To appear in Proceedings of IEA-AIE96

FEATURE SELECTION AND CLASSIFICATION – A PROBABILISTIC WRAPPER APPROACH

Huan Liu and Rudy Setiono
Department of Information Systems and Computer Science
National University of Singapore
Kent Ridge, Singapore 119260
Email: {liuh,rudys}@iscs.nus.sg

ABSTRACT

Feature selection is defined as a problem to find a minimum set of M features for an inductive algorithm to achieve the highest predictive accuracy from the data described by the original N features where $M \leq N$. A probabilistic wrapper model is proposed as another method besides the exhaustive search and the heuristic approach. The aim of this model is to avoid local minima and exhaustive search. The highest predictive accuracy is the criterion in search of the smallest M. Analysis and experiments show that this model can effectively find relevant features and remove irrelevant ones in the context of improving the predictive accuracy of an induction algorithm. It is simple, straightforward, and providing fast solutions while searching for the optimal. The applications of such a model, its future work and some related issues are also discussed.

1. INTRODUCTION

The problem of feature selection can be defined as finding relevant M features among the original N attributes, where $M \leq N$, to define the data in order to minimize the error probability or some other reasonable selection criteria. Feature selection has long been the focus of researchers of many fields - pattern recognition, statistics, machine learning (see Section 2). Many methods have been proposed. In general, they can be classified into two categories[1]: (1) the filter approach [AD94, KR92], i.e., the feature selector is independent of a learning algorithm and serves as a filter to sieve the irrelevant and/or redundant features; and (2) the wrapper approach [JKP94], i.e., the feature selector as a wrapper around a learning algorithm relying on which the relevant features are determined. The major advantage to the wrapper model is that it utilizes the induction algorithm itself as a criterion in selecting features since in the context of learning classification rules, the purpose of feature selection is to improve the performance of an induction algorithm. However, incorporating an induction algorithm in the process of feature selection is not without a cost. (More discussion below).

In each category, methods can be further divided into two types: exhaustive or heuristic search. The difficulty about feature selection can be stated as follows: except in a few very special cases, the optimal selection can only be done by testing all possible sets of M features chosen from the N features, i.e., by applying the criterion $\binom{N}{M} = \frac{N!}{M!(N-M)!}$ times. If there are M relevant features, the total number of times is $\sum_{i=1}^{M} \binom{N}{i} = O(N^M)$. This is prohibitive when N and/or M is large. In practice, heuristic methods are the way out of this exponential computation. Heuristic methods in general make use of low order (first or second) information[2] to estimate relevance of features approximately. Although the heuristic methods work reasonably well [Qui93, LW93], it is certain that they miss out the features of high order relations, for example, the parity problem. On one hand, it is a problem of exponential explosion; on the other hand, it is very likely that some relevant features will be omitted if the heuristic approach is taken. Our goal is to have an algorithm that can explore the high order relations among the N features and find M relevant features, with high probability, but without resorting to exhaustive search. In this work, the feature selection problem is redefined in terms of predictive accuracy of an inductive algorithm: to find the smallest set of M features for an inductive algorithm to achieve the highest accuracy.

2. RELATED WORK

The problem of feature selection has long been an active research topic within statistics and pat-

[1] The given references are just recent examples. Both approaches have existed for quite some time. See Section 2.

[2] First order information contains only one feature, second order information two features, etc.

tern recognition [WDJ80, DK82], but most work in this area has dealt with linear regression [Lan94] and under assumptions that do not apply to most learning algorithms [JKP94]. They pointed out that the most common assumption is monotonicity, that increasing the number of features can only improve the performance of a learning algorithm[3].

In the past few years, feature selection has received considerable attention from machine learning and knowledge discovery researchers interested in improving the performance of their algorithms and cleaning data. There are many heuristic feature selection algorithms. The RELIEF algorithm [KR92] assigns a "relevance" weight to each feature, which is meant to denote the relevance of the feature to the target concept. RELIEF samples instances randomly from the training set and updates the relevance values based on the difference between the selected instance and the two nearest instances of the same and opposite classes. According to [KR92], RELIEF assumes two-classes classification problems and does not help with redundant features. If most of the given features are relevant to the concept, it would select most of them even though only a fraction are necessary for concept description. The PRESET algorithm [Mod93] is another heuristic feature selector that uses the theory of Rough Sets to heuristically rank the features, assuming a noise-free binary domain. In order to consider higher order information among the feature. It is suggested in [LW93] to use high order information gain in feature selection. Since the last two algorithms do not try to explore all the combinations of features, it is likely that they fail on the problems like Parity and Majority functions where the combinations of a small number of features do not help in locating the relevant features. Chi2 [LS95] is another heuristic feature selector, it automatically discretizes the continuous features and removes irrelevant continuous features based on the chi-square statistics and the inconsistency found in the data. However, it cannot handle nominal features. In [JKP94], forward selection and backward elimination wrapper models are studied. Nevertheless, no conclusion is given on which one is better and no guideline is offered on which problems which method should be used. It can also be observed from the results that, in general, the latter achieves lower error rates, and the former achieves smaller numbers of features. The classic exhaustive method can be found in [AD94] which is called FOCUS, in which the authors proposed several heuristic versions to speed up the process, assuming a noise-free binary domain.

[3]The monotonicity assumption is not valid for many induction algorithms used in machine learning. See dataset 1 in Section 4 which is reproduced from [JKP94].

Another common understanding is that some learning algorithms have built-in feature selection, for example, ID3 [Qui86], FRINGE [PH90] and C4.5 [Qui93]. The results in [AD94] suggest that one should not rely on ID3 or FRINGE to filter out irrelevant features. Since C4.5 conducts test on each individual feature as well, it is not proper either to use C4.5 to find the minimum set of features. It is expected (to be shown in Section 4) that it will fail on the parity problems.

A summary is that the exhaustive search approach is infeasible in practice; and the heuristic search approach can reduce the computational time significantly, but cannot explore the combined effects among the features, will fail on hard problems (e.g., the parity problem) or cannot remove redundant features. It is right time for the third approach that relies on neither heuristics nor exhaustive search in producing solutions and with high probability, selects the optimal and/or near-optimal set(s) of relevant features.

3. PROBABILISTIC WRAPPER MODEL

This probabilistic approach is a modified version of Las Vegas Algorithms [BB96]. Las Vegas algorithms make probabilistic choices to help guide them more quickly to a correct solution. One kind of Las Vegas algorithms uses randomness to guide their search in such a way that a correct solution is guaranteed even if unfortunate choices are made. As we mentioned earlier, heuristic search methods are vulnerable to the datasets of high order relations. Las Vegas algorithms free us from worrying about such situations by evening out the time required on different situations. The time performance of a Las Vegas algorithm may not be better than that of some heuristic algorithms. With high probability, data that took a long time deterministically are now solved much faster, but data on which the heuristic algorithm was particularly good are slowed down to average by the Las Vegas algorithm. The following algorithm generates random subsets of N features; for each subset $S1$, a learning algorithm is applied to the training data in order to obtain its estimated error rate $err1$, the smallest subset with the lowest error rate is kept. In a Las Vegas algorithm, it is guaranteed that given sufficiently long time, it finds the optimal solution. In search of the minimum set of M features, LVW outputs every current best. As shown in the algorithm, the computation of each $err1$ is based on $S1$ and D_{train}; D_{test} is only used in computing $err2$. That is, at the end, the learning algorithm is applied to both training and testing data and produces its estimated error rate on the testing data. This is the rate reported below in experiments. K is the specified number maximum runs, k is the number of runs, C is the smallest number of features at present, err is the current smallest error rate. Function randomSet produces set $S1$ of features at each run, $C1$ is the number of features in $S1$. LearnAlgo can be any induction algorithm (C4.5 and ID3 are chosen in our experiments).

LVW algorithm
  $err = 0$; $k = 0$; $C = 100$;
  repeat
    $S1$ = randomSet();
    $C1$ = numOfFeatures($S1$);
    $err1$ = LearnAlgo($S1$, D_{train}, NULL);
    if ($err1 < err$ OR
      ($err1 = err$ AND $C1 < C$)) {
      output the current best;
      $k = 0$; $err = err1$;
      $C = C1$; $S = S1$; }
    $k = k + 1$;
  until err is not updated for K times;
  $err2$ = LearnAlgo(S, D_{train}, D_{test});

Some analysis shows that LVW can give a good solution, or an optimal solution if K is sufficiently large. With a good psudo random number generator [PTVF92], the selection of an optimal subset of M features can be considered non-replacement experiments. The probability of finding the optimal subset is $\frac{1}{2^N-k}$ at the $(k+1)$th experiment, although the probability of finding the optimal subset after $(k+1)$ experiments is still $\frac{2^N-1}{2^N} \times \frac{2^N-2}{2^N-1} \times ... \times \frac{1}{2^N-k} = \frac{1}{2^N}$, where N is the number of original features. In our experiments, K (shown in the LVW algorithm) is approximated as $60 \times N$. The larger an N is, the more trials LVW will try. When N is large, this approximation of $K \ll 2^N$. This analysis assumes there is one optimum. If there exist l optima, at the kth tossing, the probability of finding one optimum would be $\frac{l}{2^N-k}$.

4. EMPIRICAL STUDY

Although the induction algorithm in the LVW algorithm can be any kind, it is important that the induction algorithm be fast since the time complexity of LVW is bound by the number of runs and the time complexity of the induction algorithm. Among many choices, we chose C4.5 and ID3 in our experiments. The C4.5 program is the program that comes with Quinlan's book [Qui93]; the ID3 results were obtained by running C4.5 and using the unpruned trees.

Two types of datasets are chosen in experiments. One type is artificial data so that the relevant features are known before feature selection is conducted, which includes CorrAL [JKP94], Monks1-3 [TBB+91], and Parity5+5. The other type is real-world data including Credit, Vote, and Labor [Qui93, MA94]. The choice of these datasets can simplify the comparison of this work with some published work. These datasets were used in [JKP94] in which comparisons with different methods were described. For the artificial datasets, no cross-validation is done. For the real-world datasets, 10-fold cross validation is used to obtain the estimated accuracy on the training data. On the choice of options of C4.5, following [JKP94], we use "-m1" C4.5 flag which indicates that splitting should continue until purity on the artificial datasets, the default setting on the real-world datasets.

Artificial Data:

1. **CorrAL** The data was designed in [JKP94]. There are six binary features, A_0, A_1, B_0, B_1, I, and C. Feature I is irrelevant, feature C is correlated to the class label 75% of the time. The Boolean target concept is $(A_0 \wedge A_1) \vee (B_0 \wedge B_1)$. Both ID3 and C4.5 chose feature C as the root. This is an example of datasets in which if a feature like C is removed, a more accurate tree will result.
2. **Monk1, Monk2, Monk3** The datasets were taken from [TBB+91]. They have six features. The training datasets provided were used for feature selection. Monk1 and Monk3 only need three features to describe the target concepts, but Monk2 requires all the six. The training data of Monk3 contains some noise. These datasets can be used to show that a feature selector selects either only the relevant features or the relevant ones plus others.
3. **Parity5+5** The target concept is the parity of five bits. The dataset contains 10 features, of which 5 are uniformly random (irrelevant). The training set contains 100 instances randomly selected from all 1024 instances. Another independent 100 instances are drawn to form the

testing set. Most heuristic feature selectors will fail on this sort of problems since an individual feature does not mean anything.

Real-World Data:

4. **Vote** This dataset includes votes from the U.S. House of Representatives Congress-persons on the 16 key votes identified by the Congressional Quarterly Almanac Volume XL. The data set consists of 16 features, 300 training instances and 135 test instances.
5. **Credit** (or CRX) The dataset contains instances for credit card applications. There are 15 features and a Boolean label. The dataset was divided by Quinlan [Qui93] into 490 training instances and 200 test instances.
6. **Labor** The dataset contains instances for acceptable and unacceptable contracts. It is a small dataset with 16 features, a training set of 40 instances, and a testing set of 17 instances.

Results are shown in Tables 1 and 2. In the column of Err, $x(p\%)$ means that there are x instances misclassified, the percentage is p. For all the artificial datasets, LVW did find all the relevant features. For example, LVW rediscovered that features 1, 2 and 5 are relevant for Monk1, all six features for Monk2, features 2, 4 and 5 for Monk3, five features for Parity5+5, features A_0, A_1, B_0, B_1 for CorrAL. These results give us confidence on LVW that using accuracy as a criterion, relevant features can be selected even in the presence of noise (e.g., Monk3). For the real-world datasets, there is no knowledge about which features are relevant. However, the comparison between with and without LVW can still be performed along three dimensions: (1) tree size, (2) error rate (Err), and (3) number of features used (# Att). The results of ID3 and C4.5 with and without LVW are summarized in terms of the three dimensions. For ID3, all the figures improved after LVW is applied, i.e., except for Monk2, the number of features is reduced, tree size is smaller, and error rate is decreasing. For C4.5, the improvement is not clear-cut. Although all datasets but Monk2 have their number of features reduced, the significantly decreased error rates for CorrAL, Monk1, and Parity5+5 come with an increase in tree size. This is not without a reason (see discussion below).

The experimental results from [JKP94] are reproduced here in Table 3 for a reference purpose. See more details in the paper. Before (Bf) means *before feature selection*, Forward (Fw) means *forward stepwise selection*, Backward (Bw) means *backward stepwise selection*, Relieve (Rl) is a modified version of Relief [KR92], because of significant variance in the relevance rankings given by Relief [JKP94].

5. CONCLUSION

In a wrapper model, feature selection is closely linked to an induction algorithm, in other words, LVW is only constrained by the limitations of the induction algorithm. If the induction algorithm can handle noisy data, missing values, both continuous and discrete values, so can LVW. In this work, C4.5 is used and no special effort is needed to tailor the datasets in order to run LVW. To achieve the lowest possible error rate is the aim of both the feature selector and the induction algorithm. In addition, LVW produces intermediate solutions while working toward better ones.

A general belief is that the fewer features, the simpler the decision trees since irrelevant features are removed. However, Murphy and Pazzani [MP94] find that the smallest trees typically have lower predictive accuracy than slightly larger trees; exhaustive search for the simplest consistent theories does not necessarily lead to improvement. That means that using accuracy as a criterion may not lead to the simplest tree. This is truly reflected in the results of these datasets. The size of a decision tree is the number of leaves of the tree plus 1. The size measure does not show how many features are contained in the tree. Our experimental results show that in pursuit of high accuracy, LVW can reduce the number of features, improve the accuracy, but may not always reduce the tree size. A measure which combines the three dimensions may help in achieving high performance in all the dimensions.

Our experience with LVW is that it is slow in running many trials ($O(K)$) of different patterns. Since every random pattern should be tested by the induction algorithm, if its time cost is C_{Ind}, the minimum cost of LVW is $O(K * C_{Ind})$. For cross validations, this cost should be increased by another factor related to the number of folds of cross validations.

Due the slowness of LVW, it is not recommended to use it in applications where time is a critical factor. If it is used just once and for all for some period of time, the slowness does not do much harm. Researchers have been trying to speed up the wrapper approach [Lan94]. LVW can play a role of a bench mark for comparisons with other heuristic methods. All the heuristic FS algorithms are deterministic. Heuristic algorithms designed for particular applications can run very fast. LVW can be used to check

Table 1: Results of tree size, error rate (Err) and number of features (# Att.) for ID3 with/without LVW.

Learner	ID3					
Measure	Size		Err		# Att.	
LVW	w/o	with	w/o	with	w/o	with
CorrAL	13	13	0(0.0%)	0(0.0%)	6	4
Monk 1	43	12	101(23.4%)	12(2.8%)	6	3
Monk 2	174	174	131(30.3%)	131(30.3%)	6	6
Monk 3	42	19	42(9.7%)	0(0.0%)	6	3
P5+5	87	63	0(0.0%)	0(0.0%)	10	5
Vote	25	7	7(5.2%)	4(3.0%)	16	4
Credit	117	76	39(19.5%)	31(15.5%)	15	6
Labor	12	7	3(17.6%)	3(17.6%)	16	4

Table 2: Results of tree size, error rate (Err) and number of features (# Att.) for C4.5 with/without LVW.

Learner	C4.5					
Measure	Size		Err		# Att.	
LVW	w/o	with	w/o	with	w/o	with
CorrAL	7	9	2(12.5%)	1(6.2%)	6	4
Monk 1	18	38	105(24.3%)	24(5.6%)	6	3
Monk 2	46	46	148(34.3%)	148(34.3%)	6	6
Monk 3	12	12	12(2.8%)	12(2.8%)	6	3
P5+5	19	63	22(22.0%)	0(0.0%)	10	5
Vote	7	7	4(3.0%)	4(3.0%)	16	4
Credit	44	30	40(20.5%)	30(15.0%)	15	6
Labor	7	7	3(17.6%)	3(17.6%)	16	4

if a fast solution is also a good one when designing a heuristic algorithm.

It is noticed that the slowness is caused by the learning algorithm. This significantly limits the application of this probabilistic model. It is our current interests to find a criterion that does not rely on the accuracy of a learning algorithm. If checking the satisfaction of the new criterion can be made faster, the probabilistic model can be applied to more applications. Hence, one line of our current research is to get rid of the wrapper approach and go for the filter one (there are other reasons in addition to the speed issue). We have been trying to find such a criterion whereby the speed of the probabilistic method can be improved significantly without sacrificing the end results.

ACKNOWLEDGEMENTS

Thanks to the two anonymous referees for their comments on an earlier version of this paper and Y.C. Chew for transforming the graphs in [JKP94] into Table 3.

REFERENCES

[AD94] H. Almuallim and T.G. Dietterich. Learning boolean concepts in the presence of many irrelevant features. *Artificial Intelligence*, 69(1-2):279–305, November 1994.

[BB96] G. Brassard and P. Bratley. *Fundamentals of Algorithms*. Prentice Hall, New Jersey, 1996.

[DK82] P.A. Devijver and J. Kittler. *Pattern Recognition: A Statistical Approach.* Prentice Hall International, 1982.

[JKP94] G.H. John, R. Kohavi, and K. Pfleger. Irrelevant feature and the subset selection problem. In *Machine Learning: Proceedings of the Eleventh International Conference*, pages 121–129. Morgan Kaufmann Publisher, 1994.

[KR92] K. Kira and L.A. Rendell. The feature selection problem: Traditional methods and a new algorithm. In *AAAI-92, Proceedings Ninth National Conference on Artificial Intelligence*, pages 129–134. AAAI Press/The MIT Press, 1992.

[Lan94] P. Langley. Selection of relevant features in machine learning. In *Proceedings of the AAAI Fall Symposium on*

Table 3: Experimental results reported in John et al 94: Bf - Before, Fw - Forward, Bw - Backward, Rl - Relieve. X means the figure is not available in the original paper.

ID3 Algorithm									
	TreeSize			ErrorRate(%)			Attributes		
Dataset	Bf	Fw	Bw	Bf	Fw	Bw	Bf	Fw	Bw
CorrAL	X	X	X	X	X	X	X	X	X
Monk3*	X	X	X	X	X	X	X	X	X
Parity5+5	109	13	63	49	50	0	10	3	5
Vote	25	13	37	5	4	2	16	3	15
Credit	117	66	98	20	21	19	15	4	13
Labor	12	7	12	18	18	18	16	3	12

C4.5 Algorithm												
	TreeSize				ErrorRate(%)				Attributes			
Dataset	Bf	Fw	Bw	Rl	Bf	Fw	Bw	Rl	Bf	Fw	Bw	Rl
CorrAL	11	5	13	5	19	19	0	19	6	2	5	5
Monk3*	8	13	8	6	1	2	1	2	6	3	2	2
Parity5+5	X	X	X	X	X	X	X	X	X	X	X	X
Vote	7	4	7	7	3	3	3	3	16	1	15	15
Credit	44	16	44	41	21	19	21	18	15	3	14	14
Labor	X	X	X	X	X	X	X	X	X	X	X	X

Relevance. AAAI Press, 1994.

[LS95] H. Liu and R. Setiono. Chi2: Feature selection and discretization of numeric attributes. In *Proceedings of the 7th IEEE International Conference on Tools with Artificial Intelligence*, 1995.

[LW93] H. Liu and W.X. Wen. Concept learning through feature selection. In *Proceedings of First Australian and New Zealand Conference on Intelligent Information Systems*, 1993.

[MA94] P.M. Murphy and D.W. Aha. UCI repository of machine learning databases. FTP from ics.uci.edu in the directory pub/machine-learning-databases, 1994.

[Mod93] M Modrzejewski. Feature selection using rough sets theory. In P.B. Brazdil, editor, *Proceedings of the European Conference on Machine Learning*, pages 213–226, 1993.

[MP94] P.M. Murphy and M.J. Pazzani. Exploring the decision forest: An empirical investigation of occam's razor in decision tree induction. *Journal of Art. Intel. Res.*, 1:257–319, March 1994.

[PH90] G. Pagallo and D. Haussler. Boolean feature discovery in empirical learning. *Machine Learning*, 5:71–99, 1990.

[PTVF92] W.H. Press, S.A. Teukolsky, W.T. Vetterling, and B.P. Flannery. *Numerical Recipes in C.* Cambridge University Press, Cambridge, 1992.

[Qui86] J.R. Quinlan. Induction of decision trees. *Machine Learning*, 1(1):81–106, 1986.

[Qui93] J.R. Quinlan. *C4.5: Programs for Machine Learning.* Morgan Kaufmann, 1993.

[TBB+91] S.B. Thrun, et al. The monk's problems: A performance comparison of different learning algorithms. Technical Report CMU-CS-91-197, Carnegie Mellon University, December 1991.

[WDJ80] N. Wyse, R. Dubes, and A.K. Jain. A critical evaluation of intrinsic dimensionality algorithms. In E.S. Gelsema and Kanal L.N., editors, *Pattern Recognition in Practice*, pages 415–425. Morgan Kaufmann Publishers, Inc., 1980.