Practical Feature Subset Selection for Machine Learning

Mark A. Hall, Lloyd A. Smith {mhall, las}@cs.waikato.ac.nz
Department of Computer Science, University of Waikato, Hamilton, New Zealand.

Abstract

Machine learning algorithms automatically extract knowledge from machine readable information. Unfortunately, their success is usually dependant on the quality of the data that they operate on. If the data is inadequate, or contains extraneous and irrelevant information, machine learning algorithms may produce less accurate and less understandable results, or may fail to discover anything of use at all. Feature subset selectors are algorithms that attempt to identify and remove as much irrelevant and redundant information as possible prior to learning. Feature subset selection can result in enhanced performance, a reduced hypothesis search space, and, in some cases, reduced storage requirement. This paper describes a new feature selection algorithm that uses a correlation based heuristic to determine the "goodness" of feature subsets, and evaluates its effectiveness with three common machine learning algorithms. Experiments using a number of standard machine learning data sets are presented. Feature subset selection gave significant improvement for all three algorithms.

Keywords: Feature Selection, Correlation, Machine Learning.

1. Introduction

In machine learning, computer algorithms (learners) attempt to automatically distil knowledge from example data. This knowledge can be used to make predictions about novel data in the future and to provide insight into the nature of the target concept(s). The example data typically consists of a number of input patterns or examples of the concepts to be learned. Each example is described by a vector of measurements or *features* along with a label which denotes the category or *class* the example belongs to. Machine learning systems typically attempt to discover regularities and relationships between features and classes in a learning or *training* phase. A second phase called *classification* uses the model induced during learning to place new examples into appropriate classes.

Many factors affect the success of machine learning on a given task. The representation and quality of the example data is first and foremost. If there is much irrelevant and redundant information present or the data is noisy and unreliable, then knowledge discovery during the training phase is more difficult. Feature subset selection is the process of identifying and removing as much of the irrelevant and redundant information as possible. This reduces the dimensionality of the data and allows learning algorithms to operate faster and more effectively. In some cases, accuracy on future classification can be improved; in others, the result is a more compact, easily interpreted representation of the target concept.

This paper presents a new approach to feature selection for machine learning that uses a correlation based heuristic to evaluate the merit of features. The effectiveness of the feature selector is evaluated by applying it to data as a pre-processing step for three common machine learning algorithms.

2. Feature Subset Selection

The feature subset selection problem is well known in statistics and pattern recognition. However, many of the techniques deal exclusively with features that are continuous, or, make assumptions that do not hold for many practical machine learning algorithms. For example, one common assumption (monotonicity) says that increasing the number of features can never decrease performance.

Although assumptions such as monotonicity are often invalid for machine learning, one approach to feature subset selection in machine learning has borrowed search and evaluation techniques from statistics and pattern recognition. This approach, dubbed the *wrapper* [Kohavi and John, 1996] estimates the accuracy of feature subsets via a statistical re-sampling technique (such as cross validation) using the actual machine learning algorithm. The wrapper has proved useful but is very slow to execute as the induction algorithm is called repeatedly.

Another approach (adopted in this paper) to feature subset selection, called the *filter*, operates independently of any induction algorithm—undesirable features are filtered out of the data before induction commences. Filter methods typically make use of all the training data when selecting a subset of features. Some look 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 [Allmuallim and Deitterich, 1991]. Another method [Koller and Sahami, 1996] eliminates features whose information content (concerning other features and the class) is subsumed by some number of the remaining features. Still other methods attempt to rank features according to a relevancy score [Kira and Rendell, 1992; Holmes and Nevill-Manning, 1995]. Filters have proven to be much faster than wrappers and hence can be applied to large data sets containing many features.

2.1 Searching the Feature Subset Space

The purpose of feature selection is to decide which of the initial (possibly large number) of features to include in the final subset and which to ignore. If there are n possible features initially, then there are n^2 possible subsets. The only way to find the best subset would be to try them all—this is clearly prohibitive for all but a small number of initial features.

Various heuristic search strategies such as hill climbing and Best First [Rich and Knight, 1991] are often applied to search the feature subset space in reasonable time. Two forms of hill climbing search and a Best First search were trialed with the feature selector described below; the Best First search was used in the final experiments as it gave better results in some cases. The Best First search starts with an empty set of features and generates all possible single feature expansions. The subset with the highest evaluation is chosen and is expanded in the same manner by adding single features. If expanding a subset results in no improvement, the search drops back to the next best unexpanded subset and continues from there. Given enough time a Best First search will explore the entire search space, so it is common to limit the number of subsets expanded that result in no improvement. The best subset found is returned when the search terminates.

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 with (predictive of) the class, yet uncorrelated with (not predictive of) each other.

Equation 1 formalises the heuristic.

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

k is the number of features in the subset; $\overline{r_{ci}}$ is the mean feature correlation with the class, and $\overline{r_{ii'}}$ is the average feature intercorrelation. Equation 1 is borrowed from test theory [Ghiselli, 1964], where it is used to measure the reliability of a test consisting of summed items from the reliability of the individual items. For example, a more accurate indication of a person's occupational success can be had from a composite of a number of tests measuring a wide variety of traits (academic ability, leadership, efficiency etc), rather than any one individual test which measures a restricted scope of traits.

Equation 1 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. Figure 1 depicts the components of the CFS feature selector.

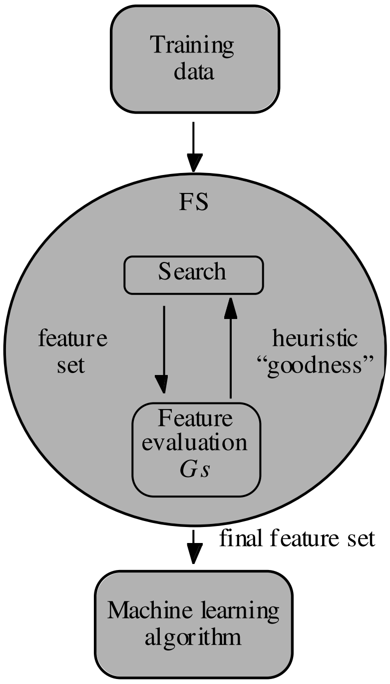

Fig. 1. CFS feature selector.

3.1 Feature Correlations

By and large most classification tasks in machine learning involve learning to distinguish between nominal class values, but may involve features that are ordinal or continuous. In order to have a common basis for computing the correlations necessary for equation 1, continuous features are converted to nominal by binning. A number of information based measures of association were tried for the feature-class correlations and feature intercorrelations in equation 1, including: the uncertainty coefficient and symmetrical uncertainty coefficient [Press *et al.*, 1988], the gain ratio [Quinlan, 1986], and several based on the minimum description length principle [Kononenko, 1995]. Best results were achieved by using the gain ratio for feature-class correlations and symmetrical uncertainty coefficient for feature intercorrelations.

If X and Y are discrete random variables, equations 2 and 3 give the entropy of Y before and after observing X. Equation 4 gives the amount of information gained about Y after observing X (and vice versa—the amount of information gained about X after observing Y). Gain is biased in favour of attributes with more values, that is, attributes with greater numbers of values will appear to gain more information than those with fewer values even if they are actually no more informative. The gain ratio (equation 5) is a non-symmetrical measure that tries to compensate for this bias. If Y is the variable to be predicted, then the gain ratio normalises the gain by dividing by the entropy of X. The symmetrical uncertainty coefficient normalises the gain by dividing by the sum of the entropies of X and Y. Both the gain ratio and the symmetrical uncertainty coefficient lie between 0 and 1. A value of 0 indicates that X and Y have no association; the value 1 for the gain ratio indicates that knowledge of Y completely predicts X; the value 1 for the symmetrical uncertainty coefficient indicates that knowledge of one variable completely predicts the other. Both display a bias in favour of attributes with fewer values.

(Eqn. 2) $$H(Y) = \sum_y p(y)\log_2(p(y))$$

(Eqn. 3) $$H(Y|X) = \sum_x p(x)\sum_y p(y|x)\log_2(p(y|x))$$

(Eqn. 4) $$\begin{aligned} \text{gain} &= H(Y) - H(Y|X) \\ &= H(X) - H(X|Y) \\ &= H(Y) + H(X) - H(X,Y) \end{aligned}$$

(Eqn. 5) $$\text{gain ratio} = \frac{\text{gain}}{H(X)}$$

(Eqn. 6) $$\text{symmetrical uncertainty} = 2.0 \times \left[\frac{\text{gain}}{H(Y)+H(X)}\right]$$

Initial experiments showed CFS to be quite an aggressive filter, that is, it typically filtered out more than half of the features in a given data set—often leaving only the 2 or 3 best features. While this resulted in improvement on some data sets it was clear on others that higher accuracy could be achieved with more features. Reducing the effect of the intercorrelations in equation 1 improves the performance. A scaling factor of 0.25 generally gave good results across the data sets and learning algorithms and was used in the experiments described below.

4. Experimental Methodology

In order to determine whether CFS is of use to common machine learning algorithms, a series of experiments was run using the following machine learning algorithms (with and without feature selection) on 12 standard data sets drawn from the UCI repository.

Naive Bayes

The Naive Bayes algorithm employs a simplified version of Bayes formula to classify each novel instance. The posterior probability of each possible class is calculated given the feature values present in the instance; the instance is assigned the class with the highest probability. Equation 7 shows the Naive Bayesian formula which makes the assumption that feature values are *independent* within each class.

(Eqn. 7) $$P(C_i|v_1, v_2, \ldots, v_n) = \frac{P(C_i)\prod_{j=1}^{n} P(v_j|C_i)}{P(v_1, v_2, \ldots, v_n)}$$

The left side of the equation is the posterior probability of class i given the feature values observed in the instance to be classified. The denominator of the right side of the equation is often omitted as it is a constant which is easily computed if one requires that the posterior probabilities of the classes sum to one. Due to the assumption that feature values are independant with the class, the Naive Bayes classifier's predictive performance can be adversely affected by the presence of *redundant* features in the training data.

C4.5 Decision Tree Generator

C4.5 [Quinlan, 1993] is an algorithm that summarises the training data in the form of a decision tree. Along with systems that induce logical rules, decision tree algorithms have proved popular in practice. This is due in part to their robustness and execution speed, and, to the fact that explicit concept descriptions are produced, which are more natural for people to interpret. Nodes in a decision tree correspond to features, and, branches to their associated values. The leaves of the tree correspond to classes. To classify a new instance, one simply examines the features tested at the nodes of the tree and follows the branches corresponding to their observed values in the instance. Upon reaching a leaf, the process terminates, and the class at the leaf is assigned to the instance.

To build a decision tree from training data, C4.5 employs a greedy approach that uses an information theoretic measure (gain ratio—cf equation 5) as its guide. Choosing an attribute for the root of the tree divides the training instances into subsets corresponding to the values of the attribute. If the entropy of the class labels in these subsets is less than the entropy of the class labels in the full training set, then information has been *gained* through splitting on the attribute. C4.5 chooses the attribute that gains the most information to be at the root of the tree. The algorithm is applied recursively to form sub-trees, terminating when a given subset contains instances of only one class.

C4.5 can sometimes overfit training data, resulting large trees. Kohavi and John (1996) have found in many cases that feature selection can result in C4.5 producing smaller trees.

IB1—Similarity Based Learner

Similarity based learners represent knowledge in the form of specific cases or experiences. They rely on efficient matching methods to retrieve these stored cases so they can be applied in novel situations. Like the Naive Bayes algorithm, similarity based learners are usually computationally simple, and variations are often considered as models of human learning [Cunningham, *et al.*, 1997].

IB1 [Aha *et al.*, 1991] is an implementation of the simplest similarity based learner known as *nearest neighbour*. IB1 simply finds the stored case closest (usually according to some Euclidean distance metric) to the instance to be classified. The new instance is assigned to the retrieved instance's class. Equation 8 shows the distance metric employed by IB1.

(Eqn. 8) $$D(x, y) = \sqrt{\sum_{j=1}^{n} f\left(x_j, y_j\right)}$$

Equation 8 gives the distance between two instances x and y; x_j and y_j refer to the jth feature value of instance x and y respectively. For numeric valued attributes $f\left(x_j, y_j\right) = \left(x_j - y_j\right)^2$, for symbolic valued attributes $f\left(x_j, y_j\right) = \left(x_j \neq y_j\right)$.

Langley and Sage (1994) have found that fewer training cases are needed by nearest neighbour to reach as specified accuracy if *irrelevant* features are removed first.

4.1 Experiment: CFS vs. No CFS

Twelve data sets 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. Table 1 summarises the characteristics of the data sets. Three of the data sets (CR, LY, and HC) contain a few continuous features; the rest contain only nominal features.

50 runs were done with and without feature selection for each machine learning 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 1).

2. Each machine learning 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 machine learning algorithm was applied to the reduced data as in step 2.

4. Accuracies over 50 runs for each machine learning algorithm were averaged.

The best first search was used with a stopping criterion of expanding 5 non-improving nodes in a row. In addition to accuracy, tree sizes for C4.5 were recorded. Smaller trees are generally preferred as they are easier to understand.

5. Results

Tables 2 and 3 show the results of using CFS with Naive Bayes and IB1 respectively. Feature selection significantly improves the performance of Naive Bayes on 7 of the 12 domains. Performance is significantly degraded on 3 domains and there is no change on 2 domains. In most cases CFS has reduced the number of features by a factor of 2 or greater. In the Horse colic domain (HC), a significant improvements can be had using only 2 of the original 28 features.

Table 1. Domain characteristics. The %missing column shows what percentage of the data sets' entries (number of features × number of instances) have missing values. Avg # of feat vals and Max/Min # feat vals are calculated from the nominal features present in the data sets.

Dom	# Inst	# Feat	%Miss	Avg # feat vals	Max/Min # feat vals	# Class vals	Train size/ Test size
MU	8124	23	1.3	5.27	12/1	2	1000/7124
VO	435	17	5.3	2	2/2	2	218/217
V1	435	16	5.5	2	2/2	2	218/217
CR	690	16	0.6	4.44	23/2	2	228/462
LY	148	19	0	2.93	8/2	4	98/50
PT	339	18	3.7	2.18	3/2	23	226/113
BC	286	10	0.3	4.56	11/2	2	191/95
DNA	106	56	0	4	4/4	2	69/37
AU	226	70	2.0	2.23	6/2	24	149/77
SB	683	36	9.5	2.83	7/2	19	450/233
HC	368	28	18.7	23.8	346/2	2	242/126
KR	3196	37	0	2.03	3/2	2	2110/1086

Table 2. Accuracy of Naive Bayes with (Naive-CFS) and without (Naive) 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 versus the number of features originally present.

Domain	Naive Bayes	Naive-CFS	p	# features/original
MU	94.75±0.68	**98.19±0.89**	0.00	09.55±0.83/23
VO	90.25±1.53	**92.08±1.61**	0.00	06.80±1.21/17
V1	87.20±1.84	**87.84±1.83**	0.00	08.20±1.11/16
CR	78.21±1.49	**85.29±2.33**	0.00	01.26±0.49/16
LY	**82.12±4.83**	70.31±10.25	0.00	04.48±2.36/19
PT	**46.87±3.07**	45.86±3.38	0.02	10.60±0.64/18
BC	72.16±2.69	72.55±3.67	0.33	03.62±0.64/10
DNA	89.21±4.97	90.58±4.22	0.06	06.04±2.09/56
AU	**80.24±4.01**	78.59±3.35	0.02	24.80±1.94/70
SB	91.30±1.74	**92.50±2.53**	0.00	24.68±1.27/36
HC	83.13±3.17	**85.83±4.43**	0.00	02.00±1.07/28
KR	87.33±1.16	**94.30±0.54**	0.00	07.32±0.91/37

Table 3. Accuracy of IB1 with (IB1-CFS) and without (IB1) feature selection.

Domain	IB1	IB1-CFS	p	# features/original
MU	99.94±0.07	99.92±0.07	0.19	09.55±0.83/23
VO	92.18±1.34	**93.92±1.68**	0.00	06.80±1.21/17
V1	88.62±1.97	88.20±2.06	0.16	08.20±1.11/16
CR	79.82±1.89	**85.54±1.07**	0.00	01.26±0.49/16
LY	**79.89±5.38**	71.46±9.90	0.00	04.48±2.36/19
PT	39.63±3.44	39.88±2.61	0.53	10.60±0.64/18
BC	71.09±3.83	72.12±3.42	0.10	03.62±0.64/10
DNA	80.31±6.36	**86.58±4.71**	0.00	06.04±2.09/56
AU	75.28±3.21	**76.69±3.44**	0.02	24.80±1.94/70
SB	90.49±1.56	**91.30±1.88**	0.02	24.68±1.27/36
HC				02.00±1.07/28
KR	**94.64±0.78**	94.34±0.52	0.01	07.32±0.91/37

Table 4. Accuracy of C4.5 with (C4.5-CFS) and without (C4.5) feature selection. The first "p" column gives the probability that the observed differences in accuracy between the two are due to sampling. The second "p" column gives the probability that the observed differences in tree size between the two is due to sampling.

Dom	C45	C45-CFS	p	Size	Size CFS	p
MU	99.6±0.4	99.5±0.6	0.3	25.0±5.4	**21.6±5.6**	0.0
VO	95.2±1.4	**95.6±1.2**	0.0	7.1±2.8	**3.6±1.7**	0.0
V1	88.7±2.0	88.3±2.2	0.1	14.0±4.8	**8.3±4.3**	0.0
CR	83.7±1.5	**85.6±1.0**	0.0	26.2±10.4	**3.1±0.9**	0.0
LY	**75.8±5.4**	69.5±9.0	0.0	19.2±5.5	**8.7±3.6**	0.0
PT	41.0±4.4	41.1±3.3	0.8	65.2±10.9	**43.5±7.6**	0.0
BC	71.8±3.3	71.4±3.3	0.5	16.3±11.8	14.1±7.3	0.2
DNA	74.6±6.5	**77.0±6.4**	0.0	16.1±4.1	**13.9±4.6**	0.0
AU	**78.5±3.8**	72.5±3.4	0.0	40.2±5.0	**27.0±5.3**	0.0
SB	89.1±1.6	89.6±2.4	0.2	85.7±7.2	**79.0±6.9**	0.0
HC	84.0±3.0	84.0±4.1	0.9	**13.9±14.0**	37.0±33.0	0.0
KR	**99.2±0.3**	94.4±0.5	0.0	49.9±5.0	**11.0±0.0**	0.0

The results of feature selection for IB1 are similar. Significant improvement is recorded on 5 of the 12 domains and significant degradation on only 2. Unfortunately there is no result on the Horse colic domain due to several features causing IB1 to crash. Because CFS is a filter algorithm, the feature subsets chosen for IB1 are the same as those chosen for Naive Bayes.

Table 4 shows the results for C4.5. CFS has been less successful here than for Naive Bayes and IB1. There are 3 significant improvements and 3 significant degradations.

However, CFS was effective in significantly reducing the size of the trees induced by C4.5 on all but 2 of the domains.

6. Conclusion

This paper has presented a new approach to feature selection for machine learning. The algorithm (CFS) uses features' performances and intercorrelations to guide its search for a good subset of features. Experimental results are encouraging and show promise for CFS as a practical feature selector for common machine learning algorithms. The correlation-based evaluation heuristic employed by CFS appears to choose feature subsets that are useful to the learning algorithms by improving their accuracy and making their results easier to understand.

Preliminary experiments with the wrapper feature selector (same domains and search method) with C4.5 show CFS to be competitive. CFS outperforms the wrapper by a few percentage points on 5 domains; on two domains the wrapper does better by a larger margin (5 and 10%). However, CFS is many times faster than the wrapper. On the Soybean domain (SB) the wrapper takes just over 8 days of CPU time to complete 50 runs on a sparc server 1000; CFS takes 8.5 minutes of CPU time.

The evaluation heuristic (equation 1) balances the predictive ability of a group of features with the level of intercorrelation or redundancy among them. Its success will certainly depend on how accurate the feature-class and feature intercorrelations are. One indication that the bias of the gain ratio and the symmetrical uncertainty coefficient may not be totally appropriate for equation 1 is that reducing the effect of the intercorrelations gave improved results. Both are strongly biased in favour of features with fewer values, the gain ratio increasingly so as the number of class labels increases. Furthermore, both measures are biased upwards as the number of training examples decreases. These factors may account for CFS's poor performance on the Lymphography (LY) and Audiology (AU) domains.

Another factor affecting performance could be the presence of feature interactions (dependencies) in the data sets. An extreme example of this is a parity concept where no single feature in isolation appears better than any other. Domingos and Pazzani (1996) have shown that there exist significant pair wise feature dependencies given the class in many standard machine learning data sets.

Future work will attempt to better understand why CFS works more effectively on some domains than others. Addressing the issues raised above (measure bias and feature interactions) may help on the domains where CFS has not performed as well. Future experiments will look at how closely CFS's evaluation heuristic correlates with actual performance by machine learning algorithms for randomly chosen subsets of features.

References

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

Almuallim, H., Deiterich, T. G. (1991). *Learning with many irrelevant features.* Proceedings of the Ninth National Conference on Artificial Intelligence. San Jose CA, AAAI Press, pp. 47–52.

Cunningham, S. J., Litten, J., Witten, I. H. (1997). *Applications of machine learning in Information retrieval.* Working paper 97/6. C.S department, University of Waikato, New Zealand.

Domingos, P., Pazzani, M. (1996). *Beyond independence: conditions for the optimality of the simple bayesian classifier*. Proceedings of the Thirteenth International Conference on Machine Learning.

Ghiselli, E. E. (1964). *Theory of Psychological Measurement.* McGraw-Hill.

Holmes, G., Nevill-Manning, C. G. (1995). *Feature selection via the discovery of simple classification rules.* Proceedings of the International Symposium on Intelligent Data Analysis (IDA-95).

Kira, K., Renedell, L. (1992). *A practical approach to feature selection*. Proceedings of the Ninth International Conference on Machine Learning. Aberdeen Scotland. Morgan Kaufmann. pp. 249–256.

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

Koller, D., Sahami, M. (1996). *Towards optimal feature selection*. Proceedings of the Thirteenth International Conference on Machine Learning (ICML-96). San Francisco, CA. Morgan Kaufmann. pp. 284-292.

Kononenko, I. (1995). *On biases in estimating multi-valued attributes*. Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence. Morgan Kaufmann. pp. 1034–1040.

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

Press, W. H., Flannery, B. P., Teukolsky, S. A., Vetterling, W. T. (1988). *Numerical Recipes in C.* Cambridge University Press.

Rich, E., Knight, K. (1991). *Artificial Intelligence.* McGraw-Hill.

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

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