to, and in some case better than those yielded by wrapper-selected sets for Nearest Neighbor and Naive Bayes classifiers. Table 5 presents results for NB, k-NN and ID3 after feature selection is performed using BDSFS. The second column in the table shows the number of features specified for BDSFS to select. Interestingly, the results show that different algorithms perform better when given different numbers of features. For example, k-NN performs better on the Chess dataset with 6 features than it does with 4, while the reverse is true for Naive Bayes. This suggests that the addition of a learning algorithm-based stopping criterion is important.

Tables 6 and 7 present the better of the two accuracies achieved by BDSFS on the Chess and Mushroom datasets, along with wrapper performance and the performance of BDSFS-2 and BBHFS. Obviously, BDSFS is a competitive alternative to wrappers in terms of performance on these training sets. It is also worth noting that while our results are not directly comparable to those obtained by Hall (1999) and by Koller and Sahami (1996) because of differences in the algorithms and datasets used, the improvements in accuracy we obtain seem comparable to those they obtain with the same family of algorithm (for example, Hall uses IB1 as an instance-based classifier while we use 1-NN).

In general the performance of BDSFS-2 and BBHFS is equivalent and the accuracies they obtain are comparable to the accuracies obtained with simple BDSFS. The one case where there is a significant difference is the performance of ID3 on the Chess dataset. BDSFS-2 and BBHFS significantly outperform BDSFS and forward selection wrappers. This is because the stopping criterion used makes the algorithm select many more features that are useful in discriminating among examples in small portions of the instance space. We see from Table 9 that this is the one case where BBHFS selects a significantly larger set of features than the wrapper. The stopping criterion we use seems to be a good way of deciding when to stop adding features in forward selection. On the other hand, it is unclear whether using the learning algorithm itself for reweighting in boosting is useful. The speedup obtained using BBHFS is also huge (see Table 8)[3].

[3]It is worth noting that the timing results are from very simple implementations, and algorithms like Naive Bayes are suitable for incremental cross-validation, which can be implemented efficiently (Kohavi, 1995).

*Table 2.* Naive Bayes Accuracies

| Dataset | No FS | NB Wrapper | ID3 Wrapper |
|---|---|---|---|
| Vote | 90.68 ± 1.65 | 94.93 ± 0.88 | 95.12 ± 1.44 |
| Chess | 87.32 ± 1.15 | 94.53 ± 0.65 | 94.30 ± 0.51 |
| Mushroom | 98.99 ± 0.20 | 99.61 ± 0.30 | 98.79 ± 0.85 |
| DNA | 86.73 ± 5.46 | 79.84 ± 7.11 | 79.73 ± 7.40 |

*Table 3.* ID3 Accuracies

| Dataset | No FS | NB Wrapper | ID3 Wrapper |
|---|---|---|---|
| Vote | 93.96 ± 1.49 | 95.58 ± 1.01 | 94.82 ± 1.74 |
| Chess | 99.29 ± 0.38 | 93.92 ± 0.63 | 94.34 ± 0.59 |
| Mushroom | 99.77 ± 0.10 | 99.60 ± 0.26 | 99.60 ± 0.35 |
| DNA | 73.30 ± 8.17 | 79.78 ± 8.67 | 78.70 ± 8.64 |

*Table 4.* k-NN Accuracies

| Dataset | No FS | NB Wrapper | ID3 Wrapper |
|---|---|---|---|
| Vote | 92.03 ± 1.66 | 91.06 ± 11.21 | 92.83 ± 7.84 |
| Chess | 89.73 ± 0.57 | 94.43 ± 0.49 | 92.77 ± 4.41 |
| Mushroom | 99.96 ± 0.05 | 99.67 ± 0.22 | 99.70 ± 0.13 |
| DNA | 78.81 ± 6.08 | 72.22 ± 9.73 | 74.27 ± 9.43 |

*Table 5.* Accuracies of Induction Algorithms using BDSFS

| Dataset | No. Feats | NB | k-NN | ID3 |
|---|---|---|---|---|
| Vote | 3 | 94.76 ± 2.60 | 93.10 ± 3.39 | 93.88 ± 3.20 |
| DNA | 7 | 85.54 ± 6.52 | 78.81 ± 6.82 | 79.38 ± 7.62 |
| Chess | 4 | 94.14 ± 0.63 | 89.25 ± 9.40 | 94.05 ± 0.49 |
| Chess | 6 | 92.69 ± 1.63 | 91.68 ± 5.12 | 93.97 ± 0.68 |
| Mushroom | 2 | 98.91 ± 0.24 | 98.89 ± 1.76 | 99.38 ± 0.16 |
| Mushroom | 3 | 97.82 ± 0.24 | 98.94 ± 0.98 | 99.43 ± 0.12 |

*Table 6.* Comparison of Naive Bayes Accuracies

| Dataset | No FS | Wrapper | BDSFS | BDSFS-2 | BBHFS |
|---|---|---|---|---|---|
| Vote | 90.68 | 94.93 | 94.76 | 94.87 ± 2.25 | 94.63 ± 2.50 |
| DNA | 86.73 | 79.84 | 85.54 | 84.70 ± 6.60 | 83.32 ± 7.82 |
| Chess | 87.32 | 94.53 | 94.14 | 94.14 ± 0.60 | 94.06 ± 1.83 |
| Mushroom | 98.99 | 99.61 | 98.91 | 99.05 ± 0.29 | 99.06 ± 0.29 |

*Table 7.* Comparison of ID3 Accuracies

| Dataset | No FS | Wrapper | BDSFS | BDSFS-2 | BBHFS |
|---|---|---|---|---|---|
| Vote | 93.96 | 94.82 | 93.88 | 93.93 ± 2.04 | 93.53 ± 2.57 |
| DNA | 73.30 | 78.70 | 79.38 | 77.78 ± 8.75 | 79.54 ± 8.73 |
| Chess | 99.29 | 94.34 | 94.05 | 99.02 ± 0.69 | 98.56 ± 0.91 |
| Mushroom | 99.77 | 99.60 | 99.43 | 99.68 ± 0.22 | 99.54 ± 0.20 |

*Table 8.* Average Times (in seconds) for a Single Trial

| Dataset | NB | NB-Wrap | BBHFS-NB | ID3 | ID3-Wrap | ID3-BBHFS |
|---|---|---|---|---|---|---|
| Chess | 0.863 | 70.868 | 2.630 | 0.650 | 50.933 | 17.050 |
| Mushroom | 1.366 | 13.963 | 1.153 | 0.431 | 8.713 | 1.401 |

*Table 9.* Median Sizes of Selected Feature Sets

| Dataset | NB-Wrap | BBHFS-NB | ID3-Wrap | ID3-BBHFS |
|---|---|---|---|---|
| Chess | 5 | 3 | 5 | 23 |
| Mushroom | 4 | 2 | 3 | 5 |
| Vote | 1 | 2 | 2 | 8 |
| DNA | 4 | 3 | 4 | 4 |

### 6.3 Small Training Sets

We have established that filters, BDSFS2 and BBHFS are competitive with wrappers on large training sets like the ones we used on the Mushroom and Chess domains. Filter methods and the hybrids also seem to perform better on the one training set that is particularly small, the DNA dataset. To compare the performance of wrappers and BBHFS on small training sets, we used training sets of 100 examples on each of the Chess, Mushroom and Vote domains. We also used another dataset, the Lymphography dataset with 92 training examples, to help clarify the results. 100 trials were run for each experiment. On the chess dataset we used 2000 test examples, on the Mushroom dataset we used 5000 and on the Vote dataset we used 300. Results for Naive Bayes are reported in Table 10 . This table also includes the previous results for the DNA dataset, with 69 training examples.

*Table 10.* Naive Bayes Accuracies With Small Training Sets

| Dataset | No FS | BBHFS | Wrapper |
|---|---|---|---|
| Vote | 90.57 ± 1.16 | 92.25 ± 4.32 | 94.57 ± 1.74 |
| DNA | 86.73 ± 5.46 | 83.32 ± 7.82 | 79.84 ± 7.11 |
| Chess | 80.56 ± 3.41 | 86.74 ± 9.10 | 86.93 ± 11.19 |
| Mushroom | 94.28 ± 1.70 | 95.87 ± 6.58 | 98.01 ± 2.50 |
| Lymph | 82.88 ± 4.76 | 83.22 ± 4.67 | 83.8 ± 4.85 |

Surprisingly, BBHFS performs worse than wrappers on the Mushroom and Vote datasets. So of the five small training set cases we have looked at, BBHFS performs better than the wrapper on the DNA dataset (which has the largest number of features), worse on the Mushroom and Vote datasets, and almost the same on the Chess and Lymphography datasets. These experiments suggest that overfitting may not be a serious problem for wrappers on datasets with single highly predictive features like Vote and Mushroom.

### 6.4 The Internet Advertisements Dataset

The Internet Ads dataset (Kushmerick, 1999) presents a challenge for feature selection algorithms because of its size. It contains 1558 features, 1555 of which are Boolean features indicating the presence or absence of a word. 86% of the examples are negative. We ran experiments with plain Naive Bayes and Naive Bayes using BBHFS as the feature selector (wrapper methods failed completely because of their prohibitive expense) on this dataset using 2500 test examples and between 100 and 500 training examples. Figure 1 shows the results. Feature selection is very useful initially in improving the accuracy of the classifier, but its usefulness in terms of improving accuracy tapers off as the number of examples available to the learning algorithm increase. This result is probably specific to the Naive Bayes algorithm, which is remarkably insensitive with respect to irrelevant features.

## 7. Conclusions and Future Work

There is a debate between supporters of filter and wrapper methods for feature selection. Wrapper methods use the bias of the induction algorithm to select features and generally perform better, especially on datasets in which optimal feature sets are learning algorithm-specific. However, the computational expense of wrapper methods is prohibitive on large datasets. We hypothesized that on most real-world datasets, the advantage of using wrappers is not significant, and feature sets that are good for one learning algorithm will also perform well with different learning algorithms. Experiments using feature sets selected by

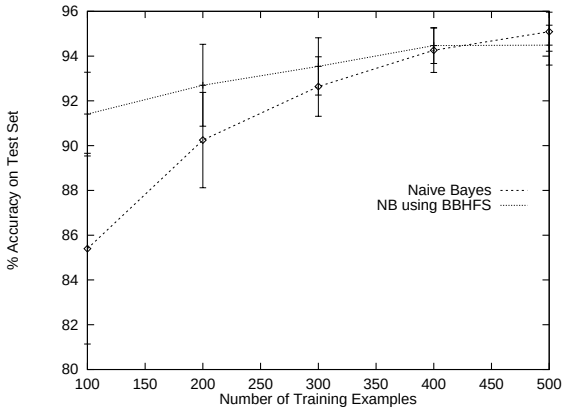


*Figure 1.* Accuracies of NB and NB using BBHFS on the Internet Advertisements dataset

NB and ID3 wrappers for three different learning algorithms verified our hypothesis. Filter methods should perform well on real-world datasets, since they are also capable of identifying good feature sets.

This paper presents an algorithm that uses boosting and incorporates some of the advantages of wrappers, such as a natural stopping criterion, into a filter method, without incurring the computational cost of wrappers. This hybrid algorithm (BBHFS) improves the performance of the learning algorithm significantly in most cases and is competitive with wrapper methods while selecting feature subsets much faster. BBHFS performs better than wrapper methods on the DNA dataset using Naive Bayes and on the Chess dataset using ID3. The Chess dataset results are particularly impressive because the algorithm selects many features which ID3 uses to induce a high-accuracy concept. The search does not select as many features for Naive Bayes, using knowledge of the learning algorithm effectively. BBHFS performs a little worse than wrappers on the small training-set Vote and Mushroom datasets. Results on other small datasets suggest that this surprising result may be related to the presence of a single highly predictive feature in both the Vote and Mushroom datasets. BBHFS scales well to datasets with large numbers of features, as evidenced by experiments with the Internet Advertisements dataset. Using a wrapper method on this dataset, which has 1558 features, is not feasible. BBHFS improves the accuracy of Naive Bayes significantly for small training sets and does not degrade it significantly as the size of the dataset increases.

Filter methods have a great advantage over wrappers because they scale much better to large datasets. However, knowledge of the learning algorithm can be put to good use as in the algorithms presented in this paper. The incorporation of some form of lookahead in forward selection methods like BBHFS would be an important step in overcoming their major shortcoming. One possibility is to learn $k$-level decision trees instead of decision stumps, although considering all possible combinations of $k$ features would increase the running time of the algorithm significantly. A potential limitation of the algorithms presented here in their present form is that boosted decision stumps are problematic in multi-class datasets. Extensions of the algorithms presented here to multi-class problems are an important direction for future research.

## Acknowledgments

I am grateful to Avi Pfeffer, who has been closely involved with this work, and to Charles Elkan and the reviewers for useful comments.

## References

Caruana, R., & Freitag, D. (1994). Greedy attribute selection. *Proceedings of ICML-94*.

Drucker, H. (1997). Fast committee machines for regression and classification. *Proceedings of KDD-97*.

Freund, Y., & Schapire, R. (1999). A short introduction to boosting. *Journal of the Jap. Soc. for AI*.

Hall, M. A. (1999). *Correlation based feature selection for machine learning*. Doctoral dissertation, The University of Waikato, Dept of Comp. Sci.

John, G. H., Kohavi, R., & Pfleger, K. (1994). Irrelevant features and the subset selection problem. *Proceedings of ICML-94*.

Kohavi, R. (1995). *Wrappers for performance enhancement and oblivious decision graphs*. Doctoral dissertation, Stanford University, Comp. Sci. Dept.

Koller, D., & Sahami, M. (1996). Toward optimal feature selection. *Proceedings of ICML-96*.

Kushmerick, N. (1999). Learning to remove internet advertisements. *Proceedings of AGENTS-99*.

Langley, P., & Sage, S. (1994). Induction of selective bayesian classifiers. *Proceedings of UAI-94*.

Mooney, R. (2000). ML software. http://www.cs.utexas.edu/users/ml/mlprogs.html.

Murphy, P., & Aha, D. (2000). UCI ML Repository. http://www.ics.uci.edu/˜mlearn/MLRepository.html.

Quinlan, J. R. (1986). Induction of decision trees. *Machine Learning*, *1*, 81–106.