Hybrid Search of Feature Subsets

Manoranjan Dash[1] and Huan Liu[2]

[1] BioInformatics Centre, National University of Singapore, Singapore 119074.
[2] School of Computing, National University of Singapore, Singapore 119260.

Abstract. Feature selection is a search problem for an "optimal" subset of features. The class separability is normally used as one of the basic feature selection criteria. Instead of maximizing the class separability as in the literature, this work adopts a criterion aiming to maintain the discriminating power of the data. After examining the pros and cons of two existing algorithms for feature selection, we propose a hybrid algorithm of probabilistic and complete search that can take advantage of both algorithms. It begins by running LVF (probabilistic search) to reduce the number of features; then it runs "Automatic Branch & Bound (ABB)" (complete search). By imposing a limit on the amount of time this algorithm can run, we obtain an approximation algorithm. The empirical study suggests that dividing the time equally between the two phases yields nearly the best performance, and that the hybrid search algorithm substantially outperforms earlier methods in general.

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. Pattern classification is inherently connected to information reduction. Features can be redundant or irrelevant. An irrelevant feature does not affect the underlying structure of the data in any way. A redundant feature does not provide anything new in describing the underlying structure. Because redundant and irrelevant information is cached inside the totality of the features, a classifier that uses all features will perform worse than a classifier that uses relevant features that maximize interclass differences and minimize intraclass differences [2]. Feature selection is a task of searching for "optimal" subset of features from all available features [4]. Its motivation is three-fold: *Simplifying the classifier; Improving the accuracy of the classifier; and Reducing data dimensionality for the classifier*. The last point is particularly relevant when a classifier is unable to handle large volume of data. The fundamental function of a feature selector is to extract, with appropriate tools, the most useful information from the representation vector (feature values), and present it in the form of a pattern vector of a lower dimensionality whose elements represent only the most significant aspects of the input data [5, 11].

In order to measure the usefulness (or optimality) of selected features, we need selection criteria to measure feature goodness. The class separability is often used as one of the basic selection criteria. When a set of features maximizes the class separability, it is considered to be well suited for classification [17]. From a statistics viewpoint, five different measurements for class separability are analyzed in [5]: error probability, interclass distance, probabilistic distance, probabilistic dependence and entropy. Information-theoretic considerations [20] suggested something similar: using a good feature of discrimination provides compact descriptions of each of the two classes, and these descriptions are maximally distinct. Geometrically, this constraint can be interpreted to mean that (i) such a feature takes on nearly identical values for all examples of the same class, and (ii) it takes on some different values for all examples of the other class. In this work, we use a selection criterion that does not try to maximize the class separability but tries to retain the discriminating power of the data defined by original features. That is, feature selection is formalized as finding the smallest set of features that is "consistent" in describing classes with the full set, where a set S of features is consistent if no two examples with the same values on S have different class labels [1]. Applying this criterion does make feature selection simpler and the class separability of the original data can be retained.

This aspect of feature selection is related to the study of search strategies. Extensive research effort has been devoted to this study [19, 18, 10]. Examples are Branch & Bound [15, 18], Relief [6, 9], Wrapper methods [7], Approximate Markov Blanket [8], and LVF [13]. 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 (Step-wise Forward Selection) - an example is Focus [1]; for the latter, it expands the search space by deleting one feature at a time (Step-wise Backward Selection) - an example is 'Branch & Bound' [15]. If the size of the smallest consistent feature set is even moderately large, Focus will take a long time. Similarly, if there are a moderate number of irrelevant features, 'Branch and Bound' will take a long time. A third, more robust, option is to start with a random feature set and expand the search space by randomly generating new subsets (e.g., LVF [13]). In the rest of the paper P is the number of patterns, N is the number of features, and M is the minimal size of features.

The above three representative methods are designed for different situations. When the minimal size (M) is small, it is sensible to apply Focus since 2^M is not large; when M is large, that is, most features are relevant, it is reasonable to apply 'Branch & Bound' assuming there exists a monotonic measure; and when M is neither large nor small, it is suitable to apply LVF. This work aims at a hybrid algorithm that takes advantage of these three algorithms.

The contributions of this paper are: adopting a selection criterion to be used by all three algorithms; and proposing and implementing a hybrid search algorithm for feature selection which is better than any of the non-hybrid algorithms in terms of time and accuracy of a classifer. In Section 2, we briefly review LVF and ABB [12] and perform experiments and analysis to investigatetheir strengths and weaknesses. In Section 3, we propose to combine LVF and ABB based on the results in Section 2, and consider when such a hybrid algorithm should be switched from LVF to ABB, given a certain amount of time, in order to achieve the best performance. In Section 4, we compare the algorithms on several datasets and evaluate the benefits of the hybrid search. The paper concludes in Section 5 with guidelines on applying FocusM, ABB and the hybrid (QBB).

2 Probabilistic Search and 'Branch & Bound'

We review two algorithms, LVF (probabilistic) [13] and ABB (complete) [12] and explore their advantages and disadvantages. Both methods use an *inconsistency rate* as the feature goodness measure (U). In [12], it is shown that U is a monotonic measure. For detailed accounts, refer to [11]. In all the algorithms S is the original feature set and D is the dataset.

2.1 LVF

This is a Las Vegas algorithm [3] for feature subset selection which makes probabilistic choices of subsets in search of an optimal set. LVF keeps the smallest subset of features randomly generated so far that satisfies a threshold (by default it is the inconsistency rate of the data with all features). It is fast in reducing number of features and can produce optimal solutions if computing resources permit. In the algorithm *inConCal* is a function that calculates the inconsistency of a feature set and returns the inconsistency rate and 'yet another solution' means a solution of the same size as the last found solution.

$\delta = inConCal(S, D)$;
$T = S$;
LVF(S, $MaxTries$, D) {
    for j=1 to $MaxTries$ {
        randomly choose a subset of features, S_j;
        if $card(S_j) \leq card(T)$
            if $inConCal(S_j, D) \leq \delta$
                if $card(S_j) < card(T)$
                    $T = S_j$;
                    output S_j;
                else
                    append S_j to T;
                    output S_j as 'yet another solution'; }
    return T; }

Experiments and Analysis
We have conducted experiments to observe how the number of valid features (M') drops as the number of randomly generated feature sets increases. A total of 10 datasets, both artificial and real, are chosen for the experiments from the UC Irvine data repository [14] (Table 1). Two typical graphs are shown in Figure 1.

Data	LED24	Lung	Lymph	Mush	Par	Promo	Soy	Splice	Vote	Zoo
N	24	56	18	22	12	57	35	60	16	16
$M'(M)$	12(5)	19(4)	8(6)	8(4)	5(3)	15(4)	12(2)	19(9)	13(8)	9(5)
#Eval	230	155	215	22	25	187	42	284	215	25
#Max	2^{24}	2^{56}	2^{18}	2^{22}	2^{12}	2^{57}	2^{35}	2^{60}	2^{16}	2^{16}

Table 1. The number of valid features drops sharply in the first few hundred runs for all the datasets. N and M are defined earlier. #Eval is number of subsets generated and evaluated. #Max is maximum possible subsets.

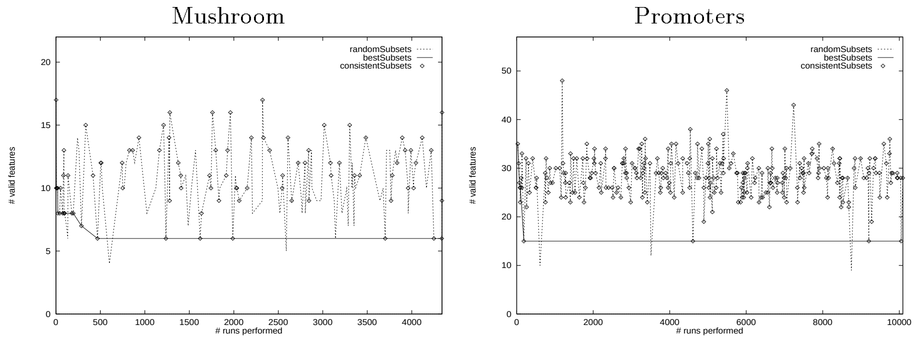


Fig. 1. The typical trends of the decreasing number of valid features versus the number of runs performed.

The *trend* found in all the experiments is that M' drops sharply from N in the first few hundred runs (one run means one feature set is randomly generated and evaluated). Afterwards, it takes quite a long time to further decrease M'. Some analysis would confirm this finding. A particular set has a probability of $1/2^N$ to be generated. At the beginning, the full set is a valid one. Many subsets can satisfy the inconsistency criterion. As M' decreases from N to M, fewer and fewer subsets can satisfy the criterion. However, the probability of a distinct set being generated is still $1/2^N$. That explains why the curves have a sharp drop in the beginning and then become flat.

2.2 ABB

Branch & Bound for feature selection was first proposed in [15]. It starts with a full set of features, removes one feature at a time. If the full set contains three features, the root is (1 1 1). Its child nodes are (1 1 0), (1 0 1), and (0 1 1), etc. When there is no restriction on expanding nodes in the search space, this could lead to an exhaustive search. However, if each node is evaluated by a measure U and an upper limit is set for the acceptable values of U, then Branch & Bound backtracks whenever an infeasible node is discovered. If U is monotonic, no feasible node is omitted as a result of early backtracking and, therefore, gained savings of search time do not violate the optimality of the selected subset. As was pointed out in [19], the measures used in [15] have disadvantages (non-monotonicity is one); the authors of [19] proposed approximate monotonicity.

ABB is a Branch & Bound algorithm where the bound is the inconsistency rate of the dataset 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 (its branch is pruned), otherwise, it grows to level $l+1$, in other words, one more feature will be removed.

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 $\wedge$ inConCal(S_2, D) $\leq \delta$)
            /* recursion */
            ABB (S_2, D); }}

The legitimacy test is to test whether a node (subset) is a child node of a pruned node. A node is illegitimate if it is a child node of a pruned one. Each node is represented by a binary vector where 1's stand for presence of a particular feature in that subset and 0's for its absence. The test is done by checking the Hamming distance between the child node under consideration and pruned nodes. If the Hamming distance with any pruned node is 1 (i.e., the difference of the two representative binary vectors is 1), the child node is the pruned node's child. Thus found all illegitimate nodes at a level.

ABB guarantees an optimal solution. However, a brief analysis can tell us that ABB's time performance can deteriorate if M is not large with respect to N. This issue is directly related to the size of the search space, or, how many nodes (subsets) have been generated. The other factors about time complexity of ABB are (1) time complexity of inconsistency checking, which is $O(P)$; and (2) time complexity of legitimacy test, which is $O(N)$. As they are relatively the same for each node, so the key factor is the search space. The search space of ABB is closely related to the number of relevant features. In general, the more the number of relevant features, the smaller the search space due to the early pruning of the illegitimate nodes.

In summary, our findings are: (1) LVF reduces the number of features quickly during the initial stage; but searches in the same way (i.e., blindly) after that, wasting the computing resource on generating subsets that are obviously illegitimate; and (2) on the other hand, ABB expands the search space quickly but is inefficient in reducing the search space although it guarantees optimal results.

3 Hybrid Search

The combination of LVF and ABB brings about a hybrid algorithm QBB. It runs LVF in the first phase and records all the subsets output by it. Next it runs ABB over each subset output by LVF. ABB, instead of starting from the complete feature set, now initializes the starting set by the subsets output by LVF, and keeps the minimum sized subsets so that ABB is more focused. In this way, LVF and ABB complement each other.

$\delta = inConCal(S, D)$; $T = S$;
QBB (S, D) {
    $S' = LVF(S, MaxTries, D)$;
    /* All legitimate subsets from LVF are in S' */
    $MinSize = card(S)$;
    for all S_j in S' {
        $T' = ABB(S_j, D)$;
        if $MinSize > card(T')$ {
            $MinSize = card(T')$;
            $T = S_j$; }}
    return T; }

There are two major issues affecting the efficacy of QBB: 1. when one should choose QBB over ABB, and 2. the crossing point at which ABB takes over from LVF. ABB may not terminate in a realistic time (a few hours) for datasets having large N but not so large M. So we restrict the size of N (say $N \leq 9$), no matter what the size of M is, ABB can stop in a reasonable time (more in Conclusion). The second issue is regarding the point where ABB takes over from LVF. If we allow only a certain amount of time to run QBB, the point at which ABB takes over from LVF is crucial for the efficiency of QBB. We conducted a number of experiments to determine a robust crossing point. Ten datasets (summarized in Table 2) with both P and N varying substantially are taken. We set five different total numbers of runs[1] (*TotalRuns* = 1K, 2K, 3K, 4K, and 5K) for LVF and ABB together. For each experiment 9 crossing points were set at every tenth of *TotalRuns*. Each of these experiments was conducted 50 times to average out the probabilistic nature of LVF in QBB. Hence, a total of 10*5*9*50 = 22500 experiments were conducted. In Figure 2, we show three graphs with different levels of details. Figure 2(a) displays the results for all the datasets when the *TotalRuns* is 3000; Figure 2(b) shows the average results over all the datasets for different *TotalRuns*; and Figure 2(c) is the average over all datasets and *TotalRuns*. These graphs show that results are better when the crossing point is around the middle of *TotalRuns* (i.e., 0.5**TotalRuns*) than when it is near the beginning or the end. The conclusion is that dividing the total time equally between LVF and ABB is a robust solution and is more likely to yield the best results. A simple analysis of this result would show that if the crossing point is quite early, then LVF might not have reduced the valid subset

[1] Here one run means generating and evaluating a subset of features.

size substantially for ABB to perform well[2], and if the crossing point is quite late, then the small sized subsets generated by LVF at the crossing point might not contain any minimal size subset.

	LED24	Lung	Lymph	Mush	Par	Promo	Soy	Splice	Vote	Zoo
D_{Tr}	800	21	100	4750	42	71	30	2126	300	50
D_{To}	1200	32	148	7125	64	106	47	3190	435	74
M	5	4	6	4	3	4	2	9	8	5
M_{QBB}	5.0	6.36	6.54	4.24	3.0	5.6	2.38	9.0	8.0	5.0

Table 2. QBB does well for most of the datasets. D_{Tr} - training data size for feature selection, D_{To} - total data size, M - minimal size, and M_{Avg} - average size by QBB.

4 Futher Experiments

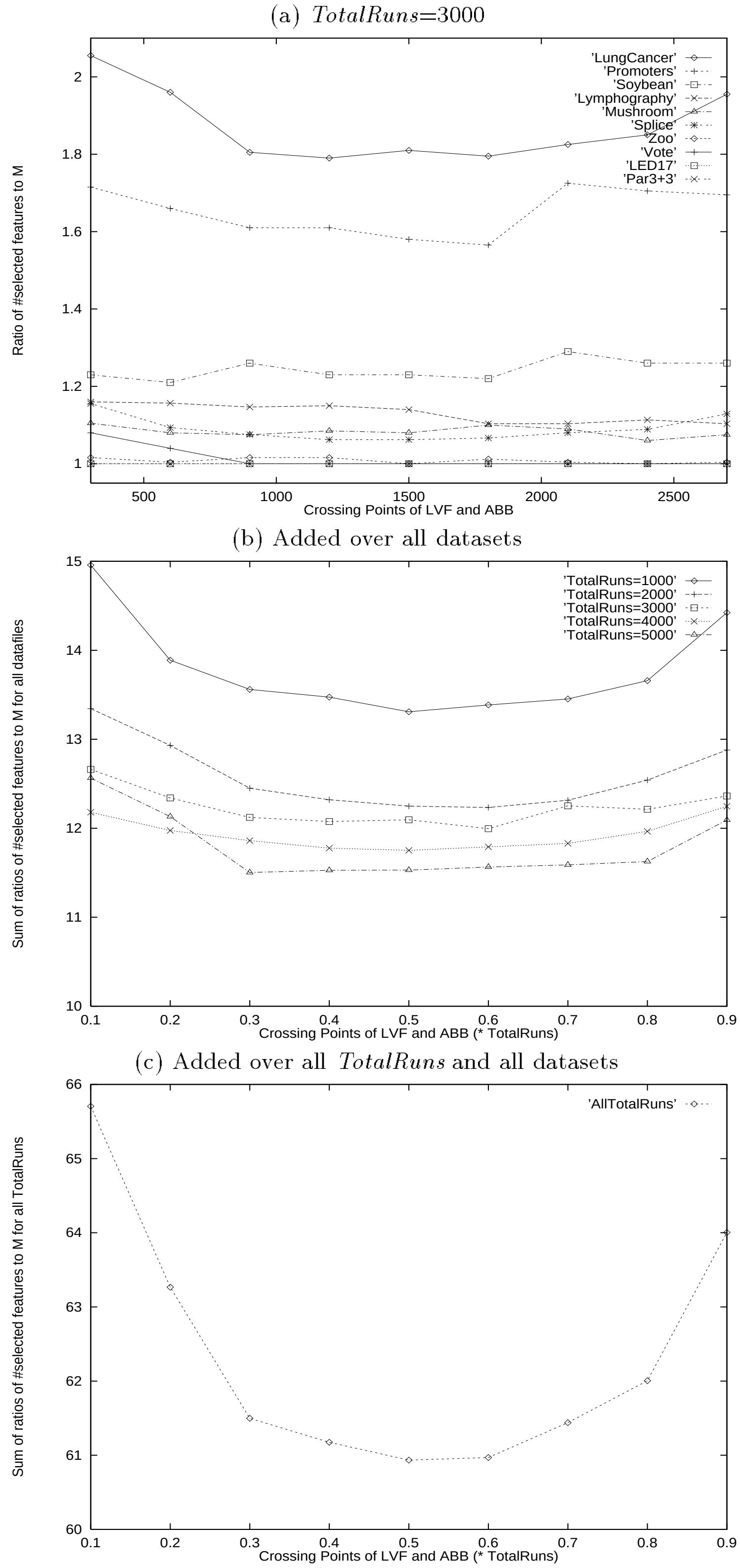
We want to verify that (1) QBB can find minimal size feature subsets most of the time; (2) it is faster than LVF, ABB, and FocusM in general; (3) Subsets chosen by QBB achieve nearly equal or better accuracy for two different kinds of classifiers (decision tree and neural networks) than the full feature sets do; and (4) QBB is particularly suitable for huge datasets. The experimental procedure is: (1) choose the datasets frequently used by the community; (2) run ABB to get the minimal size as reference; (3) compare the performance (average time and number of selected features) of QBB with that of LVF, FocusM and ABB; and (4) compare the accuracy of two different classifiers (C4.5 [16] and Back-propagation neural network [21]) over datasets before and after feature selection.

A total of 10 datasets, both artificial and real, are chosen for the experiments from the UC Irvine data repository [14]. The Par3+3 dataset is a modified version of the original Parity3 dataset. The target concept is the parity of three bits. It contains 12 features: 3 relevant, 3 redundant, others irrelevant.

Table 2 shows the results of QBB over the 10 datasets. To average out the probabilistic nature of LVF in QBB, each experiment is conducted 50 times. In each experiment *TotalRuns* is set to 5K and crossing point is 0.5 as per our previous finding. The experiments show that $M_{QBB} = M$ all the time for many datasets (e.g., LED-24, Par3+3, Vote), whereas for others M_{QBB} exceeds M by a small margin.

Figure 3 shows *a comparison of the performance (both average time and number of selected features) of QBB with that of LVF, FocusM, and ABB.* First ABB

[2] For some datasets, ABB by itself only does not terminate in realistic time (a few hours) as M is very small compared to N for these datasets.



ht

Fig. 2. Experimental results show that best results are obtained by equally dividing the time between LVF and ABB.

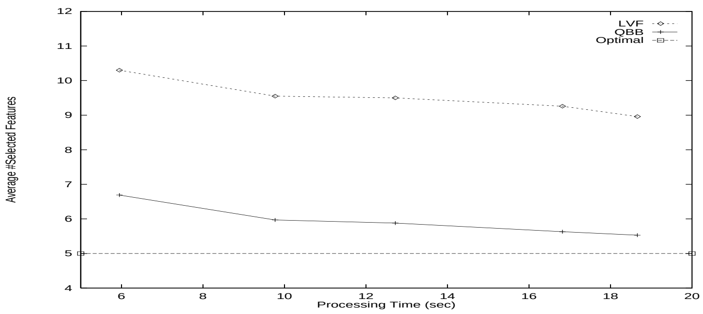


Fig. 3. Experimental results show that QBB performs well in comparison to FocusM, LVF, and ABB in general.

is run over the 10 datasets to find the M (minimal size) values. For comparison purpose we have calculated the average minimal value, M_{Avg}, over all datasets which is found to be 5. This value is used as a reference line in Figure 3. Out of the 4 competing algorithms, FocusM and ABB are deterministic, whereas LVF and QBB are non-deterministic due to their probabilistic nature. Hence, to compare LVF and QBB we chose 5 different sets of runs (i.e., *TotalRuns* = 1K, ... , 5K). QBB spends half of the time running LVF and the other half running ABB.Each experiment was repeated 50 times for each dataset for LVF and QBB, and the average size and time are reported in Figure 3. Notice that the outcomes of FocusM and ABB are not shown in the graph as the average time taken by them falls outside the range of the 'processing time' in the x-axis of the graph.For datasets having a large N and a small M values such as Lung Cancer, Promoters, Soybean, Splice datasets ABB takes very long time (a number of hours) to terminate. For datasets having large N value and substantially big M value such as Splice dataset FocusM takes many hours to terminate. The comparison in Figure 3 shows that QBB is more efficient both in average time and number of selected features compared to LVF, FocusM, and ABB. The average size of the subsets produced by QBB is close to M_{Avg} and it approaches to M_{Avg} with time. LVF is poor in finding small sized consistent subsets although it reaches a mediocre size very soon.

The error probability is often used as a validation criterion. We choose C4.5 decision tree and Back-propagation neural network as two classifiers for validation. Each dataset was divided into a training set (two-third of the original size) and a testing set (one-third of the original size). QBB is run over the training set to find a subset of features. For C4.5, we use the default settings, apply it to datasets before and after feature selection, and obtain the results of 10-fold cross-validation. This is repeated 10 times for each dataset and the average error rate and tree size are reported in Table 3. That is, QBB has been run 10 times and C4.5 100 times. It shows improvement for most of the datasets (9 out of 10) in C4.5's accuracy after feature selection.

	C4.5				Back-Propagation			
	Tree Size		Error Rate				Error Rate	
Data	Bef	Aft	Bef	Aft	Cycles	#HU	Bef	Aft
LED-24	19.0	19.0	0.0	0.0	1000	12	0.06	0.0
Lung	19.0	10.9	50.0	41.8	1000	28	75.0	75.0
Lymphography	26.9	22.1	21.8	21.4	7000	9	25.0	25.0
Mushroom	36.3	34.2	0.0	0.0	5000	11	0.0	0.0
Par3+3	12.0	15.0	41.9	0.0	1000	6	22.2	0.0
Promoters	21.4	8.2	26.3	22.1	2000	29	46.8	25.0
Soybean	7.1	9.4	2.5	2.0	1000	18	10.0	0.0
Splice	173.0	187.0	5.9	14.0	6000	30	25.64	42.33
Vote	14.5	14.2	5.3	5.3	4000	8	6.7	4.0
Zoo	17.8	17.7	7.8	6.6	4000	8	10.3	3.4

Table 3. Classifier results. #HU is number of Hidden Units.

Running Back-propagation involves the setting of some parameters, such as the network structure (number of layers, number of hidden units), learning rate, momentum, number of CYCLES (epochs), etc. In order to focus our attention on the effect of feature selection by QBB, we try to minimize the tuning of the parameters for each dataset. We fix 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 datasets. The experiment is carried out in two steps: (1) a trial run to find a proper number of CYCLES for each dataset which is determined by a sustained trend of no decrease of error; and (2) two runs on datasets with and without feature selection respectively using the number of CYCLES found in step 1. Other parameters remain fixed for the two runs in step 2. The results are shown in Table 3. In most cases, error rate decreases (6 out of 10) or does not change (3 out of 10) after feature selection.

The fact that the consistency criterion does not incorporate any bias relating to a particular classifier enables it to be used with a variety of different learning algorithms. As shown above, for the two different types of classifiers, selected features improve the performance in terms of lower error rates in most cases. On the other hand, since a set of features is deemed consistent if *any* function maps from the values of the features to the class labels, any algorithm optimizing this criterion may choose a small set of features that has a complicated function, while overlooking larger sets of features admitting simple rules. Although intuitively this should be relatively rare, it can happen in practice, as apparently was the case for the Splice dataset where both C4.5 and Back-propagation's performance deteriorate after feature selection.

QBB is quite helpful for datasets of huge N. For such datasets both Focus-M and ABB are inefficient because of their complete search from the empty set (as in Focus-M) or from the complete set (as in ABB). LVF also is not helpful because the hugeness of the dataset reduces the efficiency of a pure probabilistic search. To prove this point we ran LVF, QBB, ABB, and Focus-M on the Letter dataset with 20,000 patterns. LVF was not able to reduce the size in any of its run. Focus-M took more than 2 days to finish and produce an minimal size feature set of 11 features. ABB took more than 7 hours to produce the minimal size feature set. For QBB in all its run with totalRuns set to 5000, it produced the minimal size feature sets in less than 2 hours.

5 Concluding Remarks

We have proposed a hybrid algorithm (QBB) for feature selection which tries to maximize the use of both probabilistic and exhaustive search. The hybrid algorithm is shown to be robust, probabilistic, and approaches an optimal solution faster than previously proposed methods such as FocusM, LVF and ABB in a general setting. Two different types of classifiers (C4.5, Back-propagation) are employed to verify whether features of various datasets selected by QBB can pass the ultimate error probability criterion.

In this work, we have investigated several feature selection algorithms (FocusM, LVF, ABB). The guideline of taking maximum advantages of these different algorithms can be summarized into a simple procedure as follows:

If M is known as SMALL
  Apply **FocusM**
Else If $N \leq 9$
  Apply **ABB**
Else
  Apply **QBB**

SMALL is a floating concept which depends on the time available. Furthermore, M is usually not known *a priori*, though sometimes, we can pre-specify a number in some applications (e.g., visualization). So, without any knowledge about M, we often end up making a choice between **ABB** and **QBB**.

Acknowledgement

The authors wish to thank P. M. Long for his insightful comments.

References

1. H. Almuallim and T.G. Dietterich. Learning with many irrelevant features. In *Proceedings of the Ninth National Conference on Artificial Intelligence*, pages 547–552, Anaheim, California, 1991. AAAI Press/The MIT Press, Menlo Park, California.
2. A.L. Blumer, A. Ehrenfeucht, D. Haussler, and M.K. Warmuth. Occam's razor. In J.W. Shavlik and T.G. Dietterich, editors, *Readings in Machine Learning*, pages 201–204. Morgan Kaufmann, 1990.
3. G. Brassard and P. Bratley. *Fundamentals of Algorithms*. Prentice Hall, New Jersey, 1996.

4. M. Dash and H. Liu. Feature selection methods for classifications. *Intelligent Data Analysis: An International Journal*, 1(3), 1997. `http://www-east.elsevier.com/ida/free.htm`.
5. P.A. Devijver and J. Kittler. *Pattern Recognition: A Statistical Approach.* Prentice Hall International, 1982.
6. K. Kira and L.A. Rendell. The feature selection problem: Traditional methods and a new algorithm. In *Proceedings of the Tenth National Conference on Artificial Intelligence*, pages 129–134. Menlo Park: AAAI Press/The MIT Press, 1992.
7. R. Kohavi. *Wrappers for performance enhancement and oblivious decision graphs.* PhD thesis, Department of Computer Science, Standford University, Stanford, CA, 1995.
8. D. Koller and M. Sahami. Toward optimal feature selection. In L. Saitta, editor, *Machine Learning: Proceedings of the Thirteenth International Conference (ICML-96), July 3-6, 1996*, pages 284–292, Bari, Italy, 1996. San Francisco: Morgan Kaufmann Publishers.
9. I. Kononenko. Estimating attributes : Analysis and extension of RELIEF. In F. Bergadano and L. De Raedt, editors, *Proceedings of the European Conference on Machine Learning, April 6-8*, pages 171–182, Catania, Italy, 1994. Berlin: Springer-Verlag.
10. P. Langley. Selection of relevant features in machine learning. In *Proceedings of the AAAI Fall Symposium on Relevance.* AAAI Press, 1994.
11. H. Liu and H. Motoda. *Feature Selection for Knowledge Discovery Data Mining.* Boston: Kluwer Academic Publishers, 1998.
12. H. Liu, H. Motoda, and M. Dash. A monotonic measure for optmial feature selection. In C. Nedellec and C. Rouveirol, editors, *Machine Learning: ECML-98, April 21 - 23, 1998*, pages 101–106, Chemnitz, Germany, April 1998. Berlin Heidelberg: Springer-Verlag.
13. H. Liu and R. Setiono. A probabilistic approach to feature selection - a filter solution. In L. Saitta, editor, *Proceedings of International Conference on Machine Learning (ICML-96), July 3-6, 1996*, pages 319–327, Bari, Italy, 1996. San Francisco: Morgan Kaufmann Publishers, CA.
14. 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.
15. 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.
16. J.R. Quinlan. *C4.5: Programs for Machine Learning.* Morgan Kaufmann, 1993.
17. T.W. Rauber. *Inductive Pattern Classification Methods - Features - Sensors.* PhD thesis, Dept. of Electrical Engineering, Universidade Nova de Lisboa, Lisboa, 1994.
18. J. C. Schlimmer. Efficiently inducing determinations : a complete and systematic search algorithm that uses optimal pruning. In *Proceedings of the Tenth International Conference on Machine Learning*, pages 284–290, 1993.
19. W. Siedlecki and J. Sklansky. On automatic feature selection. *International Journal of Pattern Recognition and Artificial Intelligence*, 2:197–220, 1988.
20. S. Watanabe. *Pattern Recognition: Human and Mechanical.* Wiley Interscience, 1985.
21. A. Zell and et al. Stuttgart neural network simulator (SNNS), user manual, version 4.1. Technical Report 6/95, Institute for Parallel and Distributed High Performance Systems (IPVR), University of Stuttgart, FTP: ftp.informatik.uni-stuttgart.de/pub/SNNS, 1995.