A Monotonic Measure for Optimal Feature Selection

Huan Liu[1] and Hiroshi Motoda[2] and Manoranjan Dash[3]

[1] Dept of Info Sys & Comp Sci, National University of Singapore, Singapore 119260.
[2] Division of Intelligent Sys Sci, Osaka University, Ibaraki, Osaka 567, Japan.
[3] BioInformatics Centre, National University of Singapore, Singapore 119074.
{liuh, manoranj}@iscs.nus.edu.sg motoda@sanken.osaka-u.ac.jp

Abstract. Feature selection is a problem of choosing a subset of relevant features. In general, only exhaustive search can bring about the optimal subset. With a monotonic measure, exhaustive search can be avoided without sacrificing optimality. Unfortunately, most error- or distance-based measures are not monotonic. A new measure is employed in this work that is monotonic and fast to compute. The search for relevant features according to this measure is guaranteed to be complete but not exhaustive. Experiments are conducted for verification.

1 Introduction

The basic problem of classification is to classify a given pattern (example) to one of m known classes. A pattern of features presumably contains enough information to distinguish among the classes. When a classification problem is defined by features, the number of features (N) can be quite large. A classifier may encounter problems to learn something meaningful because the required amounts of data ($\mathcal{N}$, or the number of patterns) increase exponentially in proportion with N. The task of feature selection is to determine which features to select in order to achieve maximum performance with the minimum measurement effort [2]. Reducing features directly alleviates the measurement effort. Performance of a classifier can be its predictive accuracy, i.e., *1 - error rate*.

As was mentioned in [2], if the goal is to minimize the error rate, and the measurement cost for all the features is equal, then the most appealing function to evaluate the potency of a feature to differentiate between the classes is the Bayes Classifier. Due to the inductive nature of classification problems, no full distribution of data can be obtained. Extensive research effort was devoted to the investigation of other functions (mostly based on distance and information measures, or simply on classifiers) for feature evaluation. If there exist N features, to find an optimal subset of features without knowing how many features are relevant, it requires to explore all the 2^N subsets. When N is large, this exhaustive approach is out of the question. Therefore, various feature selection methods have been designed to avoid exhaustive search while still aiming at the optimal subset. Examples are Branch & Bound [7], Focus [1], Relief [4], Wrapper methods [3], and LVF [5].

The feature selection problem can be viewed as a search problem [9]. The search process starts with either an empty set or a full set. For the former, it expands the search space by adding one feature at a time (Sequential Forward Selection) [1]; for the latter, it expands the search space by deleting one feature at a time (Sequential Backward Selection) [7]. As we shall see, a good alternative to exhaustive search is Branch & Bound like algorithms if there exists a monotonic function of evaluating features. Assuming we have subsets $\{S_0, S_1, ..., S_n\}$, we have a measure U that evaluates each subset S_i. The monotonicity condition requires that:

$$S_0 \supset S_1 \supset ... \supset S_n \Rightarrow U(S_0) \leq U(S_1) \leq ... \leq U(S_n).$$

In this case, the search can be complete but not exhaustive. In other words, the optimal subset is guaranteed. Many distance and information based measures have been shown to be non-monotonic [9]. Many researchers pointed out that the only remaining alternative is to use the error rate of a classifier as the measure. Among many classifiers, however, only the Bayes Classifier satisfies this monotonicity condition [1] because other classifiers adopt some assumptions and employ certain heuristics [9, 2, 3]. Another disadvantage of using the error rate as a measure in the wrapper models of feature selection is it is slow to compute. For example, to construct a decision tree, it would take at least $O(\mathcal{N} \log \mathcal{N})$. We present here a measure that is monotonic as well as fast to compute ($O(\mathcal{N})$) in search of optimal subsets.

2 A Non-exhaustive yet Complete Search Algorithm

For two subsets of features, S_i and S_j, one is preferred to the other based on a measure U of feature-set evaluation. S_i and S_j are indifferent if $U(S_i) = U(S_j)$ and $\#(S_i) = \#(S_j)$ where $\#$ is the cardinality; S_i is preferred to S_j if $U(S_i) = U(S_j)$ but $\#(S_i) < \#(S_j)$, or if $U(S_i) < U(S_j)$ and $\#(S_i) \leq \#(S_j)$. As we know, the condition for Branch & Bound to work optimally is that U is monotonic.

In this work, U is an *inconsistency rate* over the data set given S_i. The inconsistency rate is calculated as follows: (1) two patterns are considered inconsistent if they match all but their class labels, for example, patterns (0 1 1) and (0 1 0) match with respective to the first two attributes, but are different in the last attribute (class label); (2) the inconsistency count is the number of all the matching patterns minus the largest number of patterns of different class labels: for example, there are n matching patterns, among them, c_1 patterns 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 $(n - c_3)$; and (3) the inconsistency rate is the sum of all the inconsistency counts divided by the total number of patterns ($\mathcal{N}$). By employing a hashing mechanism, we can compute the inconsistency rate approximately with a time complexity of $O(\mathcal{N})$.

[1] But it requires the full distribution of the data.

A proof outline is given to show that this inconsistency rate measure is monotonic, i.e., if $S_i \subset S_j$, then $U(S_i) \geq U(S_j)$. Since $S_i \subset S_j$, the discriminating power of S_i can be no greater than that of S_j. It's known that the discriminating power is reversely proportional to the inconsistency rate. Hence, the inconsistency rate of S_i is greater than or equal to that of S_j, or $U(S_i) \geq U(S_j)$. The monotonicity of the measure can also be proved as follows. Consider three simplest cases of $S_k (= S_j - S_i)$ without loss of generality: (i) features in S_k are irrelevant, (ii) features in S_k redundant, and (iii) features in S_k relevant. If features in S_k are irrelevant, based on the definition of irrelevancy, these extra features do not change the inconsistency rate of S_j since S_j is $S_i \cup S_k$, so $U(S_j) = U(S_i)$. Likewise for case (ii) based on the definition of redundancy. If features in S_k are relevant, that means S_i does not have as many relevant features as S_j. Obviously, $U(S_i) \geq U(S_j)$ in the case of $S_i \subset S_j$. It is clear that the above results remain true for cases that S_k contains irrelevant, redundant as well as relevant features.

ABB is a Branch & Bound algorithm with its bound set to the inconsistency rate δ of the data set with the full set of features. It starts with the full set of features S^0, removes one feature from S_j^{l-1} in turn to generate subsets S_j^l where l is the current level and j specifies different subsets at the lth level. If $U(S_j^l) > U(S_j^{l-1})$, S_j^l stops growing (the branch is pruned), otherwise, it grows to level $l+1$, in other words, one more feature will be removed. In short, ABB seeks the smallest S_j whose inconsistency rate is δ. S is the full feature set and D the data set.

```
δ = inConCal(S, D);
ABB (S, D) {
    /* subset generation */
    For all feature f in S {
        S_1 = S − f; /* remove one feature at a time */
        enQueue(Q, S_1);} /* add at the end */
    while notEmpty(Q) {
        S_2 = deQueue(Q); /* remove at the start */
        if (S_2 is legitimate ∧ inConCal(S_2, D) ≤ δ)
            /* recursion */
            ABB (S_2, D); }}
```

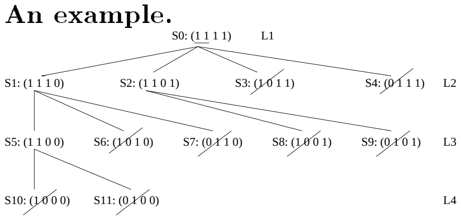
Function inConCal() calculates the consistency rate of data given a feature subset. Care has to be taken in implementing the algorithm such that (1) no duplicate subset will be generated via proper enumeration; and (2) no child node of a pruned node will be generated by ensuring that the Hamming distance between a new subset at the current level and any pruned subset at the parent level is greater than 1[2] (this is the legitimacy test in ABB).

It is not required anymore to specify the size of a desired subset, M, or a bound for the measure as normally required by Branch & Bound. Thus, its name

[2] A full set of N attributes entails an N-bit binary array in which ith value 1 means ith attribute is chosen to include in the subset.

ABB. At the end of search, we just need to report the legitimate subsets with the smallest cardinality as the optimal subsets.

An example.



Refer to the figure: there are four features, assuming only the first two are relevant. The root $S_0 = (1\ 1\ 1\ 1)$ of the search tree is a binary array with four '1's. Following ABB, we expand the root to four child nodes by turning one of the four '1's into '0' (L2). All four are legitimate: $S_1 = (1\ 1\ 1\ 0)$, $S_2 = (1\ 1\ 0\ 1)$, $S_3 = (1\ 0\ 1\ 1)$, and $S_4 = (0\ 1\ 1\ 1)$. Since one of the relevant features is missing, $U(S_3)$ and $U(S_4)$ will be greater than $U(S_0)$ where U is the inconsistency rate on the given data. Hence, the branches rooted by S_3 and S_4 are pruned and will not grow further. Only when a new node passes the legitimacy test will its inconsistency rate be calculated. Doing so improves the efficiency of ABB because $\mathcal{N}$ (number of patterns) is normally much larger than N (number of attributes). The rest of the nodes are generated and tested in the same spirit.

3 Empirical Study

The objectives of this empirical study are to verify: 1. ABB indeed finds optimal subsets for various data sets, and 2. features selected are good for various learning algorithms. We select two groups of data sets: one with known relevant features and the other with unknown relevant features as shown in Table 1. All data sets are from [6] except for Corral [3]. For the first group of 5 data sets we compare the subsets selected by ABB with the known. For the second group we compare the outputs of ABB with that of Focus, a popular method in literature that guarantees optimal subsets. For the second objective we choose two different learning algorithms: a decision tree method (C4.5 [8]) and a standard back-propagation neural network (SNNS [10]). Two thirds of the data is used for selecting features by ABB and Focus. The other one third is the testing data for SNNS. We run 10-fold cross validation with C4.5 on the whole data. Results showed that ABB indeed finds optimal subsets as validated by Focus and *a priori* knowledge. Focus does breadth first search starting from the empty set and stops after reaching the first consistent subset. In fact, the subset found by Focus can be just one of the solutions of ABB.

While running ABB and Focus, we found an interesting fact that *"ABB and Focus complement each other with respect to time taken to reach optimal subset"*. To verify this, we collected the number of subsets evaluated by ABB and Focus in Table 1. ABB and Focus adopt different search directions. So, if the size of the optimal subset is not small, choose ABB, otherwise, choose Focus. To take advantage of both algorithms one may run both simultaneously till any one of the two algorithms stops.

Data set	D_{Tr}	D_{To}	C	N	M	ALL #	ABB #	Focus #
CorrAL	32	64	2	6	4	64	14	42
Monk1	124	432	2	6	3	64	12	24
Monk2	169	432	2	6	6	64	7	63
Monk3	122	432	2	6	3	64	19	35
Par3+	341	512	2	9	3	512	265	46
WBC	463	699	2	9	4	2^9	188	145
LED-7	400	600	10	7	5	2^7	9	99
Letter	5980	8968	26	16	9	2^{16}	1971	42,634
LYM	100	148	4	18	6	2^{18}	82,156	23,167
Vote	300	435	2	16	8	^{16}	301	39,967
KrVsKp	2131	3196	2	36	29	2^{36}	4367	$> 2^{28}$

Table 1. D_{Tr} - training set, D_{To} - total set, C - no. of classes, N - no. of original features; M - no. of selected features, All # - no. of all possible subsets, ABB # - no. of subsets evaluated by ABB, Focus # - no. of by Focus.

Based on the subsets found for each data set, we obtain the results shown in Table 2. In general, C4.5 (10-fold cross validation) gave better or equally good accuracy after feature selection. But the results for tree size are interesting, some showing larger tree sizes after feature selection as pointed out by $\leftarrow$ in Table 2 Researchers noticed that smallest trees do not necessarily give the best predictive accuracy. What is observed here is that better accuracy may not mean a smaller tree size. We also noticed that "after" feature selection, in most cases, C4.5 used all features selected by ABB, which indicates that features selected by ABB are relevant in decision tree induction. However, C4.5 did choose features not selected by ABB in the "before" setting, e.g., in the case of CorrAL data.

To run the neural network classifier, we fixed the learning rate as 0.1, the momentum as 0.5, one hidden layer, the number of hidden units as half of the original input units for all data sets. We found a proper number of CYCLES for each data set by observing a sustained trend of no decrease of error (MSE) in a trial run. Later, with these parameters, two runs of SNNS were made on data sets with and without feature selection via ABB respectively. This experiment is very simplistic and designed to get some rough idea about the effect of selected features to a neural network classifier. In most cases, their error rates drop. Error rates for Letter are dubiously high. Due to the complication of parameter setting, more sophisticated experiments are being planned.

4 Conclusion

We demonstrated that the inconsistency rate is a monotonic measure and it is fast to compute. With such a measure, Branch & Bound is a good deterministic algorithm (the search is not exhaustive, yet complete), The new method ABB is simple to implement and guarantees optimal subsets of features. The empirical

Data set	C4.5					NN			
	Tree Size		Error Rate %			CYCLES	#HU	Error Rate %	
	Before	After	Before	After				Before	After
CorrAL	14.6	13.0	6.0	0.0		1000	3	4.55	**9.09**
Monk1	43.0	41.0	0.7	0.0		1000	3	50.68	37.84
Monk2	16.3	16.3	21.1	21.1		1000	3	29.73	29.73
Monk3	19.0	19.0	1.1	1.1		1000	3	12.16	0.0
Par3+3+3	13.0	**15.0**	17.2	0.0	←	1000	5	59.09	9.09
WBC	38.0	36.0	6.6	6.0		1000	5	8.05	6.78
LED-7	19.0	19.0	0.0	0.0		1000	4	0.0	0.0
Letter	6660.0	6113.0	28.1	27.9		15000	8	75.5	61.42
LYM	26.9	**29.6**	21.8	21.0	←	7000	9	25.0	**29.17**
Vote	16.0	**19.0**	2.8	2.3	←	4000	8	6.67	4.0
KrVsKp	54.8	**64.8**	0.52	**0.83**	←	8000	18	2.07	1.50

Table 2. Results of C4.5 (10-fold cross validation) and Back-propagation neural network. #HU - number of hidden units.

study suggests that (1) ABB removes irrelevant, redundant, and/or correlated features even with the presence of noise (as in Monk3 with 5% noise); and (2) the performance of a classifier with the features selected by ABB also improves. Another finding from this study is Focus and ABB complement each other due to their opposite search directions.

References

1. H. Almuallim and T.G. Dietterich. Learning with many irrelevant features. In *Proceedings of AAAI*, 1991.
2. M. Ben-Bassat. Pattern recognition and reduction of dimensionality. In P. R. Krishnaiah and L. N. Kanal, editors, *Handbook of statistics-II*, pages 773–791. North Holland, 1982.
3. G.H. John, R. Kohavi, and K. Pfleger. Irrelevant feature and the subset selection problem. In *Proceedings of ICML*, pages 121–129. Morgan Kaufmann, 1994.
4. K. Kira and L.A. Rendell. The feature selection problem: Traditional methods and a new algorithm. In *Proceedings of AAAI*, pages 129–134. 1992.
5. H. Liu and R. Setiono. A probabilistic approach to feature selection - a filter solution. In *Proceedings of ICML*, pages 319–327. Morgan Kaufmann, 1996.
6. C.J. Merz and P.M. Murphy. UCI repository of machine learning databases. `http://www.ics.uci.edu/~mlearn/MLRepository.html` . Irvine, CA: University of California, Department of Information and Computer Science, 1996.
7. 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.
8. J.R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993.
9. W. Siedlecki and J Sklansky. On automatic feature selection. *International Journal of Pattern Recognition and Artificial Intelligence*, 2:197–220, 1988.
10. A. Zell and et al. Stuttgart neural network simulator (SNNS), user manual, version 4.1. ftp.informatik.uni-stuttgart.de/pub/SNNS, 1995.