A Probabilistic Approach to Feature Selection – A Filter Solution

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

Abstract

Feature selection can be defined as a problem of finding a minimum set of M relevant attributes that describes the dataset as well as the original N attributes do, where $M \leq N$. After examining the problems with both the exhaustive and the heuristic approach to feature selection, this paper proposes a probabilistic approach. The theoretic analysis and the experimental study show that the proposed approach is simple to implement and guaranteed to find the optimal if resources permit. It is also fast in obtaining results and effective in selecting features that improve the performance of a learning algorithm. An on-site application involving huge datasets has been conducted independently. It proves the effectiveness and scalability of the proposed algorithm. Discussed also are various aspects and applications of this feature selection algorithm.

1 Introduction

The problem of feature selection can be defined as finding M relevant attributes among the N original attributes, where $M \leq N$, to describe 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) the filter approach [Almuallim and Dietterich, 1994; Kira and Rendell, 1992], i.e., the feature selector is independent of a learning algorithm and serves as a filter to sieve the irrelevant and/or redundant attributes; and (2) the wrapper approach [John *et al.*, 1994], i.e., the feature selector works as a wrapper around a learning algorithm relying on which the relevant attributes are determined. Although the wrapper approach has certain advantages, it is not as general as the filter approach because (1) any learning algorithm is biased, choosing relevant attributes according to a particular learning algorithm is equivalent to changing the data to fit the learning algorithm, (2) the wrapper approach is restricted by the time complexity of the learning algorithm [Langley, 1994], and (3) when the dataset is too large, it may cause a problem in running some learning algorithms - recall that one of the purposes of applying feature selection is to reduce the data. In addition, it may be impractical to employ computationally intensive learning algorithms such as neural nets or genetic algorithms. Furthermore, a good feature selector provided by the filter approach can always be used in the wrapper approach due to the former's independence of any learning algorithm. Not vice versa, though. Therefore, this paper adopts the *filter* approach. In each category, feature selection methods can be further divided into two types: exhaustive or heuristic search. The difficulty of feature selection can be explained 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 attributes, i.e., by applying the criterion $\binom{N}{M} = \frac{N!}{M!(N-M)!}$ times. If there are M relevant attributes, the total number of times is $\sum_{i=0}^{M} \binom{N}{i} = O(N^M)$. 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[1] to approximately estimate the relevance of attributes. Although the heuristic methods work reasonably well [Quinlan, 1993; Liu and Wen, 1993], it is certain that they miss out the attributes with high order correlations, for example, the parity problem. Hence, on one hand, it is a

[1]First order information contains only one attribute, second order information two attributes, etc.

problem of exponential explosion; on the other hand, it is likely that some relevant attributes will be omitted if the heuristic approach is taken. Our goal becomes clear, i.e., to have a reasonably fast algorithm that can find M relevant attributes with high probability.

Based on the study in line with [Cheeseman *et al.*, 1991; Selman, 1995], this work proposes a probabilistic approach, in particular, a Las Vegas algorithm, that makes probabilistic choices to help guide the search more quickly to find a correct set (or sets) of M attributes. Las Vegas algorithms (LV's) use randomness to guide their search in such a way that a correct solution is guaranteed even if unfortunate choices are made: it will only take longer if this happens [Brassard and Bratley, 1996]. Better still, the proposed algorithm will not keep a user wait forever, it provides the current best solution(s) while the probabilistic searching for the best set of M attributes continues. This paper will start with a review of the previous approaches in Section 2. Section 3 explains our design of the LV algorithm for feature selection, a correctness criterion, and some theoretic analysis of why a correct solution should be expected. Section 4 describes the experimental study with artificial, real-world datasets and an on-site application involving huge datasets reported independently by a local institution[2]. Section 5 discusses various aspects of the algorithm and further work.

2 Previous Approaches and Problems

The problem of feature selection has long been an active research topic within statistics and pattern recognition [Narendra and Fukunaga, 1977; Wyse *et al.*, 1980; Devijver and Kittler, 1982], but most work in this area has dealt with linear regression [Langley, 1994] and is under assumptions that do not apply to most learning algorithms [John *et al.*, 1994]. Researchers 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 in cleaning data.

All the feature selection methods (refer to the related work sections in [Kira and Rendell, 1992; Langley, 1994; John *et al.*, 1994]) can be grouped into two categories: exhaustive or heuristic search of an optimal set of M attributes. For example, Almuallim and Dietterich's FOCUS algorithm [Almuallim and Dietterich, 1994] starts with an empty feature set and carries out exhaustive search until it finds a minimal combination of features that are sufficient to construct a hypothesis consistent with a given set of examples. It works on binary, noise-free data. As pointed out earlier, its time complexity is $O(N^M)$. They proposed three heuristic algorithms to speed up the searching [Almuallim and Dietterich, 1994].

There are many heuristic feature selection algorithms. The Relief algorithm [Kira and Rendell, 1992] 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 [Kira and Rendell, 1992], Relief assumes two-class 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 [Modrzejewski, 1993] 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 attribute, Liu and Wen suggest [1993] to use high order information gains to select features. Since the last two algorithms do not try to explore all the combinations of features, it is certain that they fail on the parity problems (parity-5, parity-10, etc.) where the combinations of a small number of attributes do not help in finding the relevant attributes. Chi2 [Liu and Setiono, 1995] is another heuristic feature selector. It automatically discretizes the continuous attributes and removes irrelevant continuous attributes based on the chi-square statistics and the inconsistency found in the data. If an attribute's values are discretized into one interval, the attribute can be removed since it does not help differentiate different patterns. With the nominal and continuous attributes being mixed, Chi2 considers the latter only since nominal attributes cannot be further discretized.

Another common understanding is that some learning algorithms have built-in feature selection, for example, ID3 [Quinlan, 1986], FRINGE [Pagallo and Haussler, 1990] and C4.5 [Quinlan, 1993]. The results in [Almuallim and Dietterich, 1994] suggest that one should not rely on ID3 or FRINGE to filter out irrelevant features. Since C4.5 conducts test on each individual attribute as well, it is not proper either to use C4.5 to find the minimum set of attributes. It is expected (to be shown in Section 4) that it will fail badly on the parity problems.

[2] Japan - Singapore AI Center, Singapore

[3] The monotonicity assumption is not valid for many induction algorithms used in machine learning. See for dataset 1 (CorrAL) in Section 4 which is reproduced from [John *et al.*, 1994].

To sum up, the exhaustive search approach is infeasible in practice; and the heuristic search approach can reduce the search time significantly, but will fail on hard problems (e.g., the parity problem) or cannot remove redundant attributes. It is right time for a third approach that is fast in producing solutions and selects the optimal and/or near-optimal set(s) of relevant features.

3 A Probabilistic Approach - LVF

The proposed probabilistic approach is a Las Vegas Algorithm [Brassard and Bratley, 1996]. 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 correlations. 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.

3.1 Algorithm and inconsistency criterion

The LVF [4] algorithm below generates a random subset, S, from N features in every round. If the number of features (C) of S is less than the current best, i.e., $C < C_{best}$, the data D with the features prescribed in S is checked against the inconsistency criterion (to be explained later). If its inconsistency rate is below a pre-specified one (γ), C_{best} and S_{best} are replaced by C and S respectively; the new current best (S) is printed. If $C = C_{best}$ and the inconsistency criterion is satisfied, then an equally good current best is found and printed. MAX_TRIES is set to $77 \times N$ [5] in our experimental study following the rule-of-thumb that the more attributes a dataset has (in other words, the larger N is), the harder the problem of feature selection (parity-5 is more difficult than parity-2, e.g.), and hence more tries are needed. When LVF loops MAX_TRIES times, it stops. In our experiments (Section 4.2), the best S obtained last is chosen for further tests using a learning algorithm. When there is a tie, one is chosen at random.

LVF algorithm
    Input: MAX-TRIES,
        D - dataset,
        N - number of attributes;
        γ - allowable inconsistency rate,
    Output: sets of M features satisfying
        the inconsistency criterion
    $C_{best} = N$;
    for i=1 to MAX-TRIES
      S = randomSet(seed);
      C = numOfFeatures(S);
      if ($C < C_{best}$)
        if (InconCheck(S, D) $< \gamma$);
          $S_{best} = S$; $C_{best} = C$;
          print_Current_Best(S)
      else if (($C = C_{best}$) and
          (InconCheck(S, D) $< \gamma$))
        print_Current_Best(S)
    end for

The *inconsistency* criterion (InconCheck(S, D) $< \gamma$) is the key to the success of LVF. The criterion specifies to what extent the dimensionally reduced data can be accepted. The inconsistency rate of the data described by the selected features is checked against a pre-specified rate (γ). If it is smaller than γ, it means the dimensionally reduced data is acceptable. The default value of γ is 0 unless specified. The inconsistency rate of a dataset is calculated as follows: (1) two instances are considered inconsistent if they match except for their class labels; (2) for all the matching instances (without considering their class labels), the inconsistency count is the number of the instances minus the largest number of instances of class labels: for example, there are n matching instances, among them, c_1 instances belong to label_1, c_2 to label_2, and c_3 to label_3 where $c_1 + c_2 + c_3 = n$. If c_3 is the largest among the three, the inconsistency count is $(n - c_3)$; (3) the inconsistency rate is the sum of all the inconsistency counts divided by the total number of instances.

3.2 Theoretic analysis

Here, we show that LVF will select the optimal set(s) of M features. Also, the larger number of optima is, the more likely LVF will find M features, in presence of redundant attributes, according to the inconsistency criterion.

With a good pseudo random number generator [Press *et al.*, 1992], the selection of an optimal subset of

[4] F stands for a filter version of Las Vegas algorithms.

[5] 77 is chosen by experiment. MAX_TRIES can be defined according to applications in hand or based on the experience from experimentation. Too small or too big a MAX_TRIES will affect the performance of LVF. The compromise is made between good and fast solutions. The longer LVF runs, the better its results are. Refer to the analysis in Section 3.2.

M features can be considered non-replacement experiments. The probability of finding the optimal subset at the $(k+1)$th experiment is $\frac{1}{2^N-k}$, and the probability of having to conduct $(k+1)$ experiments before finding the optimal subset 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. When N is large, MAX_TRIES $\ll 2^N$. Here we assume there is only one optimum.

In presence of redundant attributes (it is quite common in the real-world data), using the inconsistency criterion, this means that the number of optima (l) is larger than 1. Therefore, since at the $(k+1)$th tossing, the probability of finding one optimum is $\frac{l}{2^N-k}$, it is more likely for LVF to find an optimal feature set. In other words, redundant attributes help find an optimal solution faster.

4 Experimental Results

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 [John *et al.*, 1994], Monks1-3 [Thrun *et al.*, 1991], and Parity5+5. The other type is real-world data including Credit, Vote, Labor, and Mushroom [Quinlan, 1993; Murphy and Aha, 1994]. The choice of these datasets simplifies the comparison of this work with some published work. These datasets except Mushroom were used in [John *et al.*, 1994] in which comparisons with different methods were described. Nevertheless, the experiments here can alone demonstrate the effectiveness of LVF owing to the analysis given in the previous sections (1, 2 and 3).

Artificial Data:

1. **CorrAL** The data was designed in [John *et al.*, 1994]. 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 [Thrun *et al.*, 1991]. 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 [Quinlan, 1993] 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.
7. **Mushroom** The dataset has a total of 8124 instances, of which 1000 instances are randomly selected for testing, the rest are used for training. The data has 22 discrete attributes. Each attribute can have 2 to 10 values.

4.1 Results

For the artificial datasets, the evaluation of LVF is simple since the relevant attributes are known. However, for the real-world datasets, it is not clear what the relevant features are. Therefore, whether the selected features are relevant or not can be only determined indirectly. One way is to see the effect of feature selection through a learning algorithm. These results are reported in Section 4.2. LVF is run 100 times on each training dataset. The numbers of selected features and frequency are reported in Table 1 under the condition that the inconsistency criterion be satisfied. Also reported is a sample of these selected features for each dataset which can be directly used by readers.

For the artificial datasets, the relevant attributes are always selected, albeit a few of irrelevant ones are also chosen sometimes. For the problem as hard as Parity5+5, LVF correctly identifies the correct attributes all the time, plus one irrelevant attribute sometimes. For the real-world datasets, the number of attributes is reduced at least by half to less than one fifth of the original. In the next section, how effective these

Table 1: Results of 100 runs of LVF on the datasets with one example of the minimum set of features for each dataset.

Dataset	# Att	# Selected Att (Frequency)	Features (1 minimum set)
CorrAL	6	4 (86), 5 (14)	A0, A1, B0, B1
Monk1	6	3 (100)	A1, A2, A5
Monk2	6	6 (100)	A1 - A6
Monk3[6]	6	3 (100)	A2, A4, A5
Parity5+5	10	5 (71), 6 (29)	A1 - A5
Vote	16	8 (7), 9 (53), 10 (30), 11 (10)	A1 - A4, A9, A11, A13, A16
Credit (CRX)	15	5 (100)	A2, A3, A4, A9, A14
Labor	16	3 (44), 4 (56)	A2, A10, A11
Mushroom	22	4 (57), 5 (43)	A4, A5, A12, A22

[6] Allowing 5% inconsistency. If not, four attributes are selected: the above chosen 3 plus A1.

Table 2: 10-fold cross validation results on Tree Size and Error Rates of ID3 and C4.5 before and after applying LVF to the datasets. Bef stands for Before; Aft for After; P-val for P value of t-test; and "-" means that all values for the two groups in comparison are equal.

	ID3						C4.5					
	TreeSize			ErrorRate(%)			TreeSize			ErrorRate (%)		
Dataset	Bef	Aft	P-val	Bef	Aft	P-val	Bef	Aft	P-val	Bef	Aft	P-val
CorrAL	10.6	11.4	.2277	37.5	20.8	.1205	10.6	11.4	.2277	37.5	20.8	.1205
Monk1	103.7	41.0	.0001	4.5	0.0	.0217	43.0	41.0	.2105	.7	0.0	.1039
Monk2	217.4	217.4	-	37.0	37.0	-	16.3	16.3	-	21.1	21.1	-
Monk3	20.6	19.0	.1679	1.4	1.1	.5109	19.0	19.0	-	1.1	1.1	1
Parity5+5	79.0	62.8	.0001	48.5	1.5	.0001	47.0	62.8	.0015	45.5	1.5	.0001
Vote	29.8	27.1	.2878	5.3	5.3	.9934	14.5	6.1	.0001	5.3	5.5	.8357
Credit	139.8	144.8	.5033	16.8	19.7	.1001	54.0	38.0	.0164	15.5	15.2	.8490
Labor	15.2	8.4	.0001	15.3	14.0	.8218	7.3	7.0	.3434	17.3	14.3	.6631
Mushroom	29	29	1	0.0	0.0	-	29	29	1	0.0	0.0	-

selected features are will be determined by cross validations of a learning algorithm.

4.2 Further verification

Table 1 shows that those features in the last column are necessary in order to satisfy the inconsistency criterion (the inconsistency rate is 0 except for Monk3). As mentioned above, it is clear for the artificial datasets whether the relevant features are chosen or not, but for the real-world datasets, indirect evaluation is necessary by checking a learning algorithm's performance before and after feature selection. C4.5 [Quinlan, 1993] is chosen here because (1) it works well on most data sets as reported by many researchers; and (2) it employs a heuristic to find simplest tree structures [Quinlan, 1986; Quinlan and Rivest, 1989; Quinlan, 1995]. 10-fold cross validation is applied and the default settings of C4.5 are used in the experiment. For the experiments of "after feature selection", only the features shown the last column of Table 1 are used. Given in Table 2 are the average accuracy rates of C4.5 before and after applying feature selection to the datasets. ID3 results are also given by taking unpruned trees of C4.5.

Results in Table 2 suggest that the performance of both ID3 and C4.5 has improved in general. That is, the tree size is getting smaller and the accuracy higher. For the artificial datasets, this experiment further shows that with the relevant attributes, ID3 and C4.5 are doing better than with the full set of attributes. For the real-world datasets, ID3 and C4.5 are also doing better with the selected attributes. This indicates that LVF has selected relevant attributes for these datasets. In particular, ID3 and C4.5 did badly on Parity5+5 before feature selection. Nevertheless, ID3 and C4.5 did as well on Mushroom with 22 attributes as with 4 attributes. This demonstrates that ID3 and C4.5 do select relevant features for some datasets, though not for all. Given also in Table 2 are the results of t-test. The lower a P-value is, the more confident we are in rejecting the NULL hypothesis that the two averages are the same.

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

The experimental results from [John *et al.*, 1994] 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 [Kira and Rendell, 1992], because of significant variance in the relevance rankings given by Relief [John *et al.*, 1994].

4.3 Applying LVF to huge datasets

Feature selection is particularly useful when datasets are huge since many learning algorithms may encounter difficulties. Feature selection can help reduce the dimensionality of the datasets so that more learning algorithms can be chosen to induce rules. Hence, the huge datasets are also an ultimate test for a feature selection algorithm. It is obvious that the wrapper approach is not suitable in this case since some favorite learning algorithms may have problems with handling the huge datasets. LVF had an opportunity to undergo a real test of huge datasets.

The datasets involved are related to the service industry. LVF was given to a local institution, who was in need of a method to reduce the number of attributes before applying some machine learning algorithms to the datasets due to the huge size of the datasets. The datasets are confidential. The limited information we have is the size of datasets and the number of attributes. One dataset (let us call it HD1) has 65,000 instances and 59 attributes; the other (HD2) has 5,909 instances and 81 attributes. Both datasets are discrete, attribute values range from 2 to 13. According to the report from the testing site (the institution ran LVF independently and without modification), LVF found that 10 and 35 attributes were needed for describing HD1 and HD2 respectively without sacrificing their discriminating power, after several hours of running LVF on Sun Sparc. They did another experiment in which only 10,000 instances of HD1 were used, it took LVF about 5 minutes to complete its run and obtained the same results. LVF has *significantly* reduced the number of attributes. At present, the institution applies it to more datasets for the preprocessing purpose in search of simple rules.

5 Discussion and Future Work

The special advantages of this method are (1) simple to implement; (2) fast to obtain results; and (3) not affected by any bias of a learning algorithm. This feature selector prints out a possible solution whenever it is found; afterwards it reports either a better one or equally good ones. This is a really nice feature because while it works hard to find the optimal solution, it provides those near optimal solutions. The longer it runs, the better solutions we get. It stops when a specified MAX-TRIES is exceeded. In this way, it avoids the exponential computation problem. If the resources permit, LVF can be run on as many machines as we wish by having different seeds in LVF. The time to find the optimal solution can be reduced and the chance to find it becomes greater.

5.1 Filter and/or wrapper

Although it is natural to use this feature selector as a filter, it is straightforward to use it in a wrapper approach by replacing inconsistency checking with accuracy comparisons by a learning algorithm on different sets of features. However, since inconsistency checking is $O(n)$ [7] and a fast learning algorithm requires at least $O(n \log n)$, where n is the number of patterns in the training data, it is suggested to use inconsistency as a criterion to select features. In addition, for the filter approach, only for those better selections with fewer number of attributes, the inconsistency criterion will be tested. It is noticed in the experiments that the number of features selected is quickly reduced approximately by half, since the number of relevant features is usually small (refer to Table 1). Comparing to the wrapper approach, because the criterion is the predictive accuracy, for every newly generated set of features, the criterion must be tested. In other words, the number of criterion testing is significantly larger. Experiments [Liu and Setiono, 1996] have confirmed that for the above datasets, the wrapper model normally spends a few hours[8] while the filter model usually takes several minutes on a dedicated SUN SPARC20 for each experiment of LVF on the publically available datasets.

Normally, our feature selector will report several sets of M features for a given problem. In other words, in terms of inconsistency rate, they are equally good. Choosing which set should depend on the particular learning algorithm employed for application. One set can be chosen if the learning algorithm gives the highest accuracy on this set among others. This is a *proper*, recommended use of the wrapper approach for feature selection.

5.2 Inconsistency criterion

There may be a problem with using inconsistency as a feature selection criterion when one attribute alone (such as social security number) can guarantee that there is no inconsistency in the data. Obviously, this attribute is irrelevant for rule induction. The problem can be solved by leaving this attribute out of the feature selection process. If there is no prior knowledge, it will just take one run of LVF to locate this kind of attributes (Recall that one run of LVF has MAX_TRIES loops). Another run of LVF on the other attributes will recognize the correct set of features.

LV works on discrete attributes only since it relies on the inconsistency calculation. One way is to apply a discretization algorithm (e.g., Chi2) to deiscreze the continuous attributes first before one runs LVF. Other possibilities are (1) simply treat a continuous attribute as a discrete one in some cases; (2) apply LVF only to the discrete attributes when the number of attributes is large. Another concern about the inconsistency criterion is that the reduced data may not be ideal for a particular learning algorithm. Although this phenomenon has not been observed in the experiments, more empirical study is needed for confirmation. A filter/wrapper combined model mentione above may help in this regard.

5.3 Speed of LVF and value of M

In all the experiments, it is shown that M - the number of relevant features is usually small. Since only those sets whose number of features is smaller than or equal to the current best will be checked for inconsistency, when M becomes smaller and smaller, many randomly generated sets of features need not be tested at all. In any case, the inconsistency checking is $O(N)$ in our implementation. As reported by the users of on-site application, it took LVF about 5 minutes to run on a dataset of 10,000 instances and 59 attributes. It is natural to expect that the more patterns and attributes, the more time LVF requires; but the time required increases nearly linearly due to the reasons above.

5.4 Handling noisy data and multiple class values

Noisy data can be easily handled by initializing the minimum allowable inconsistency rate (i.e., γ) to a certain value. In all the experiments reported here, γ is obtained by calculating the inconsistency among the original training data. It is 0 except for Monk3. If the noise level is reflected in the inconsistency among the data, LVF automatically takes care of it. However, if there is no link between the noise level and the inconsistency rate, prior knowledge about the noise level is needed. For the instance of the Monk3 problem, the training data contains 5% noise (6 items are contaminated) and no inconsistency. If $\gamma = 0$, LVF found 4 attributes although only three are relevant. When the noise level was known a priori ($\gamma = 5\%$), LVF then found the three relevant attributes.

As pointed out in Section 2, several popular feature selectors work only on binary class values. LVF does not impose such a constraint. This allows LVF to be applied to various applications.

[7] Precisely, it is close to $O(n)$ since this is achieved by implementing hashing in inconsistency checking. Another method of $O(n)$ can be found in [Almuallim and Dietterich, 1994].

[8] Using 10-fold cross validations to obtain accuracy is another factor that increases the time.

5.5 Scaling-up

Section 4.3 shows that LVF can scale up. However, improvement is still possible. When datasets are huge, the running time of LVF is no doubt longer. In order to speed up the preprocessing, one way of improving the present one-go approach is to go for incremental sampling. The idea is as follows: when the dataset contains a large enough number of instances (say 20,000), take 10% of it (called the training data) to run LVF and check the inconsistency criterion based on its selected features on the remaining 90% of the data, add those patterns causing inconsistencies to the training data, rerun LVF, then check again; continue the process until the number of inconsistencies is below a tolerable value (γ).

5.6 Using LVF as a reference

With some knowledge about the data, a heuristic feature selection method can be designed. The advantages of doing so are (1) the heuristic method can be deterministic; and (2) since it is specially designed, it can be made faster. However, it is not a simple task to fully test the effectiveness of a heuristic method. The LVF method can help in verification. Since it has been established that LVF gives all the best solutions with high probability (= 1 if given a reasonably long time), the heuristic solutions can be compared with the solutions by LVF. If there is no matching solution or if the difference is big, it can be said safely (or it is true with high probability) that the heuristic method is not properly designed. In general, if determinism is not required, LVF can be always used as a first choice.

6 Conclusion

A probabilistic approach to feature selection is proposed in contrast to the two common approaches (exhaustive and heuristic). Theoretic analysis and empirical study show that the proposed approach is simple to implement, fast to get results, and guaranteed to find the optimal if resources permit. *It can scale up, too.* All these have been achieved due to (1) the probabilistic approach; (2) the inconsistency criterion (which makes a filter solution possible); and (3) the linear time cost of inconsistency checking, among others. It has been successfully used by a local institution in preprocessing huge datasets. LVF is available for trial.

Also presented here is an extensive discussion on the choice of the filter model vs. the wrapper model, on why LVF is fast to obtain results, on practical issues such as noise handling, multiple class values and scaling-up, and on when and how to use of this approach (e.g., choosing between a heuristic method and LVF).

Acknowledgments:

Thanks H.Y. Lee for the suggestions on an earlier version of this paper; H.L. Ong and A. Pang for providing the results on their applying LVF to huge datasets at Japan - Singapore AI Center; and Y.C. Chew for transforming the graphs in [John *et al.*, 1994] into tables. Thanks also go to the two anonymous referees for their valuable comments.

References

[Almuallim and Dietterich, 1994] 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.

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

[Cheeseman *et al.*, 1991] P. Cheeseman, B. Kanefsky, and W.M. Taylor. Where the really hard problems are. In *Proceedings of IJCAI91*, pages 331–337. Morgan Kaufman, 1991.

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

[John *et al.*, 1994] 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.

[Kira and Rendell, 1992] 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.

[Langley, 1994] P. Langley. Selection of relevant features in machine learning. In *Proceedings of the AAAI Fall Symposium on Relevance.* AAAI Press, 1994.

[Liu and Setiono, 1995] 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. **http://www.iscs.nus.sg/~liuh/tai95.ps**

[Liu and Setiono, 1996] H. Liu and R. Setiono. Feature selection and classification - a probabilistic wrapper approach. In *Proceedings of the 9th International Conference on Industrial and Engineering Applications of AI and ES*, 1996. **http://www.iscs.nus.sg/~liuh/wrapper.ps**

[Liu and Wen, 1993] 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.

[Modrzejewski, 1993] 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.

[Murphy and Aha, 1994] 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.

[Narendra and Fukunaga, 1977] P.M. Narendra and K. Fukunaga. A branch and bound algorithm for feature subset selection. *IEEE Trans. on Computer*, C-26(9):917–922, September 1977.

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

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

[Quinlan and Rivest, 1989] J.R. Quinlan and R.L. Rivest. Inferring decision trees using the minimum description length princple. *Information and Computation*, 80(3), 1989.

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

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

[Quinlan, 1995] J.R. Quinlan. Mdl and categorical theories (continued). In *Proceedings of International Conference on Machine Learning*, 1995.

[Selman, 1995] B. Selman. Stochastic search and phase transitions: AI meets physics. In C.S. Mellish, editor, *Proceedings of IJCAI95*, volume 1, pages 998–1002, 1995.

[Thrun *et al.*, 1991] 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.

[Wyse *et al.*, 1980] 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.