Floating Search Methods for Feature Selection with Nonmonotonic Criterion Functions

P. Pudil[*], F.J. Ferri[†], J. Novovičová[*] and J. Kittler

Dept. of Electronic & Electrical Engineering
University of Surrey. Guildford GU2 5XH, UK

1 Introduction

Feature selection (FS) constitutes one of the two principal phases of pattern recognition system design, the other being the design of pattern classification stage which employs the selected features. Basically, it is the process of choosing the input to the pattern recognition system. Accordingly, the main goal of FS is to select a subset of d features from the given set of D measurements, $d < D$, without significantly degrading (or possibly even improving due to the "peaking phenomenon" [5]) the performance of the recognition system. Assuming that a suitable criterion function has been chosen to evaluate the effectiveness of feature subsets, FS is reduced to a search problem that detects an optimal feature subset based on the selected measure. Owing to the fact that the optimization is carried out in a combinatorial search space aggravated by various restrictive conditions posed by real problems such as a limited sample size of training sets, the problem is very complex.

Though according to Cover [3] an exhaustive search, whether explicit or implicit (as afforded by the *Branch and Bound* –B&B– method [9]), is a necessary procedure to guarantee the finding of an optimal subset, in most practical applications this approach is computationally prohibitive and the mainstream of research on FS has thus been directed toward sequential suboptimal search methods. A comprehensive treatment of these methods can be found in [4]. Among the suboptimal search procedures the *plus-l-take away r* method (or (l, r) search method) of Stearns [12] and their recent extensions, have repeatedly been demonstrated to be most effective [8, 2].

The heuristic basis of most sequential search methods is that the criterion of feature set effectiveness used in conjunction with these methods is monotonic. This implies that when adding a feature to the current set, the value of the criterion function does not decrease. Not surprisingly, the majority of FS studies reported in the literature involve the use of class separability measures which are known to satisfy the set inclusion monotonicity property [4].

However, as pointed out by Siedlecki and Sklansky [11], Ben-Bassat [1] and other researchers, probabilistic separability measures do not induce over any arbitrary set of features the same preference order as would be obtained by comparing the classification system error rates. This is due to the fact that classifier error rates capture two basic performance aspects: i) probabilistic class separability and ii) any structural errors imposed by the form of the classifier [7]. As the second aspect is not reflected in FS based exclusively on probabilistic separability measures, the resulting features may perform poorly

[*]Permanently at Inst. of Information Theory and Automation. Academy of Sciences of the Czech Republic
[†]Contact person. Permanently at Dept. Informàtica i Electrònica. Universitat de València. SPAIN

when applied as the input to the classifier. For this reason the only promising and legitimate way of evaluating features must be through the error rate of the classifier being designed.

Barring any small sample set effects, the first component of the system error, as alluded to earlier, is monotonic. However, structural errors can cause nonmonotonicity and consequently a complete breakdown of the premise behind sequential search methods. In order to overcome this problem, Siedlecki and Sklansky [11] explored the idea of approximate monotonicity together with other methods like genetic algorithms and Monte Carlo approaches to develop search algorithms which are tolerant to nonmonotonicity. Despite some progress, the advocated optimisation techniques are not yet satisfactory. They appear to cope with nonmonotonicity and perform well for FS problems of moderate sizes, but this property does not seem to extend to large scale problems.

In this paper we return to the sequential selection procedures with backtracking and show that a family of suboptimal search algorithms which we call the Floating Search methods are very efficient and effective even on problems of high dimensionality involving nonmonotonic feature selection criterion functions. The Floating Search methods are related to the *plus-l-take away r* algorithm, but in contrast to the latter, the number of forward and backtracking steps is dynamically controlled instead of being fixed beforehand. The purpose of this paper is to present the Floating Search procedures and show that they can cope with nonmonotonic feature set criterion functions. We shall demonstrate that the Floating Search procedures construct in parallel the feature sets of all dimensionalities up to a specified threshold. By means of sequential forward and backward selection these sets are updated whenever the modification results in a better performance. In consequence, the resulting feature sets, as in the case of the (l, r) sequential algorithms, are not necessarily nested. By the same token, the selection process can correct for any effects caused by nonmonotonicity of the FS criterion.

Although the Floating Search methods can be applied in both the bottom up and top down mode, we focus our discussion on the bottom up approach which is invariably the only feasible way of selecting features in high dimensional problems. We shall also illustrate the advantages of performing FS in conjunction with the classification error rate as a FS criterion instead of indirect probabilistic class separability measures. We shall also observe that although the Floating Search methods do not consistently outperform other sequential search algorithms with backtracking, they are irrefutably an important addition to the armoury of FS techniques.

The paper can be outlined as follows. In Section 2 the Floating methods are described using the same algorithmic format as (l, r)-search and their main similarities and differences are overviewed. The behaviour of Floating Search methods used in conjunction with nonmonotonic criteria is discussed in Subsection 2.3. Section 3 describes the results of experiments in FS using the classification system error rate as a measure of FS effectiveness to demonstrate the tolerance of Floating Search methods to nonmonotonicity. Finally, the work is summarised and the main conclusions are drawn in Section 4.

2 Sequential Search Algorithms

2.1 The *plus* l *− take away* r Algorithm

The well-known Sequential Forward Selection (SFS) and Sequential Backward Selection (SBS) are step-optimal only since the best (the worst) feature is always added (discarded) in SFS and SBS, respectively. This results in nested feature subsets without any chance to correct the decision in later steps, causing the performance to be often far from optimal.

A definitive improvement can be obtained by combining SFS and SBS to avoid the nesting effect. The *plus l-take away r* method [12] consists of applying Sequential Forward Selection (SFS) l times followed by r steps of Sequential Backward Selection (SBS) with this fixed cycle of forward and backward selection repeated until the required number of features is reached. Therefore, it results in a forward method if $l > r$ or a backward one if $l < r$. In this case, features that have been previously added can be removed in posterior steps thus avoiding the nesting effect. In this case, the method allows a "fixed backtracking" defined by the values of l or r depending whether the search is top down or bottom up.

The *plus l – take away r* algorithm and, consequently, the SFS ((1,0)-search) and SBS ((0,1)-search) algorithms, can be described in an algorithmic way as follows:

***plus l – take away r* Algorithm**

Input:

$Y = \{y_j \mid j = 1, \ldots, D\}$ //available measurements//

Output:

$X_k = \{x_j \mid j = 1, \ldots, k,\ x_j \in Y\},\ k = 0, 1, \ldots, D$

Initialisation:

`if` $l > r$ `then` $k := 0$; $X_0 := \emptyset$; `go to` **Step 1**

`else` $k := D$; $X_D := Y$; `go to` **Step 2**

Termination:

Stop when k equals the number of features required

Step 1 *(Inclusion)*

`repeat` l `times`

$x^+ := \arg \max_{x \in Y - X_k} J(X_k + x)$ {the most significant feature[1] *with respect to* X_k

$X_{k+1} := X_k + x^+$; $k := k + 1$

`go to` **Step 2**

Step 2 *(Exclusion)*

`repeat` r `times`

$x^- := \arg \max_{x \in X_k} J(X_k - x)$ {the least significant feature *in* X_k

$X_{k-1} := X_k - x^-$; $k := k - 1$

`go to` **Step 1**

This procedure can be generalised to search a larger region of the space by substituting l steps forward by the search of the best superset that can be formed by adding l features (the same for backward) [6]. Even though the problem of nested features can be partially overcome with this procedure, another problem arises; there is no way of predicting the best values of l and r to obtain good enough solutions with a moderate amount of computation.

2.2 Sequential Floating Selection

The idea behind the previous methods aimed at counteracting the nesting effect, can be more efficiently implemented by considering conditional inclusion and exclusion of features controlled by the value of the criterion itself. The Sequential Floating Forward Selection (SFFS) procedure consists of applying after each forward step a number of backward steps as long as the resulting subsets are better than the previously evaluated ones at that level [10]. Consequently, there are no backward steps at all if the performance cannot be improved. The same applies for the Sequential Floating Backward Selection (SBFS) procedure. Thus backtracking in these algorithms is controlled dynamically and, as a consequence, no parameter setting is needed at all.

The SFFS method can be described algorithmically in a similar way to the previous method as follows:

[1] x^+ satisfies $J(X_k + x^+) = \max_{x \in Y - X_k} J(X_k + x)$

SFFS Algorithm

Input:

$Y = \{y_j \mid j = 1, \ldots, D\}$ //available measurements//

Output:

$X_k = \{x_j \mid j = 1, \ldots, k,\ \ x_j \in Y\},\ \ k = 0, 1, \ldots, D$

Initialisation:

$X_0 := \emptyset; \quad k := 0$

(in practice one can begin with $k = 2$ by applying SFS twice)

Termination:

Stop when k equals the number of features required

Step 1 *(Inclusion)*

$x^+ := \arg \max_{x \in Y - X_k} J(X_k + x)$ $\left\{\text{the most significant feature } \textit{with respect to } X_k\right.$

$X_{k+1} := X_k + x^+; \ \ k := k + 1$

Step 2 *(Conditional Exclusion)*

$x^- := \arg \max_{x \in X_k} J(X_k - x)$ $\left\{\text{the least significant feature } \textit{in } X_k\right.$

```
if J(X_k - {x^-}) > J(X_{k-1}) then
      X_{k-1} := X_k - x^- ;  k := k - 1
      go to Step 2
else
      go to Step 1
```

The backward counterpart of this algorithm can be obtained in a straightforward manner by substituting inclusion by exclusion and the initialisation in the previous description. Both algorithms allow a "self-controlled backtracking" so they can eventually find good solutions by adjusting the trade-off between forward and backward steps dynamically. It is possible to say that, in a certain way, they compute only what they need without any parameter setting.

2.3 Discussion

The heuristic basis of most of the sequential feature set search algorithms is the assumption that the FS criterion is monotonic, that is any change in feature set size and therefore feature set information content is positively correlated with the change in the value of the criterion function. Thus if a feature set is augmented, the criterion function value will increase. However, with the exception of the B&B algorithm, the selection or elimination of features at any stage of sequential search algorithms does not rely on the monotonicity property. In other words the process of feature set augmentation, say from cardinality k to $k+1$ is performed without exploiting any relation between the values of the criterion function of the augmented set and of the set X_k. More specifically, given a set X_k, the augmented set X_{k+1} is determined as $X_{k+1} = X_k + x^+$ such that

$$J(X_k + x^+) = \max_{x \in Y - X_k} J(X_k + x)$$

Note that the value of $J(X_{k+1})$ can in principle be smaller than $J(X_k)$. In other words the set inclusion monotonicity property plays no explicit role in determining feature x^+. The same comment applies to the process of set reduction.

It follows that both the *(l,r)* and Floating Search algorithms can cope with nonmonotonic criteria. The particular advantage of the Floating Search methods over the *(l,r)* search is that they can make more than one sweep through feature set subsets to achieve good performance. Although this does not guarantee that the results obtained with Floating Search methods will always be superior to those yielded by *(l,r)* schemes, in practice we found that searching with dynamic backtracking leads to a very robust performance and, if one had to choose between feature set search methods then the Floating Search procedures would be the winner. Of course, a more sensible approach is to select features by applying algorithms with both fixed and dynamic backtracking and selecting the overall best results for each feature set cardinality. From this point of view Floating Search methods represent an indispensable addition to the pattern recognition system design tool box.

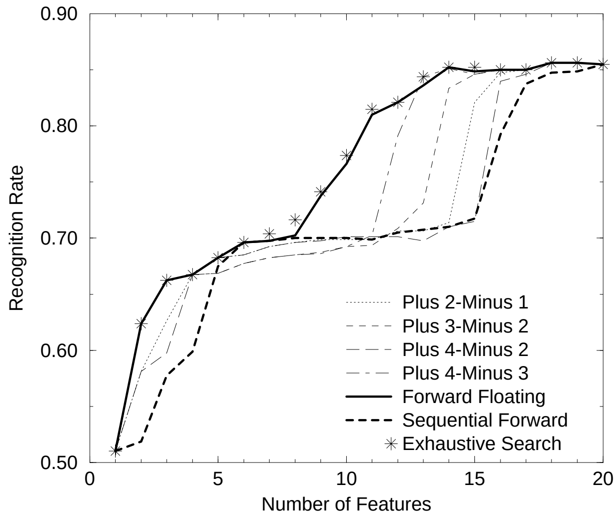

Figure 1: Performance of different forward methods on the diagnostic problem.

3 Experimental Study

We consider first the same 20-dimensional diagnostic problem used in previous work [6, 2]. However, the criterion function used is the recognition rate of the Gaussian classifier. As it can be seen from Figures 1 and 2, the use of a nonmonotonic criterion function does not modify the main conclusions of previous studies with respect to the performance of sequential methods used in conjunction with probabilistic separability measures [2]. The superiority of the SFFS method when compared to other forward methods is clear from Figure 1. Though in the case of backward methods the superiority of Floating search is not as significant as in the forward case, we can state that it is at least as good as the best sequential method. Most importantly, SFFS and SBFS methods yield results very close to the optimal ones obtained by exhaustive search.

To study the efficiency of the methods we consider in this paper the number of feature subsets evaluated rather than the actual time required by the procedures. The number of subsets evaluated by the floating procedures depends on the particular data and ranges from 500 for the SBFS method (comparable to the (1,2)-search), to 1200 for the SFFS method (comparable to the (4,3)-search). This result seems to support the idea that floating methods do in fact as many computation as needed to optimize the solution. Consequently, in the forward case the SFFS clearly improves the results of the (l, r)-search by doing as much computation as the (4,3) method. But, in the (easier) backward case, the SBFS method obtains nearly the optimal result with less computation than the (1,2)-search.

To test the different algorithms in a larger search space we also consider a document recognition problem. The problem involves discrimination between correct and defective records of banking documents consisting of 360 optical measurements. Previous experimentation showed that both classes could be conveniently modelled by Gaussian distributions. For this very large-dimensional problem only forward procedures are taken into account due to both computational and numerical problems. Consequently, feature subsets of size 1 to 80 only are obtained using the different methods. The results are shown in Figure 3.

Even though all the methods but the SFS obtain comparable solutions as the number of features approaches 80, it is possible to see from Figure 3 that the SFFS method is clearly superior from 25 to 60 features while the (4,3)-search is slightly better around 60 features. This fact emphasises that with heuristic search methods, regardless how sophisticated they are, it is impossible to achieve an overall superiority of one method over another. Nevertheless, the SFFS method gives an appropriate efficiency-effectiveness trade-off without requiring any parameter setting at all. In particular, the SFFS method performs around 60000 subset evaluations (comparable to (2,1)) while the (4,3) method does approximately 120000.

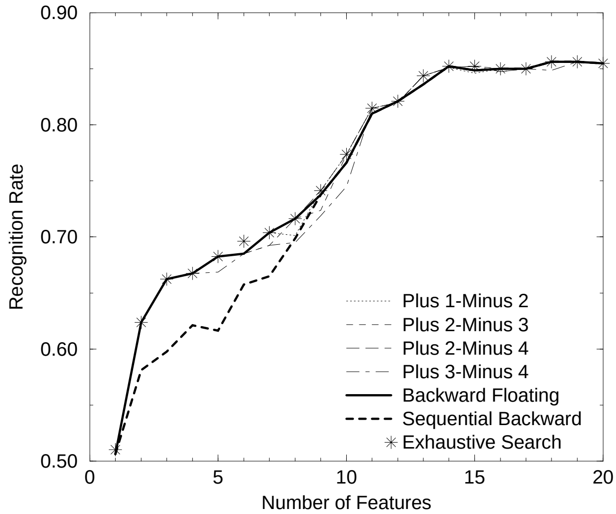

Figure 2: Performance of different backward methods on the diagnostic problem.

To test the same methods under more "apparent" nonmonotonicity (i.e. with a clear peak in the criterion function), we have selected a subproblem of the document recognition problem by arbitrarily taking 40 measurements from the 360 given. The only interesting point of this experiment is to show that the SFFS method behaves properly when a peak is encountered. From Figure 4 it is clear that the SFFS method is the only one that detects the peak in the recognition rate. We can see that though it does not yield the best results for all the values of d, it constitutes a good compromise and the solution it gives for 29 features seems to be intuitively the best. Again, with respect to the number of subsets evaluated, 5000, the method is comparable to the (4,3)-search.

4 Concluding Remarks

This paper presents the Floating Search algorithms as an advanced version of the well known family of sequential search procedures with backtracking, i.e. the (l, r) search procedures. The concept

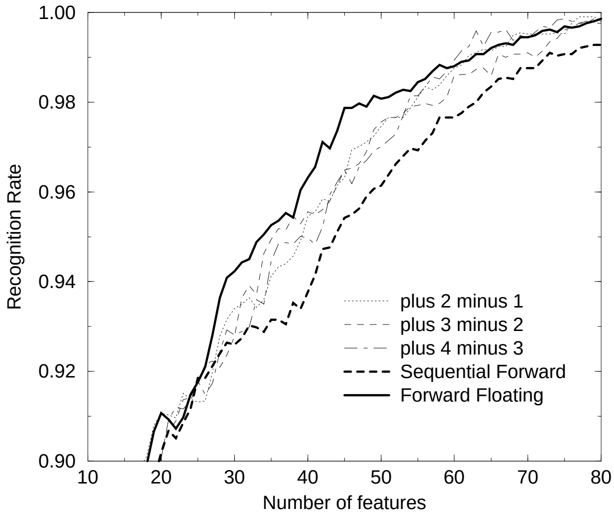

Figure 3: Performance of SFFS and (l, r) methods on the document recognition problem.

of conditional backtracking is introduced and its main advantages as far as feature subset search is concerned are discussed and demonstrated using experiments involving real data. In particular, the Floating Search methods are shown to be specially suited in conjunction with nonmonotonic feature selection criterion functions. We have shown that the Floating Search methods represent a good compromise between computational efficiency and performance. They appear to produce results very close to those obtained using the B&B algorithm without none of its main drawbacks such as computation and monotonicity requirements.

Although the main conclusions concerning Floating Search are highly positive, some open questions still remain. First, there is no theoretical bound on the computational cost of the algorithms due to their heuristic nature. In order to avoid excessive computation, the algorithms could be modified by constraining the maximum level of backtracking. Also, the consideration of both conditional inclusion and exclusion simultaneously could lead to improved Floating Search algorithms in terms of computational efficiency. These issues will be addressed in our future work.

References

[1] M. Ben-Bassat. Irrelevant features in pattern recognition. *IEEE Transactions on Computers*, C-27:746–749, August 1978.

[2] N. Choakjarernwanit. *Feature Selection in statistical pattern recognition.* PhD thesis, University of Surrey, England, 1992.

[3] T. M. Cover and J. M. Van Campenhout. On the possible orderings in the measurement selection problem. *IEEE Transactions on Systems Man and Cybernetics*, SMC-7:657–661, September 1977.

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

[5] R. Duda and P. Hart. *Pattern Classification and Scene Analysis.* Wiley, 1973.

[6] J. Kittler. Feature set search algorithms. In *Pattern Recognition and Signal Processing, C. H. Chen, Ed.*, pages 41–60, The Netherlands: Sijthoff and Noordhoff, 1978.

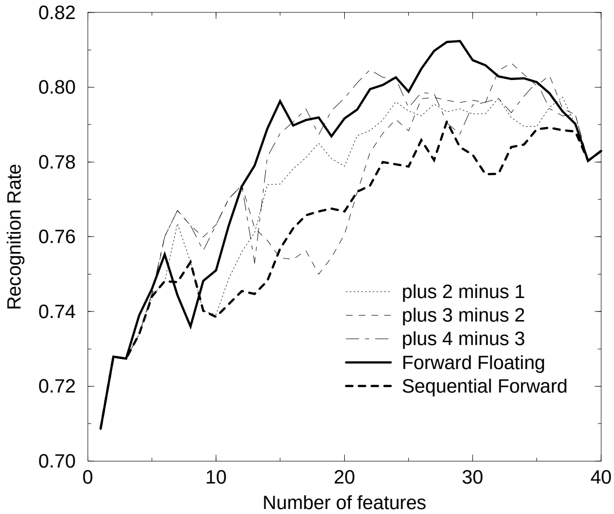

Figure 4: Performance of SFFS and (l, r) methods on a 40-dimensional document recognition subproblem.

[7] J. Kittler. Computational problems of feature selection pertaining to large data sets. In E. S. Gelsema and L. N. Kanal, editors, *Pattern Recognition in Practice*. North Holland, Amsterdam, 1980.

[8] J. Kittler, A. Etemadi, and N. Choakjarernwanit. Feature selection and extraction in pattern recognition. In *Pattern Recognition and Image Processing in Physics, 37th Scottish Universities Summer School in Physics, R. A. Vaughan, Ed.*, pages 81–100, Adam Hilger, Bristol, 1991.

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

[10] P. Pudil, J. Novovičová, and S. Bláha. Statistical approach to pattern recognition: Theory and practical solution by means of PREDITAS system. *Kybernetika*, 27:Supplement, 1–78, 1991.

[11] W. Siedlecki and J. Sklansky. On automatic feature selection. *International Journal of Pattern Recognition and Artificial Intelligence*, 2(2):197–220, June 1988.

[12] S. D. Stearns. On selecting features for pattern classifiers. In *Third Int. Conf. on Pattern recognition*, pages 71–75, Coronado, CA, 1976.