Transactions on Information and Communications Technologies vol 19 © 1998 WIT Press, www.witpress.com, ISSN 1743-3517

Experimental Feature Selection using the Wrapper Approach

J. A. Baranauskas & M. C. Monard
Department of Computer Science and Statistics
Institute of Mathematics and Computer Science
University of São Paulo - Brazil
EMail: {jaugusto,mcmonard}@icmsc.sc.usp.br

Abstract

Machine learning methods provide algorithms for mining databases in order to help analyze the information, find patterns, and improve prediction accuracy. In practice, the user of a data mining tool is interested in accuracy, efficiency, and comprehensibility for a specific domain which may be reached through feature selection.

In this work we use the wrapper approach for Feature Subset Selection. The FSS algorithm from $\mathcal{MLC}$++ library was used to run experiments with datasets containing many features. Accuracies for five inducers using all features, features found by FSS as well as the union of all those selected features are presented.

Results confirm the superiority of FSS wrapper approach but in some cases the computational cost is excessive.

1 Introduction

In the last few years, the extraction of higher-level knowledge from low-level data has become a challenge. With huge volumes of data accumulated from several sources, human ability to analyze and understand it is not sufficient or even efficient enough to extract useful knowledge.

New techniques to support the extraction of useful knowledge from the growing volume of data have been developed in Data Mining

— DM — one step in a more complex process called Knowledge Discovery in Databases — KDD[4].

One way to transform raw data into useful knowledge is using Machine Learning — ML — algorithms. Usually in Data Mining several representative samples from the database are taken and presented to a ML algorithm. Then the knowledge extracted by ML algorithms using those samples is combined in some way[3].

In many cases, databases containing large numbers of attributes are becoming more and more common. This is easy to understand, for instance, in the medical diagnosis field where the goal is to learn diagnosis rules for different diseases from several medical records. Generally, these records contain much more information (features) than is really necessary to describe each disease.

In the specific case of learning the diagnosis for cardiac and hormonal diseases, it is known that the relevant features for one disease are not the same for the other. So not to lose any important information, generally all features are used and it is left to the ML algorithm to select the most relevant ones.

It is well known that many ML induction algorithms degrade in performance — accuracy precision and run time — when given too many features. In fact traditional ML algorithms were not developed to deal with high dimensionality. So it is not straightforward using ML algorithms directly in these databases. One possible solution to this problem consists in reducing this dimensionality through feature selection.

Research in ML has sought to automate the selection of features, and many different algorithms have been developed for this purpose. There are, basically, three main approaches that have been pursued: filter, embedded and wrapper[2]. This paper reports the results of experiments measuring the performance of the wrapper feature subset selection on several databases containing many features.

The next sections are organized as follows: Section 2 introduces Feature Subset Selection and the wrapper model for selection of relevant features. Section 3 describes experiments and results using wrapper selection in four real-world datasets with many features. Finally, in Section 4 the conclusions are presented.

2 Feature Subset Selection

There are two aspects to be considered in conceptual learning: which features to use in describing the concept and deciding how to combine those features.

In supervised learning, a learning algorithm (or *inducer*) is given a set of n training examples. Each example X is an element of the set $F_1 \times F_2 \times \ldots \times F_m$, where F_i is the domain of the ith feature. Training examples are tuples $(\mathbf{X}, Y)$ where Y is the label, output or class. The Y values are typically drawn from a discrete set of classes $\{1, \ldots, K\}$ in the case of *classification* or from the real values in the case of *regression*. In this work we will refer to classification. Given a set of training examples, the learning algorithm outputs a *classifier* such that, given a new instance, it accurately predicts the label Y.

One of the central problems is the selection of relevant features and the elimination of irrelevant ones. There are several reasons for doing Feature Subset Selection — FSS. One of them is that they improve accuracy since many induction algorithms degrade in performance when given too many features. Another reason is that FSS improves comprehensibility, which is the ability for humans to understand the data and the classification rules induced by the learning algorithm. Finally, FSS can reduce measurement cost since in some domains measuring features may be expensive.

FSS can be described as a state space search where each node (state) represents a feature subset, the value of a node is the estimated prediction accuracy, and the operators are commonly add/delete feature.

Methods for feature selection that have been developed can be grouped into three classes: those that *embed* the selection within the basic induction algorithm, those that use feature selection to *filter* features during a pre-processing step ignoring the induction algorithm, and those that treat feature selection as a *wrapper* around the induction process, using the induction algorithm as a black box[1].

It should be noted that induction algorithms differ considerably in their emphasis on focusing on relevant features. In this work we will concentrate on the wrapper approach where the induction algorithm is used as a black box. Specifically, we will use the wrapper method implemented in $\mathcal{MLC}$++ where the search is conducted in the space of subsets with add/delete operators using best-first search, and the heuristic for the search is the estimated prediction accuracy using

cross-validation[6].

$\mathcal{MLC}$++ is a library of C++ classes and tools developed at Stanford University[5]. $\mathcal{MLC}$++ provides general machine learning algorithms as well as a wide variety of tools that can be used by end users. Some algorithms support visual output of the classifiers and may generate output for Silicon Graphic's MineSet^{TM} product.

It is considered that the wrapper method should provide a better estimate of accuracy than filter methods since wrappers methods use the same induction algorithm that will be used on the feature subset selected, *i.e.*, they run a search using the inducer itself to determine which attributes in the database are useful for learning.

On the other hand, the computational cost of wrapper methods can be very high since they have to call the induction algorithm for each feature set considered, as can be seen in the next section.

3 Experimental Results

In order to evaluate FSS using the wrapper approach, we ran experiments on four real-world datasets taken from the UCI Data Repository [8]. Table 1 summarizes the datasets descriptions in terms of the total number of instances as well as the number of continuous and discrete attributes used for describing the instances. It also shows, for each domain, the names of the classes and proportion of instances in each class. The datasets have no missing values.

As explained in the previous section, the FSS approach needs one inducer that will be used as a black box for feature selection. We have chosen five inducers for FSS in our experiments: C4.5, CN2, Instance Based (IB), Naive-Bayes (NB) and Table Majority (TM)[7]. To run the experiments each dataset was submitted to the FSS algorithm implemented in $\mathcal{MLC}$++. Since the FSS algorithm looks for the best features for a given inducer, five features subsets were found, one for each of the inducer used in the experiment. Results are shown in Table 2 for each dataset and inducer used, information related to CPU time consumed by FSS using the given inducer, number and proportion of selected features as well as the selected features (features numbering starts at zero) are shown. The CPU time (in seconds) spent to obtain each feature subset is related to a standard *Indigo 2* Silicon Graphics workstation. The table line *union* represents conventional set union of all selected features. The CPU times for the

dna dataset and inducers CN2 and IB are not available since their execution was suspended after more than ten days running in background. In these two cases, we used the best features found by the algorithm up to that moment.

Dataset	# Instances	# Features (continuous,discrete)	Class	Class %	Majority Error
anneal	898	38 (6,32)	1	0.89%	23.83%
			2	11.02%	on value 3
			3	76.17%	
			5	7.46%	
			U	4.45%	
dna	3186	180 (0,180)	1	24.07%	48.09%
			2	24.01%	on value 3
			3	51.91%	
genetics	3190	60 (0,60)	N	51.88%	48.12%
			EI	24.04%	on value N
			IE	24.08%	
sonar	208	60 (60,0)	M	53.37%	46.63%
			R	46.63%	on value M

Table 1: Datasets Descriptions

In Table 3, for instance, the first line (*all features*) shows accuracies obtained using all features (the original dataset) with C4.5, CN2, IB, NB and TM for the anneal dataset. The second line (*FSS(C4.5)*) shows results using only features selected by FSS using C4.5 as inducer and tested with C4.5, CN2, IB, NB and TM and similary to all others rows. Tables 4, 5 and 6 show those results for dna, genetics and sonar datasets, respectively.

After running the FSS algorithm, the accuracy for all features and for each feature subset selected was evaluated using 10-fold stratified cross-validation (each fold contains approximately the same proportion of class labels as the original dataset) against each inducer. The results obtained for each dataset are shown in Tables 3, 4, 5 and 6. Columns represent each inducer used; each row exhibits accuracy and standard deviation of the accuracy using all features, the features selected by FSS using C4.5 as inducer, the features selected by FSS using CN2 as inducer, and so on. In order to verify the inducer behavior accuracy, we decided to include an extra experiment: testing all inducers with the union of all features selected through FSS algorithm using the five algorithms (the *union* line in these tables).

Dataset	FSS Inducer	CPU time (s)	# Features (% total)	Selected Features
anneal	C4.5	3721.0	28 (73.68%)	0 2 3 4 6 7 8 12 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 32 33 35 36 37
	CN2	87607.7	23 (60.53%)	1 2 3 4 7 8 10 11 13 14 15 17 18 19 20 21 22 23 24 25 31 32 34
	IB	4138.5	12 (31.58%)	0 2 4 6 7 8 10 11 12 19 24 32
	NB	187.1	11 (28.95%)	0 2 10 12 15 19 20 23 24 31 33
	TM	565.5	25 (65.79%)	0 1 3 4 5 8 10 11 12 13 16 17 18 19 20 22 23 24 25 28 29 30 35 36 37
	union		37 (97,37%)	
dna	C4.5	47896.5	25 (13.89%)	62 64 71 81 82 83 84 89 92 93 94 95 96 99 104 108 117 119 127 143 146 150 153 165 167
	CN2	N/A	15 (8.33%)	41 54 63 81 83 84 85 89 92 93 94 95 96 99 104
	IB	N/A	11 (6.11%)	81 83 84 89 92 93 94 95 99 104 156
	NB	7966.5	40 (22.22%)	16 18 23 39 45 48 51 53 54 56 57 59 60 62 63 69 71 72 74 75 77 80 81 83 84 89 92 93 94 95 96 97 99 104 118 119 127 133 157 179
	TM	11012.2	10 (5.56%)	81 83 84 89 92 93 94 95 96 104
	union		53 (29.44%)	
genetics	C4.5	8546.3	14 (23.33%)	15 17 23 27 28 29 30 31 33 34 35 45 48 59
	CN2	42479.4	5 (8.33%)	6 28 29 30 31
	IB	36520.4	5 (8.33%)	27 28 29 30 31
	NB	2279.7	25 (41.67%)	0 6 7 11 12 13 17 18 20 22 24 25 27 28 29 30 31 33 34 38 39 43 49 50 59
	TM	1627.1	6 (10.00%)	27 28 29 30 31 34
	union		30 (50.00%)	
sonar	C4.5	569.2	7 (11.67%)	3 10 30 35 45 50 51
	CN2	5726.9	12 (20.00%)	0 10 11 15 22 25 26 29 33 39 45 57
	IB	2161.3	21 (35.00%)	0 3 4 5 7 8 9 11 13 15 29 30 31 34 37 40 41 42 45 52 57
	NB	126.7	10 (16.67%)	5 10 11 15 27 31 35 39 48 52
	TM	83.0	1 (1.67%)	58
	union		33 (55.00%)	

Table 2: Selected Features by FSS Algorithm.

anneal	C4.5	CN2	IB	NB	TB
all features	92.54±1.18	90.42±1.08	99.33±0.25	91.65±0.71	76.84±0.22
FSS(C4.5)	92.55±1.24	89.53±0.76	94.88±0.67	75.06±0.92	85.19±0.59
FSS(CN2)	91.65±1.09	96.44±0.94	98.78±0.31	87.75±0.96	86.41±1.18
FSS(IB)	91.32±1.14	83.63±0.77	99.78±0.15	87.75±0.77	97.55±0.54
FSS(NB)	78.51±0.59	79.30±0.45	90.31±0.58	88.42±0.60	87.31±0.78
FSS(TB)	81.29±0.39	79.29±0.42	99.67±0.24	73.60±1.25	99.67±0.17
union	92.54±1.18	90.42±1.08	99.33±0.25	91.54±0.76	76.84±0.22

Table 3: Accuracy using FSS 10-stratified cross validation folds for the anneal dataset.

dna	C4.5	CN2	IB	NB	TB
all features	92.40±0.46	88.15±0.62	74.23±0.44	94.04±0.34	61.24±0.47
FSS(C4.5)	95.45±0.38	91.94±0.61	88.23±0.58	94.35±0.44	66.32±0.55
FSS(CN2)	94.38±0.38	94.89±0.41	92.62±0.45	94.88±0.44	91.40±0.38
FSS(IB)	93.97±0.34	93.80±0.47	93.72±0.34	94.13±0.51	94.29±0.46
FSS(NB)	93.82±0.35	91.91±0.53	85.31±0.43	96.83±0.23	64.34±0.69
FSS(TB)	94.85±0.35	94.76±0.35	94.07±0.31	94.22±0.37	94.73±0.35
union	94.07±0.39	90.91±0.61	81.39±0.60	96.11±0.25	63.18±0.63

Table 4: Accuracy using FSS 10-stratified cross validation folds for the dna dataset.

As expected, all inducers have the same or better accuracy when using its own subset of features. Another fact that can be observed for these experiments is that features selected by FSS using one specific inducer do not always improve the accuracy of the other inducers. However in the dna dataset, the accuracy was always better when using any subset of selected features to feed each one of the five inducers than the accuracy obtained when all features were used.

An interesting point is related to features union which means that each inducer is allowed to look at FSS relevant features of all inducers including itself. For dna and genetics datasets accuracy was better than using all features for all inducers; for the anneal dataset it was not better only for NB inducer; finally, the sonar dataset was not better with C4.5 and NB inducers.

Figure 1 shows experimental error comparison for each of the five inducers and the four datasets used in the experiments. For each dataset the error related to the accuracy of each inducer (Tables 3, 4, 5 and 6) when all features, features selected by FSS for the same inducer — pairs (FSS(i),i) for i in {C4.5, CN2, IB, NB, TB} from those tables, *i.e.*, diagonal table entries — and the union of these

genetics	C4.5	CN2	IB	NB	TB
all features	94.17±0.39	79.53±1.45	78.75±0.71	95.45±0.32	60.94±0.28
FSS(C4.5)	94.67±0.30	81.54±2.01	86.61±0.54	94.51±0.31	64.23±0.48
FSS(CN2)	89.69±0.50	89.47±0.53	88.71±0.63	89.09±0.58	89.44±0.55
FSS(IB)	91.07±0.48	85.32±2.64	90.88±0.62	90.75±0.55	91.69±0.55
FSS(NB)	93.95±0.50	85.58±2.14	82.60±0.50	96.21±0.26	62.95±0.49
FSS(TB)	93.51±0.46	81.98±2.68	90.38±0.63	93.61±0.54	92.95±0.52
union	94.45±0.40	86.51±1.87	81.79±0.59	95.58±0.31	62.41±0.48

Table 5: Accuracy using FSS 10-stratified cross validation folds for the genetics dataset.

sonar	C4.5	CN2	IB	NB	TB
all features	69.74±1.97	71.19±3.30	85.60±2.00	69.26±4.53	53.38±0.54
FSS(C4.5)	83.19±2.30	75.98±3.00	72.19±2.37	68.36±2.59	53.38±0.54
FSS(CN2)	72.14±3.05	81.32±3.46	84.12±2.03	71.69±2.74	53.38±0.54
FSS(IB)	74.98±3.73	70.28±3.28	92.81±1.27	67.33±2.19	53.38±0.54
FSS(NB)	74.93±2.33	73.98±2.84	84.57±2.77	78.74±3.24	53.38±0.54
FSS(TB)	51.00±1.21	61.50±1.35	55.24±2.35	53.88±2.50	56.24±1.51
union	67.81±3.16	73.16±4.05	86.07±1.65	68.33±2.89	53.38±0.54

Table 6: Accuracy using FSS 10-stratified cross validation folds for the sonar dataset.

features are used.

It can be observed that for each inducer used there is a considerable variation of the error depending on the dataset used. Except for the sonar dataset, C4.5 seems to be the one that better adapt itself for learning from different subset of features;

4 Conclusions

In this work we have used the wrapper approach for Feature Subset Selection. The FSS algorithm from $\mathcal{MLC}$++ library was used to run experiments with datasets containing many features. Accuracies for the five inducers using all features, features found by FSS as well as the union of all those selected features are presented. Results confirm the superiority of the FSS wrapper approach although in some cases the computational cost is excessive.

While most studies of supervised machine learning discuss accuracy on an unseen test set as the performance component, in many cases it is equally or more important to induce comprehensible struc-

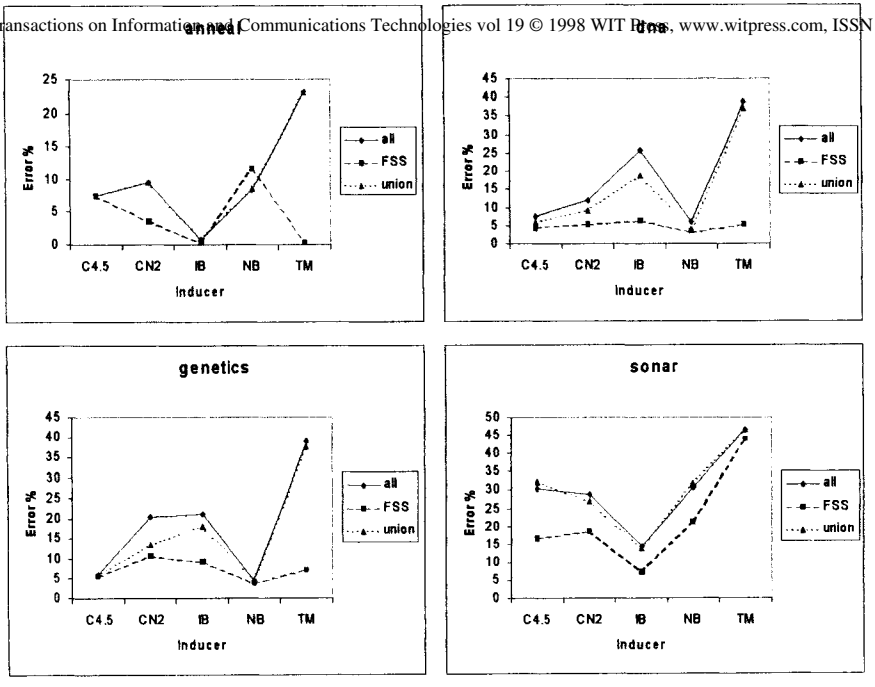

Figure 1: Error Comparison for all Datasets.

tures which give the users new insight regarding this data.

We are currently trying to verify if the rules generated by symbolic algorithms, like CN2 for example, when using all features and the ones selected by FSS improve human comprehensibility.

Acknowledgments: We are grateful to Chandler Caulkins and Jaqueline Brigladori Pugliesi for helpful comments on the draft of this paper. This work was partially supported by National Research Councils CNPq and FINEP.

References

[1] Blum, A.L. & Langley, P., Selection of Relevant Features and Examples in Machine Learning, *Artificial Intelligence*, pp. 245-271, 1997.

[2] Dietterich, T.G., Machine Learning Research: Four Current Directions, Draft of May 23, Oregon State University,

Transactions on Information and Communications Technologies vol 17, © 1998 WIT Press, www.witpress.com, ISSN 1743-3517

[3] Fayyad, U.M., Djorgovski, S.G. & Weir, N., From Digitized Images to On-Line Catalogs: Data Mining a Sky Survey, *AI Magazine*, Vol. 17, No. 2, pp. 51-66, 1996.

[4] Fayyad, U.M., Piatetsky-Shapiro, G. & Smyth, P., The KDD Process for Extracting Useful Knowledge from Volumes of Data, *Communications od the ACM*, Vol. 39, No. 11, pp. 27-34, 1996.

[5] Kohavi, R., John, G., Long, R., Manley, D. & Pfleger, K., $\mathcal{MLC}$++: A Machine Learning Library in *C++*, In *Tools with Artificial Intelligence*, IEEE Computer Society Press, pp. 740-743, 1994.

[6] Kohavi, R. & Sommerfield, D., Feature Subset Selection Using the Wrapper Model: Overfitting and Dynamic Search Space Topology, *Proceedings of the First International Conference on Knowledge Discovery and Data Mining* (KDD-95), Menlo Park, Calif., American Association for Artificial Intelligence, 1995.

[7] Kohavi, R., Sommerfield, D. & Dougherty, J., Data Mining using $\mathcal{MLC}$++: A Machine Learning Library in *C++*, In *Tools with Artificial Intelligence*, IEEE Computer Society Press, pp. 234-245, 1996.

[8] Merz, C.J. & Murphy, P.M., UCI Repository of Machine Learning Datasets, University of California, Irvine, CA, `http://www.ics.uci.edu/~mlearn/MLRepository.html`, 1998.