ELSEVIER

Pattern Recognition Letters 20 (1999) 1157–1163

Pattern Recognition Letters

www.elsevier.nl/locate/patrec

Adaptive floating search methods in feature selection ☆

P. Somol [a,b,c,*], P. Pudil [a,c], J. Novovičová [a,c], P. Paclík [a,c]

[a] *Department of Pattern Recognition, Institute of Information Theory and Automation, Academy of Sciences of the Czech Republic, 182 08 Prague 8, Czech Republic*

[b] *Faculty of Mathematics and Physics, Charles University, Prague, Czech Republic*

[c] *Joint Laboratory of Faculty of Management, University of Economics, Prague and Institute of Information Theory and Automation, Czech Academy of Sciences, Czech Republic*

Abstract

A new suboptimal search strategy for feature selection is presented. It represents a more sophisticated version of "classical" floating search algorithms (Pudil et al., 1994), attempts to remove some of their potential deficiencies and facilitates finding a solution even closer to the optimal one. © 1999 Elsevier Science B.V. All rights reserved.

Keywords: Pattern recognition; Feature selection; Search methods

1. Introduction

In feature selection, a search problem of finding a subset of d features from a given set of D measurements, $d < D$, has been of interest for a long time. Since the optimal methods (exhaustive search or the Branch-and-Bound method which is restricted to monotonous criteria) are not suitable for high-dimensional problems, research has concentrated on suboptimal search methods. Two well-known basic approaches to the required feature set construction are usually recognized: the "bottom up" and "top down" one.

A number of search methods has been developed, starting with *sequential backward selection* (SBS) and its "bottom up" counterpart known as *sequential forward selection* (SFS). Both of them suffer from the so-called "nesting effect". Attempts to prevent the nesting of feature subsets led to the development of the *Plus-l-Minus-r* search method by Stearns (1976) and to generalization of SBS, SFS and Plus-*l*-Minus-*r* algorithms proposed by Kittler (1978).

According to the comparative study made by Jain and Zongker (1997), probably the most effective known suboptimal methods are currently the sequential floating search methods, proposed by Pudil et al. (1994). In comparison to the Plus-*l*-Minus-*r* method, the "floating" search treats the "nesting problem" even better, since there is no need to specify any parameters such as *l* or *r*. The number of forward (adding)/backward (removing) steps is determined dynamically during the method's run so as to maximize the criterion function (see Fig. 1).

2. Preliminaries

Pudil et al. (1991, 1994) presented the definitions of *individual significance* of a single feature

☆ Electronic Annexes available. See www.elsevier.nl/locate/patrec.

* Corresponding author.

PII: S 0 1 6 7 - 8 6 5 5 (9 9) 0 0 0 8 3 - 5

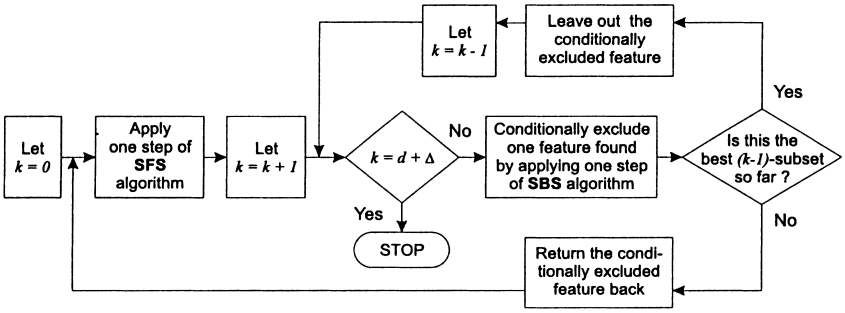
Let $k = 0$
Apply one step of SFS algorithm
Let $k = k + 1$
$k = d + \Delta$
Yes
STOP
No
Conditionally exclude one feature found by applying one step of SBS algorithm
Is this the best (k-1)-subset so far ?
Yes
Leave out the conditionally excluded feature
Let $k = k - 1$
No
Return the conditionally excluded feature back

Fig. 1. Simplified flow chart of SFFS algorithm.

with respect to the set and *in the set*. Before discussing the adaptive floating search algorithms formally, the following generalization of these definitions has to be introduced.

Let $X_k = \{x_i : 1 \leqslant i \leqslant k,\ x_i \in Y\}$ be the set of k features from the set $Y = \{y_i : 1 \leqslant i \leqslant D\}$ of D available features. Let T_o be generally the tuple of o features. The value $J(T_o)$ of the feature selection criterion function if only the features t_i, $i = 1, 2, \ldots, o$, $t_i \in T_o$ are used will be called the *individual significance* $S_0(T_o)$ *of the feature o-tuple*.

The *significance* $S_{k-o}(T_o)$ *of the feature o-tuple* $T_o = \{t_i : 1 \leqslant i \leqslant o,\ t_i \in X_k\}$ *in the set* X_k is defined by

$$S_{k-o}(T_o) = J(X_k) - J(X_k \setminus T_o). \tag{1}$$

The *significance* $S_{k+o}(U_o)$ *of the feature* o-*tuple* $U_o = \{u_i : 1 \leqslant i \leqslant o,\ u_i \in Y \setminus X_k\}$ from the set $Y \setminus X_k$ *with respect to the set* X_k is defined by

$$S_{k+o}(U_o) = J(X_k \cup U_o) - J(X_k). \tag{2}$$

Denote by T_o^i the ith o-tuple belonging to the set of all $\Theta = \binom{k}{o}$ possible o-tuples from X_k, $1 \leqslant i \leqslant \Theta$. We shall say that the feature o-tuple T_o^n from the set X_k is:

1. the *most significant* (*best*) feature o-tuple *in the set* X_k if
$$S_{k-o}(T_o^n) = \max_{1 \leqslant i \leqslant \Theta} S_{k-o}(T_o^i)$$
$$\Rightarrow J(X_k \setminus T_o^n) = \min_{1 \leqslant i \leqslant \Theta} J(X_k \setminus T_o^i); \tag{3}$$
2. the *least significant* (*worst*) feature o-tuple *in the set* X_k if
$$S_{k-o}(T_o^n) = \min_{1 \leqslant i \leqslant \Theta} S_{k-o}(T_o^i)$$
$$\Rightarrow J(X_k \setminus T_o^n) = \max_{1 \leqslant i \leqslant \Theta} J(X_k \setminus T_o^i). \tag{4}$$

We shall say that the feature o-tuple U_o^r from the set $Y \setminus X_k$ is:

1. the *most significant* (*best*) feature o-tuple *with respect to the set* X_k if
$$S_{k+o}(U_o^r) = \max_{1 \leqslant i \leqslant \Psi} S_{k+o}(U_o^i)$$
$$\Rightarrow J(X_k \cup U_o^r) = \max_{1 \leqslant i \leqslant \Psi} J(X_k \cup U_o^i), \tag{5}$$
where $\Psi = \binom{D-k}{o}$ is the number of all the possible o-tuples from $Y \setminus X_k$;
2. the *least significant* (*worst*) feature o-tuple *with respect to the set* X_k if
$$S_{k+o}(U_o^r) = \min_{1 \leqslant i \leqslant \Psi} S_{k+o}(U_o^i)$$
$$\Rightarrow J(X_k \cup U_o^r) = \min_{1 \leqslant i \leqslant \Psi} J(X_k \cup U_o^i). \tag{6}$$

Remark. For $o = 1$ all the terms relating to the feature o-tuple significance coincide with the terms relating to the *individual significance* of a feature.

3. Adaptive floating search (AFS) properties

For the sake of simplicity, let us denote original floating search methods (SFFS and SBFS) together as "classical floating search" methods and denote them by CFS. CFS methods use only single feature adding or removing, respectively, in the course of the algorithm.

Our new search strategy aims to utilize the best of both generalized strategies and classical floating strategies. The course of search is similar to that of CFS, but the individual search steps are generalized. However, the new algorithm is by no means just a generalized version of CFS.

The basic generalization of CFS would be to replace the simple SFS or SBS steps inside the CFS procedure by their generalized versions GSFS(o) or GSBS(o), respectively. Unfortunately for a potential user, it is generally not known which value of o to choose to get the best results. Moreover, if the user chooses the value of o too high for his particular problem, this leads to useless increase of computing time.

As opposed to the above mentioned generalized methods, the AFS method does not need the user-specified level of generalization. This level (value of o) is determined dynamically in the course of the search according to the current situation so as to achieve better results. In this sense, AFS brings about a similar improvement in comparison with the generalized search strategies as the CFS search brought about in comparison with simple "non-generalized" strategies (CFS introduced dynamical "floating" of adding/removing steps).

Because of the time-exacting character of generalized steps (especially when used in high-dimensional problems) we introduced a user defined parametric limit $r_{\max}$, restricting the maximum generalization level which the method can use. The current generalization level, which changes in the course of search, is denoted by o. The AFS is called "adaptive" because of its ability to adjust the limit under which the actual generalization level can be automatically set. Simply said, the nearer the current subset size (k) is to the final one (d), the higher is the generalization limit. This characteristic aims to save computing time by limiting the generalization levels while the current subset is still far from the desired one. Therefore, we introduce the variable r representing the actual generalization limit for a given dimension.

To summarize, $r_{\max}$ is a user-specified absolute generalization limit, r is the actual generalization limit determined adaptively by the algorithm for the current subset ($r \leqslant r_{\max}$ always holds), o is the current generalization level depending on the current situation ($o \leqslant r$ always holds).

Remark. For $r_{\max} = 1$ the AFS is identical to classical floating search.

Adaptive determination of r is done as follows: at the beginning of every forward or backward algorithm phase, respectively:
1. if $|k - d| < b$, let $r = r_{\max}$
2. else if $|k - d| < b + r_{\max}$, let $r = r_{\max} + b - |k - d|$
3. else let $r = 1$

Here b denotes the neighbourhood of the final dimension, where the highest generalization levels are allowed. Basically it is possible to set $b = 0$. The adaptive setting of r and the meaning of parameter b is shown in Fig. 2.

Thus, in the generalized course of the AFS algorithm, $o = 1$ is used in usual algorithm stages (e.g., in the beginning). Only special algorithm stages (when conditional forward, respectively backward steps brought no improvement) allow increasing of o and, therefore, a more detailed search.

By setting $r_{\max}$ or b to higher values, the user has a possibility to let the algorithm perform a more thorough search with better chances to find the optimal solution, of course at the expense of longer computation time. The setting of these two parameters is not so critical as setting the generalization level in classical GSFS(o), respectively GSBS(o) and Plus-l-Minus-r. Increasing $r_{\max}$ or b does not lead to a different search, but to a more detailed search.

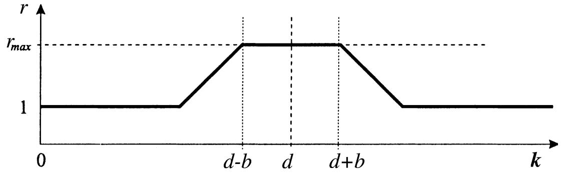

Fig. 2. The meaning of the user parameters $r_{\max}$ and b for adjusting the adaptive generalization.

Remark. Every AFS algorithm run includes steps indentical with CFS ones. Also for this reason we expect the AFS algorithm to find equal or better solutions than CFS. The computer time needed for AFS is expected to be substantially longer (due to the generalization) than for CFS. However, if we constructed the generalized floating search in the simple way, it would consume incomparably more time.

CFS were found to occasionally prefer worse working subsets in the course of search. [1] To describe that case, let us remind the principle of CFS (specifically SFFS) first:

1. Add the *most significant feature* to the current subset of size k. Let $k = k + 1$.
2. Conditionally remove the *least significant feature* from the current subset.
3. If the current subset is the best subset of size $k - 1$ found so far, let $k = k - 1$ and go to step 2. Else return the conditionally removed feature and go to step 1.

Note that backward steps are conditional. Only backward steps bringing improvement are allowed. On the other hand, forward steps cannot be conditional. If they were, the algorithm could theoretically fall into an infinite cycle (repeated by conditional adding and removing a feature). Because of their unconditionality, the forward steps can lead to finding a subset which is worse than the best one of a given dimension found so far. A less promising "search branch" is thus uselessly followed.

Removal of this problem is simple. If the forward step found a subset worse than the best one known so-far the current one is forgotten and the so-far best one becomes the current one. Note that this "violent" swapping of current subset cannot lead to infinite cycling, as finding of a worse subset by the forward step must have been preceded by finding a better subset in some lower dimension.

Now, having defined the notion and discussed the included principles we can describe the ASFFS and ASBFS algorithms.

4. ASFFS algorithm

The adaptive sequential forward floating search (ASFFS) is basically a "bottom up" procedure.

The algorithm is initialized by setting $k = 0$ and $X_0 = \emptyset$. In order to keep the algorithm description traceable, we did not include all the steps needed to ensure its proper functioning, especially when the current dimension gets near to 0 or D. Such steps serve to avoid the algorithm running outside the meaningful dimension boundaries.

Suppose the so-far best values of criterion function $J(X_i)$ are stored as $J_i^{\max}$ for all $i = 1, 2, \ldots, D$. The corresponding so-far best feature subsets X_i are also stored. Initial values of $J_i^{\max}$ for all $i = 1, 2, \ldots, D$ should be set to lowest possible value. Furthermore, suppose k is the size of the current subset.

A. Forward phase

Each phase begins with adaptive setting of r: *If* $|k - d| < b$, *let* $r = r_{\max}$. *Else if* $|k - d| < r_{\max} + b$, *let* $r = r_{\max} + b - |k - d|$. *Else let* $r = 1$.

Step 1. Let $o = 1$.

Step 2 (*Conditional inclusion*). Using the basic GSFS(o) method, select the *most significant* o*-tuple* U_o^m from the set of available measurements $Y \setminus X_k$ *with respect to the set* X_k, then add it to X_k to form feature set X_{k+o}.

Step 3. If $J(X_{k+o}) > J_{k+o}^{\max}$, let $J_{k+o}^{\max} = J(X_{k+o})$, let $k = k + o$ and go to step 6. (*The so-far best subset of size* $k + o$ *was found.*)

Step 4 (*Conditional increase of generalization step*). If $o < r$, let $o = o + 1$ and go to step 2. (*The conditionally included features are removed.*)

Step 5 (*None of the subsets tested in the forward phase were better than the so-far best ones*). Forget the current subset X_k. Let $k = k + 1$. Now consider the so-far best subset of size k to be the current subset X_k.

Step 6 (*Testing the terminating condition*). If $k \geqslant d + \Delta$, stop the algorithm.

B. Backward phase

Each phase begins with adaptive setting of r: *If* $|k - d| < b$, *let* $r = r_{\max}$. *Else if* $|k - d| < r_{\max} + b$, *let* $r = r_{\max} + b - |k - d|$. *Else let* $r = 1$.

Step 7. Let $o = 1$.

[1] We are grateful for the critical remarks by our colleague Dr. R.P.W. Duin from the Delft University of Technology.

Step 8 (*Conditional exclusion*). Using the basic GSBS(o) method, select the *least significant o-tuple* T_o^l *in the set* X_k, then remove it from X_k to form feature set X_{k-o}.

Step 9. If $J(X_{k-o}) > J_{k-o}^{\max}$, let $J_{k-o}^{\max} = J(X_{k-o})$, let $k = k - o$ and go back to the beginning of **Backward Phase**. (*The so-far best subset of size* $k - o$ *was found.*)

Step 10 (*Conditional increase of generalization step*). If $o < r$, let $o = o + 1$ and go to step 8. (*The conditionally excluded features are returned.*)

Step 11 (*None of the subsets tested in the backward phase were better than the so-far best ones.*) Go to **Forward phase**.

End. {*ASFFS*}

A simplified flowchart of the ASFFS algorithm is given in Fig. 3. The terminating condition $k = d + \Delta$ in the flowchart means that in order to fully utilize the potential of the search, we should not stop the algorithm immediately after it reaches for the first time the dimensionality d. By leaving it to float up and down a bit further, the potential of the algorithm is better utilized and a subset of dimensionality d outperforming the first one is usually found. In practice we can let the algorithm either go up to the original dimensionality D, or, if D is too large, then the value of Δ can be determined heuristically (e.g., according to the value of the maximum number of backtracking steps prior to reaching d for the first time).

5. ASBFS algorithm

The algorithm is initialized in the same way as ASFFS, except $k = D$ and $X_D = Y$. The ASBFS (adaptive sequential backward floating search) is the "top down" counterpart of the ASFFS procedure. Since it is analogous to the forward one, due to the lack of space it is not described here.

6. Experimental results

The performance of adaptive floating search has been compared with that of "classical" floating search on a number of real data. Here we just present the results of ASFFS and SFFS on two sets of data (as both concern two-class problems, in both the cases the Bhattacharyya distance was used as the criterion function; a PC with Pentium II 350 was used):

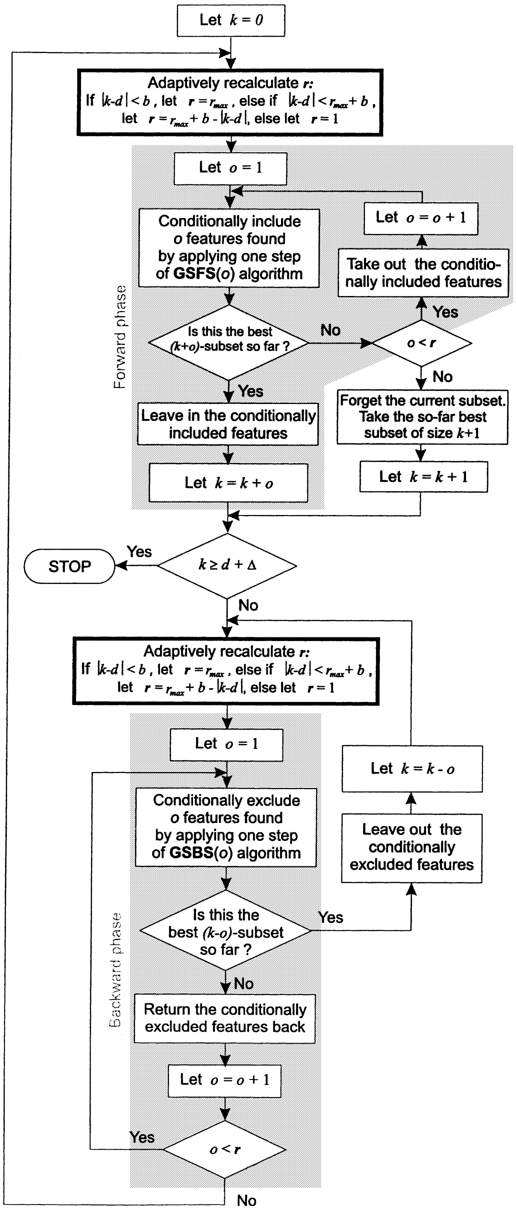
Let k = 0
Adaptively recalculate r:
If |k-d| < b, let r = r_max, else if |k-d| < r_max + b,
let r = r_max + b - |k-d|, else let r = 1
Forward phase
Let o = 1
Conditionally include o features found by applying one step of GSFS(o) algorithm
Let o = o + 1
Take out the conditionally included features
Is this the best (k+o)-subset so far ?
No
Yes
o < r
Yes
No
Leave in the conditionally included features
Forget the current subset. Take the so-far best subset of size k+1
Let k = k + o
Let k = k + 1
STOP
Yes
k ≥ d + Δ
No
Adaptively recalculate r:
If |k-d| < b, let r = r_max, else if |k-d| < r_max + b,
let r = r_max + b - |k-d|, else let r = 1
Backward phase
Let o = 1
Let k = k - o
Conditionally exclude o features found by applying one step of GSBS(o) algorithm
Leave out the conditionally excluded features
Is this the best (k-o)-subset so far ?
Yes
No
Return the conditionally excluded features back
Let o = o + 1
Yes
o < r
No

Fig. 3. Simplified flow chart of ASFFS algorithm.

1. 65-dimensional mammogram data – the dataset was obtained from the Pattern Recognition and

Image Modeling Laboratory (PRIM lab.) at University of California, Irvine.

2. 60-dimensional sonar data – the dataset was obtained from the Machine Learning Database at University of California, Irvine.

From the results we can see that ASFFS yielded better results than classical SFFS. Alhough in these examples the improvement may seem to be marginal, we have to be aware of the fact that finding a different feature subset with only a marginal increase in the criterion value can cause a better performance of the classifier which may prove to be crucial in certain applications.

Furthermore, more essential than the absolute value of improvement is the fact that the adaptive search is capable of finding a solution closer to the optimal one (of course at the expense of longer computation time as documented, e.g., on Fig. 4 for 25 selected features, where ASFFS found a better subset than SFFS, due to a more thorough search) (see Fig. 5).

7. Conclusion

Two new methods of adaptive floating search have been presented. Owing to a more thorough search than classical floating search, they have a potential of finding a solution even closer to the optimal one. The trade-off between the quality of solution and the computational time can be controlled by user's setting of certain parameters.

For further reading, see (Devijver and Kittler, 1982; Siedlecki and Sklansky, 1988; Ferri et al., 1994).

Discussion

Gimel'farb: I have two questions. The first question is: How do you avoid the threat of local minima of the criterion function in the search? Because in feature selection, the feature that we add depends on the feature we start with, and this may heavily influence your results. The second question is: How can you explain such non-linear time behaviour?

Somol: Let me answer the second question first. I think that till a certain point, this method does the same kind of search as classical floating search, meaning that it adds or removes only single features. And every time it tries to find a better subset, increasing the criterion function. But at some point it starts using a generalisation step, meaning that it tries groups of two, three or more features at a time. So eventually, the depth of search may reach the generalisation limit, which of course would be accompanied by a sharp increase of the time. As for the time behaviour, the simple reason is that the algorithm is heuristic. However, one possible explanation is in the fact that the generalisation limit is a special function of the dimensionality. This may be the cause that we get different generalisation limits for the same stages

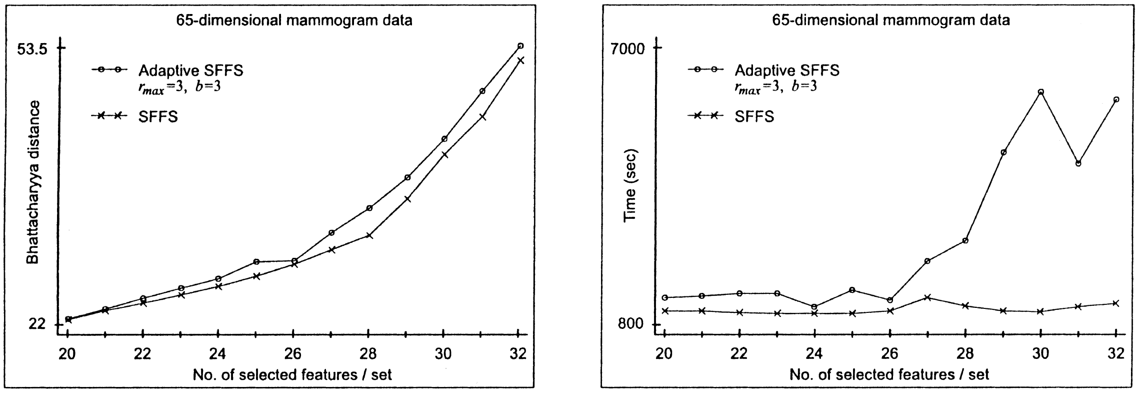

Fig. 4. Results of forward algorithms on mammogram data.

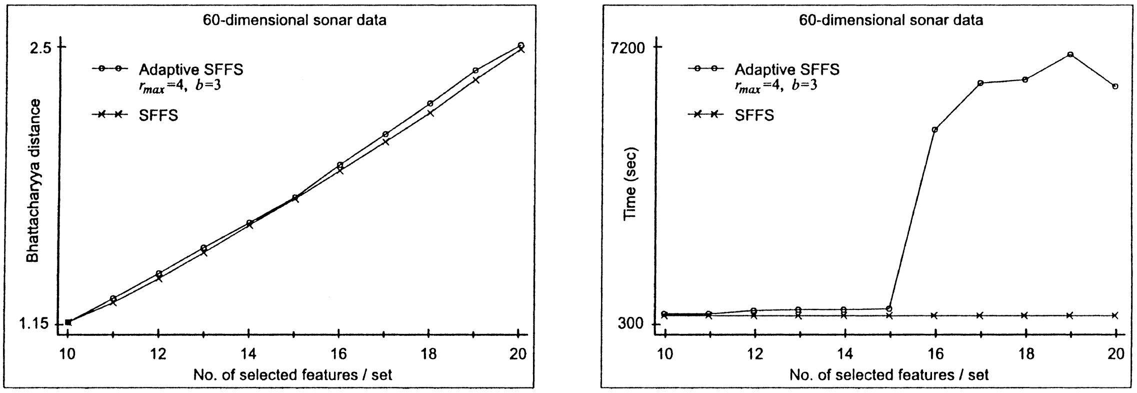

Fig. 5. Results of forward algorithms on sonar data.

of the algorithm, when searching subsets of different cardinalities. And for the first question: I would ask my co-author.

Pudil: First, I would like to further comment on the time behaviour, which I would call non-monotonic, rather than non-linear. Obviously, this is data-dependent; with other data, the behaviour may be different and such jumps may happen not at all. At a certain moment, the algorithm switches to deeper search, because it finds that the generalised versions start to find better solutions. But that is at the expense of longer computational time because of the deeper search. And as far as the first question is concerned, obviously as we all know, the sub-optimal methods are heuristic. The method presented here is only an improvement of previous versions, but it cannot guarantee to find the real optimum. However, in most cases when we compared it, for instance with Branch and Bound, it yielded practically always the same solution.

Egmont-Petersen: Have you considered using assessment criteria other than the statistical distance measure you used? With your criterion, you do not need a classifier. However, if you would use, for instance, the error rate, you would need to train a lot of classifiers.

Somol: I have used various distance measures, not only the Bhattacharrya distance, but also the Mahalobis distance, and the behaviour of the algorithm was similar.

Pudil: For classical floating search, we used the apparent error rate, because it does not depend on monotonic criteria like the Branch and Bound method for example. So there we used the error rate.

Acknowledgements

The authors greatly acknowledge the support by research grants from the Czech Grant Agency No.402/97/1242 and the Czech Ministry of Education Nos. VS96063, ME187/98 and Aktion 23b20.

References

Devijver, P., Kittler, J., 1982. Pattern Recognition: A Statistical Approach. Prentice-Hall, New York.

Ferri, F.J., Pudil, P., Hatef, M., Kittler, J., 1994. Comparative study of techniques for large-scale feature selection. In: Gelsema, E.S., Kanal, L.N. (Eds.), Pattern Recognition in Practice IV. Elsevier, Amsterdam, pp. 403–413.

Jain, A., Zongker, D., 1997. Feature selection: Evaluation, application and small sample performance. IEEE Transactions on PAMI 19, 153–158.

Pudil, P., Novovičová, J., Bláha, S., 1991. Statistical approach to pattern recognition: theory and practical solution by means of PREDITAS System. Kybernetika 27 (Supplement), 1–78.

Pudil, P., Novovičová, J., Kittler, J., 1994. Floating search methods in feature selection. Pattern Recognition Letters 15 (11), 1119–1125.

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