# Feature Subset Selection: A Correlation Based Filter Approach

**Mark A. Hall, Lloyd A. Smith ([mhall, las]@cs.waikato.ac.nz)**

**Department of Computer Science, University of Waikato, Hamilton, New Zealand.**

## Abstract

*Recent work has shown that feature subset selection can have a positive affect on the performance of machine learning algorithms. Some algorithms can be slowed or their performance adversely affected by too much data some of which may be irrelevant or redundant to the learning task. Feature subset selection, then, is a method for enhancing the performance of learning algorithms, reducing the hypothesis search space, and, in some cases, reducing the storage requirement. This paper describes a feature subset selector that uses a correlation based heuristic to determine the goodness of feature subsets, and evaluates its effectiveness with three common ML algorithms: a decision tree inducer (C4.5), a naive Bayes classifier, and an instance based learner (IB1). Experiments using a number of standard data sets drawn from real and artificial domains are presented. Feature subset selection gave significant improvement for all three algorithms; C4.5 generated smaller decision trees.*

## 1. Introduction

We live in the information age where accumulating data is easy and storing it is relatively inexpensive. Unfortunately, as the amount of machine readable information increases, the ability to understand and make use of it does not keep pace with this growth. Traditional statistical analysis is time consuming as each hypothesis must be formulated and tested individually. Moreover, it requires a background in mathematics in order to understand the results. Machine learning (ML) aims to automate the process of information discovery and be of use to people from a wide range of backgrounds.

ML programs often make assumptions or apply heuristics that trade some accuracy of the resulting model for speed of execution, and comprehensibility of the result. While these assumptions and heuristics are reasonable and often yield good results, under certain conditions the presence of *redundant* and *irrelevant* information they can result in the opposite of their intended effect: slowed execution, less understandable results, and much reduced accuracy.

In the typical supervised machine learning task, data is represented as a table of examples or instances. Each instance is described by a fixed number of measurements or features along with a label that denotes the category (class) the instance belongs to. Feature selectors are algorithms that are applied to the data before it reaches an ML program. Their aim is to reduce the dimensionality of the data by removing the irrelevant and redundant information; thus allowing the ML program to operate more effectively.

This paper describes a feature selector that uses a correlation based heuristic to determine the usefulness of features, and evaluates its effectiveness with three common ML algorithms.

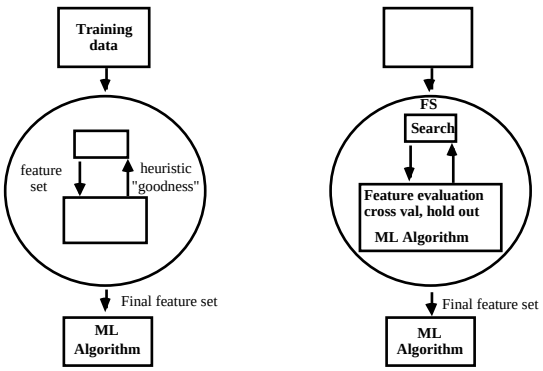
## 2. Feature Selection: Filters and Wrappers

In ML, feature selectors can be characterised by their tie with the induction algorithm that ultimately learns from the reduced data. One paradigm, dubbed the *Filter* [Kohavi and John, 1996], operates independent of any induction algorithm unnecessary attributes are filtered out of the data before induction takes place. Some filter methods strive for consistency in the data that is, they note when every combination of values for a feature subset is associated with a single class label [Almuallim and Deitterich, 1991]. Other filter methods rank features according to a relevancy score [Kira and Rendell, 1992; Holmes and Nevill-Manning,1995]

Another school of thought argues that the bias of a particular induction algorithm should be taken into account when selecting features. This method, dubbed the *Wrapper* [Kohavi and John, 1996], uses the induction algorithm along with a statistical re-sampling technique such as cross-validation (CV) to evaluate feature subsets.

Often, the strengths of the wrapper approach advocated by its proponents are hailed as weaknesses by the filter camp and vice versa. For example, wrappers often achieve better results than filters due to fact that they are tuned to the specific interaction between an induction algorithm and its training data. However, they are much slower than filters as they must repeatedly call the induction algorithm and must be re-run when a different induction algorithm is used.

The feature selector described in this paper is a filter. The author believes that filters will ultimately be feasible in cases where the wrapper is not that is, in real data sets where there may be a large number of features.



### 2.1 Search

Most feature selection algorithms perform a search through the space of feature subsets. Since the size of this space is exponential in the number of features, an exhaustive search is usually intractable. Greedy hill climbing search strategies such as forward selection and backward elimination [Kittler, 1978] are often applied to search the feature subset space in reasonable time. Although simple, these searches often yield good results [Kohavi and John, 1996] comparable to more sophisticated AI search strategies such as Best First search and Beam search [Rich and Knight, 1991].

Greedy hill climbing expands the current node and moves to the child with the highest evaluation. Nodes are expanded by applying search space operators to them (add or delete a single feature for forward selection and backward Elimination respectively).

Table 1: Hill climbing search

1. Let $s \leftarrow$ start state.
2. Expand $s$ by applying search operators.
3. Evaluate each child $t$ of $s$.
4. Let $s' \leftarrow$ child $t$ with highest evaluation $e(t)$.
5. If $e(s') > e(s)$ then $s \leftarrow s'$, go to step 2.
6. Return $s$.

Forward selection starts with an empty set of features; Backward elimination starts with a full set of features. Forward selection will terminate when no child node is better than its parent, while backward elimination will continue to delete features as long as a child is not worse than its parent.

Forward selection and backward elimination are the used in the experiments described in section 4.

## 3. CFS: Correlation-based Feature Selection

Like the majority of feature selection programs, CFS uses a search algorithm along with a function to evaluate the merit of feature subsets. The heuristic by which CFS measures the goodness of feature subsets takes into account the usefulness of individual features for predicting the class label along with the level of intercorrelation among them. The hypothesis on which the heuristic is based can be stated:

> Good feature subsets contain features highly correlated (predictive of) with the class, yet uncorrelated with (not predictive of) each other.

Equation 1 formalises the heuristic.

$$G_s = \frac{k\overline{r_{ci}}}{\sqrt{k + k(k-1)\overline{r_{ii'}}}}$$

—

—

It is, in fact, Pearson s correlation, where all variables have been standardised. The numerator can be thought of as giving an indication of how predictive of the class a group of features are; the denominator of how much redundancy there is among them. The heuristic goodness measure should filter out irrelevant features as they will be poor predictors of the class. Redundant features should be ignored as they will be highly correlated with one or more of the other features.

### 3.1 Feature Correlations

By and large most classification tasks in ML involve learning to distinguish between nominal class values, but may involve features that are ordinal or continuous. A measure based on conditional entropy [Press et al., 1988] is used to measure correlations between features and the class, and between features. Continuous features are first converted to nominal by binning. If $X$ and $Y$ are discrete random variables with respective ranges $R_x$ and $R_y$, Equations 2 and 3 give the entropy of $Y$ before and after observing $X$.

$$H(Y) = -\sum_{y \in R_y} p(y)\log\big(p(y)\big)$$

$$(\ \mid\ ) = -\sum_{x \in R_x} p(x) \sum_{y \in R_y} p(y|x)\log\big(p(y|x)\big)$$

Equation 4 gives a measure of correlation or dependency of $Y$ on $X$. This measure is sometimes called the *uncertainty coefficient* of $Y$ [Press et al., 1988].

$$C(Y|X) = \frac{H(Y) - H(Y|X)}{H(Y)}$$

- Ä IB1: An instance based learner. Classifies a new instance by comparing it with a data base of instances seen during training. The class of the most similar training instance is assigned to the new instance.
- Ä Naive Bayes: A bayesian classifier that assumes *independence* between features in order to simplify the calculation of conditional probabilities.
- Ä C4.5: A decision tree inducer. Builds a decision tree for the training data in a greedy fashion by always testing features that provide the most *gain* in information. Leaves in the tree correspond to classifications.

### 4.1 Domains

Data sets from seven natural domains and three artificial domains were drawn from the UCI repository of machine learning databases. These data sets were chosen because of the prevalence of nominal features and their predominance in the literature. Three additional artificial domains were borrowed from work by Langley and Sage [1994]. They each have 3

relevant features to which a further 27 irrelevant features have been added. These boolean domains exhibit an increasing level of feature interaction. It is expected that CFS will have difficulty dealing with domains with high levels of feature interaction. Under these conditions, the usefulness of individual features is difficult to distinguish.

Table 2: Natural and artificial domains

| Domain | Total # Features | Train / Test |
|---|---|---|
| Mushroom | 22 | 1000 / 7124 |
| Vote | 16 | 218 / 217 |
| Vote1 | 15 | 218 / 217 |
| Credit | 15 | 228 / 462 |
| Lymphography | 18 | 98 / 50 |
| Primary tumor | 17 | 226 / 113 |
| Breast Cancer | 9 | 191 / 95 |
| $x_1x_2x_3$<br>$x_1x_2 \vee x_1x_3 \vee x_2x_3$<br>$x_1x_2x_3 \vee \overline{x_1}\,\overline{x_2}\,\overline{x_3}$ | 30 (27 irrelevant) | 400 / 800 |
| Monks1 | 6 (3 irrelevant) | 124 / 432 |
| Monks2 | 6 | 169 / 432 |
| Monks3 | 6 (3 irrelevant) | 122 / 432 |

### 4.3 Experiment 1: CFS vs No CFS

50 runs were done with and without feature selection for each ML algorithm on each data set. For each run the following procedure was applied:

1) A data set was randomly split into a training and test set (sizes are given in Table 2)
2) Each ML algorithm was applied to the training set; the resulting model was used to classify the test set.
3) CFS was applied to the training data, reducing its dimensionality. The test set was reduced by the same factor. Each ML algorithm was applied to the reduced data as in step 2.
4) Accuracies over 50 runs for each ML algorithm were averaged.

Forward selection and backward elimination searches were used. Results for forward selection are given as there was (on average) little difference between the two search methods.

In addition to accuracy, tree sizes for C4.5 were recorded. Smaller trees are generally preferred as they are easier to understand.

### 4.4 Experiment 2: CFS vs Wrapper

A second experiment was run in order to get a feel for how well CFS performs against the popular Wrapper feature selector. Using the same data sets as in experiment 1, feature selection by the Wrapper (10 fold CV on the train set) was compared with feature selection by CFS using C4.5 as the final induction algorithm. Due to the length of time taken by the Wrapper to perform its search, only two runs were done on each data set: one using a forward selection search, the other a backward elimination search.

## 5. Results

Tables 3 and 4 show the results for IB1 and Naive Bayes, respectively, for experiment 1. Feature selection significantly improves the performance of IB1 on 3 of natural domains and 3 of the artificial domains. Performance is significantly degraded on 2 of the natural and 3 of the artificial domains. CFS has successfully removed the 27 irrelevant attributes from the first two boolean domains (B1 & B2). As expected, CFS is not effective on the third of the boolean domains (B3). Due to the high degree of feature interaction in this domain, none of the relevant features in isolation can be distinguished from the irrelevant ones. Similarly, on the Monks1 and Monks2 domains there are strong feature interactions. As can be seen from the last column of Table 1, CFS has drastically reduced the number of features in each data set. In the Vote domain, a major improvement can be had using only one of the original features.

Table3: Accuracy of IB1 with (IB1-CFS) and without (IB1) feature selection. The p column gives the probability that the observed difference between the two is due to sampling (confidence is 1 minus this probability). Bolded values show where one is significantly better than the other at the 0.95 level. The last column shows the number of features selected (rounded to the nearest feature, minus std. deviations) versus the number features originally present.

| Domain | IB1 | IB1-CFS | p | #f/orig |
|---|---|---|---|---|
| Mush | **99.96 ±0.06** | 98.52 ±0.07 | 0.0 | 1/22 |
| Vote | 92.16 ±1.41 | **95.64 ±0.87** | 0.0 | 1/16 |
| Vote1 | 88.48 ±1.99 | 88.83 ±2.27 | 0.262 | 5/15 |
| Credit | 80.13 ±1.67 | **84.65 ±1.91** | 0.0 | 1/15 |
| Lymph | **78.56 ±6.27** | 75.32 ±5.52 | 0.005 | 6/18 |
| Primary | 36.66 ±3.54 | **37.58 ±3.22** | 0.028 | 11/17 |
| Breast | 70.68 ±3.90 | 70.17 ±3.66 | 0.376 | 3/9 |
| B1 | 86.52 ±0.87 | **100.0 ±0.0** | 0.0 | 3/30 |
| B2 | 72.45 ±1.34 | **100.0 ±0.0** | 0.0 | 3/30 |
| B3 | **79.55 ±1.36** | 74.09 ±3.73 | 0.0 | 7/30 |
| Monks1 | **86.40 ±1.88** | 74.58 ±1.74 | 0.0 | 1/6 |
| Monks2 | **72.22 ±1.78** | 63.72 ±2.89 | 0.0 | 1/6 |
| Monks3 | 91.67 ±1.46 | **96.85 ±0.89** | 0.0 | 2/6 |

Table 4: Accuracy of Naive Bayes with and without feature selection.

| Domain | Naive Bayes | Naive-CFS | p | #f/orig |
|---|---|---|---|---|
| Mush | 94.75 ±0.68 | **98.49 ±0.13** | 0.0 | 1/22 |
| Vote | 90.24 ±1.53 | **95.56 ±1.03** | 0.0 | 1/16 |
| Vote1 | 87.20 ±1.84 | **89.04 ±1.63** | 0.0 | 5/15 |
| Credit | 78.21 ±1.49 | **83.78 ±3.78** | 0.0 | 1/15 |
| Lymph | **81.96 ±4.77** | 79.14 ±5.92 | 0.001 | 6/18 |
| Primary | **46.87 ±3.07** | 45.90 ±3.20 | 0.014 | 11/17 |
| Breast | 71.67 ±3.69 | 71.02 ±3.58 | 0.086 | 3/9 |
| B1 | 96.41 ±1.03 | **100 ±0** | 0.0 | 3/30 |
| B2 | 98.50 ±1.14 | **100 ±0** | 0.0 | 3/30 |
| B3 | 75.10 ±0.98 | **75.64 ±0.66** | 0.0 | 7/30 |
| Monks1 | 72.74 ±1.89 | **75.0 ±0.0** | 0.0 | 1/6 |
| Monks2 | 63.74 ±2.08 | **65.10 ±1.84** | 0.0 | 1/6 |
| Monks3 | 97.08 ±0.57 | 97.16 ±0.39 | 0.359 | 2/6 |

The results of feature selection for Naive Bayes are even more successful. Significant improvement is recorded on 9 of the 13 domains and significant degradation on only 2. Interestingly, small improvement is recorded on those artificial domains with feature interactions. It turns out that

Naive Bayes is unable to learn these concepts and similar accuracy can be achieved through simply predicting the most frequent class value. Because CFS is a filter algorithm, the feature subsets chosen for Naive Bayes are the same as those chosen for IB1.

The results for C4.5 (not shown) are less successful. There were 2 significant improvements and 3 degradations on the natural domains. On the domains where accuracy was degraded, it was by 1 or 2 percent. However, CFS was effective in significantly reducing the size of the trees induced by C4.5 (Table 5).

Table 5: Tree sizes induced by C4.5 before and after feature selection

| Domain | Size-C4.5 | Size-CFS | p |
|---|---|---|---|
| Mush | 25.0 ±5.39 | **10.0 ±0.0** | 0.0 |
| Vote | 7.12 ±2.75 | **3.0 ±0.0** | 0.0 |
| Vote1 | 14.04 ±4.79 | **6.88 ±2.95** | 0.0 |
| Credit | 26.3 ±10.51 | **4.12 ±4.37** | 0.0 |
| Lymph | 19.18 ±5.49 | **14.38 ±6.26** | 0.0 |
| Primary | 65.18 ±10.88 | **51.02 ±11.13** | 0.0 |
| Breast | 16.3 ±11.6 | 14.74 ±11.05 | 0.38 |
| B1 | 7±0 | 7±0 | 1.0 |
| B2 | 11±0 | 11±0 | 1.0 |
| B3 | 49.32 ±31.65 | **6.12 ±10.9** | 0.0 |
| Monks1 | 20.06 ±8.84 | **5 ± 0** | 0.0 |
| Monks2 | 1.28 ±1.98 | 1±0 | 0.322 |
| Monks3 | 13.76 ±2.8 | **12.16 ±1.38** | 0.0 |

### 5.2 Experiment 2

Table 6 compares CFS with the Wrapper. The best result (in terms of the number of improvements) is given by the Wrapper using a backward search. This was expected as the Wrapper is tuned to the specific bias of C4.5. Moreover, it has been shown that the Wrapper using a backward search can identify relevant features even when they interact [Kohavi and John, 1996]. The next best result is given by CFS using a forward search. Table 7 shows the CPU time (in seconds) taken by each feature selector. CFS is many times faster than the Wrapper. As can be seen from the table, the most costly approach is the Wrapper using a backward search.

Table 6: Accuracy of C4.5, C4.5 with CFS forward (fs) and backward (be) search, and C4.5 with Wrapper (CV) forward and backward search.

| Domain | C4.5 | C4.5-CFS-fs | C4.5-CV-fs | C4.5-CFS-be | C4.5-CV-be |
|---|---|---|---|---|---|
| Mush | **99.0** | 98.5 | 98.8 | 98.5 | 98.8 |
| Vote | 92.6 | **94.9** | **94.9** | **94.9** | **94.9** |
| Vote1 | **92.2** | 91.2 | 89.9 | 91.2 | 89.9 |
| Credit | 83.8 | **85.9** | **85.9** | 81.4 | **85.9** |
| Lymph | 76.0 | 74.0 | **78.0** | 74.0 | **78.0** |
| Primary | 38.1 | **40.7** | 38.1 | **40.7** | 39.8 |
| Breast | **74.7** | 73.7 | 73.7 | 73.7 | **74.7** |
| B1 | **100.0** | **100.0** | 87.0 | **100.0** | **100.0** |
| B2 | **100.0** | **100.0** | 74.0 | **100.0** | **100.0** |
| B3 | 87.6 | 75.8 | 75.4 | 75.8 | **100.0** |
| Monk1 | 75.7 | 75.0 | 75.0 | 75.0 | **88.9** |
| Monk2 | 65.0 | **67.1** | **67.1** | **67.1** | 65.3 |
| Monk3 | 97.2 | 97.2 | 97.2 | 97.2 | 97.2 |

Table 7: CPU time taken by feature selectors (Sparc 1000).

| Domain | C4.5-CFS-fs | C4.5-CV-fs | C4.5-CFS-be | C4.5-CV-be |
|---|---|---|---|---|
| Mush | 11 | 1115 | 11 | 13316 |
| Vote | 1 | 258 | 1 | 2611 |
| Vote1 | 2 | 632 | 2 | 1930 |
| Credit | 2 | 260 | 2 | 1284 |
| Lymph | 2 | 708 | 2 | 1763 |
| Primary | 2 | 1765 | 2 | 2479 |
| Breast | 1 | 89 | 1 | 361 |
| B1 | 6 | 297 | 6 | 19198 |
| B2 | 6 | 585 | 6 | 16097 |
| B3 | 9 | 271 | 8 | 18776 |
| Monks1 | 1 | 83 | 1 | 183 |
| Monks2 | 2 | 55 | 2 | 166 |
| Monks3 | 1 | 128 | 2 | 206 |

## 6. Conclusion

This paper presents a feature selection algorithm (CFS) that operates independently of any induction algorithm. Results show that its evaluation heuristic chooses feature subsets that are useful to common machine learning algorithms by improving their accuracy and making their results easier to understand. A comparison with the Wrapper approach to feature selection shows CFS to be many times faster making its application to domains with many features more feasible.

Future work will attempt to better understand why CFS works more effectively on some natural domains than others. Comparing the learning curves produced by ML algorithms on natural domains to those produced on carefully controlled artificial domains may provide some insight. Extension of CFS to deal with feature interactions will also be explored. One approach might be to model higher order dependencies (correlation between pairs of features and the class).

## REFERENCES


[Almuallim and Deitterich, 1991] H. Allmuallim and T.G. Deitterich, Learning with many irrelevant features, *Proc. of the Ninth Nat. Conference on AI,* 47 52, 1991.

[Ghiselli, 1964] E. E. Ghiselli, *Theory of Psychological Measurement,* McGraw-Hill, 1964.

[Holmes and Nevill-Manning, 1995] G. Holmes and C.G. Nevill-Manning, Feature selection via the discovery of simple classification rules, *Proc. Int. Symp. on Intelligent Data Analysis (IDA-95)* , 1995.

[Kira and rendell, 1992] K. Kira and L. Rendell, A practical approach to feature selection, *Proc. of the Ninth Int. Conference on ML,* 249 256, 1992.

[Kittler, 1978] J. Kittler, Feature set search algorithms, in C.H Chen (ed) *Pattern Regognition and Signal Processing* , Sijhoff and Noordhoff, the Netherlands, 1978.

[Kohavi and John, 1996] R. Kohavi and G.H. John, Wrappers for feature subset selection, *AIJ special issue on relevance,* (in press).

[Langley and Sage, 1994] P. Langley and S. Sage, Scaling to domains with irrelevant features, in R. Greiner (ed) *Computational Learning Theory and Natural Learning Systems,* MIT Press, 1994.

[Press *et al.,* 1988] W.H. Press, *Numerical Recipes in C,* Cambridge, 1988.

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