Comparison of Classifier-Specific Feature Selection Algorithms

Mineichi Kudo[1], Petr Somol[2], Pavel Pudil[2], Masaru Shimbo[1], Jack Sklansky[3]

[1] Division of Systems and Information Engineering
Graduate School of Engineering
Hokkaido University, Sapporo 060-8628, Japan.
mine@main.eng.hokudai.ac.jp
[2] Department of Pattern Recognition
Inst. of Information Theory and Automation
Academy of Science of the Czech Republic
Pod vodarenskou vezi 4, 182 08 Prague, Czech Republic
[3] Department of Electrical Engineering
University of California, Irvine, California 92687, USA

Abstract. The performance and speed of three classifier-specific feature selection algorithms, the sequential forward (backward) floating search (SFFS (SBFS)) algorithm, the ASFFS (ASBFS) algorithm (its adaptive version), and the genetic algorithm (GA) for large-scale problems are compared. The experimental results showed that 1) ASFFS (ASBFS) has better performance than does SFFS (SBFS) but requires much computation time, 2) much training in GA with a larger number of generations or with a larger population size, or both, is effective, 3) the performance of SFFS (SBFS) is comparable to that of GA with less training, and the performance of ASFFS (ASBFS) is comparable to that of GA with much training, but in terms of speed GA is better than ASFFS (ASBFS) for large-scale problems.

1 Introduction

Many algorithms for feature selection have been proposed [1,2,3,4,5,6,7]. These algorithms are divided into two categories: in one group [1,2,4,5] the goodness of a feature subset is measured by the value of a given criterion function, and in the other group [3,6,7] the goodness is measured by their own criteria. Algorithms belonging to the former group are called "classifier-specific feature selection algorithms" because their criterion functions are usually based on a correct recognition rate by a certain classifier. This is useful when we know in advance which classifier will be used after feature selection. Algorithms of the latter group are called "classifier-independent feature selection algorithms" because their criterion functions are based on an approximation of class-conditional probability density functions. This means that any classifier-independent feature selection algorithm should not depend on particular classifiers to be used but depend on the ideal "Bayes classifier." This approach is useful when we do not know which

F.J. Ferri et al. (Eds.): SSPR&SPR 2000, LNCS 1876, pp. 677–686, 2000.
© Springer-Verlag Berlin Heidelberg 2000

classifiers will be used or when we wish to avoid over-fitting of the criterion function depending on a specific classifier.

In this study, we compare the three promising algorithms for classifier-specific feature selection. One is the genetic algorithm, hereafter called GA, which is an effective algorithm for finding sub-optimal solutions in optimization problems. GAs have been shown to be effective for feature selection. The second algorithm is the sequential forward (backward) floating search method, hereafter called SFFS (SBFS), which is a genelarized sequential search method and has also been widely used for feature selection. The third algorithm is a recently developed adaptive version of the SFFS (SBFS), hereafter called ASFFS (ASBFS), in which the local greedy search is replaced by a combinational search. The aim of this study is to examine how these algorithms work in large-scale practical problems in which the number of features is more than 50.

2 Comparative Studies

There have been some comparative studies of classifier-specific feature selection algorithms [2,8,9].

Ferri *et al.* [2] concluded that GA and SFFS are comparable in performance, but as the dimensionality increases the result of GA becomes worse than that of SFFS. Jain and Zongker [8] compared GA with SFFS for Kittler's artificial data and reported a tendency of premature convergence when GA was used. On the other hand, Kudo and Sklansky [9] reported that for small-scale and medium-scale problems SFFS or SBFS is better than GA, and GA is better than SFFS or SBFS for large-scale problems. In addition, as the dimensionality increases, GA becomes faster than SFFS and SBFS. They concluded that GA is better than SFFS and SBFS in the following two points: 1) GA is controllable in terms of execution time because it can be terminated whenever we want by limiting the number of generations, and 2) the results of GA can be improved with more trails and with more training. In this paper, we present a comparison of the performance and speed of a sophisticated version of SFFS (SBFS) and GA with more training.

3 Algorithms

Here, we find a feature subset X_d of size $d(< D)$ from the original feature set Y of size D.

3.1 SFFS(SFBS) Algorithm

Among many reported sequential search algorithms, the best in terms of compromise between speed and performance are the SFFS (Sequential Forward Floating Search) method and the SBFS method (its backward version), proposed in 1994 by Pudil *et al.* [4]. The number of forward (adding) / backward (removing) steps is determined dynamically during the method's run so as to maximize the criterion function (Fig. 1).

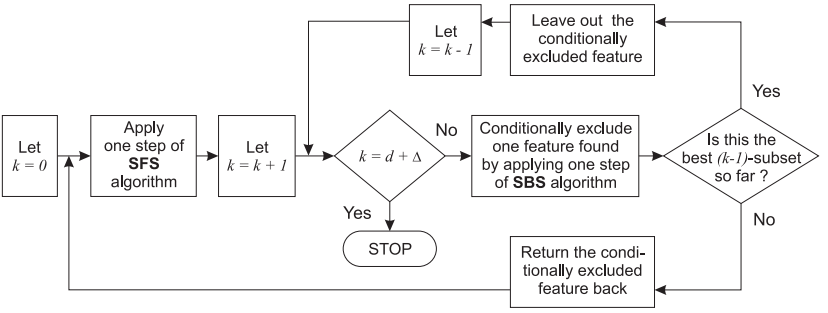

Fig. 1. Simplified flow chart of SFFS algorithm

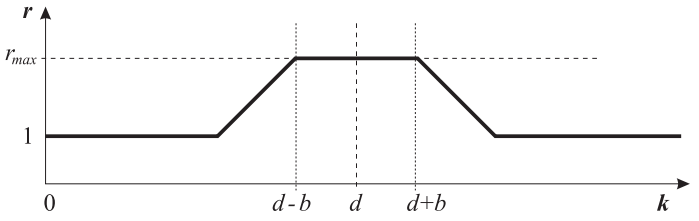

Fig. 2. The meaning of the user parameters r_{max} and b for adjusting the adaptive generalization

3.2 ASFFS (ASBFS) Algorithm [10]

The ASFFS (Adaptive Sequential Forward Floating Search) and ASBFS (its backward version) are sophisticated versions of SFFS and SBFS, respectively.

In addition to the floating search in backtrack, ASFFS enables a more sophisticated search in the local forward search (SFS step in Fig. 1) in SFFS; that is, the method uses a combinational search of o features, GSFS(o), instead of SFS (Sequential Forward Search). Here, the parameter o is upper-bounded by r, and r is upper-bounded by r_{max}, which is arranged to be large near the desired number of features d and be small far from d with an additional parameter b (Fig. 2). The simplified flowchart of the ASFFS algorithm is shown 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 back a bit further, the potential of the algorithm is better utilized and a subset of size 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). ASBFS is initialized in the same way as ASFFS, except that $k = D$ and $X_D = Y$. The ASBFS (Adaptive Sequential Backward Floating Search) is the "top-down" counterpart to the ASFFS procedure.

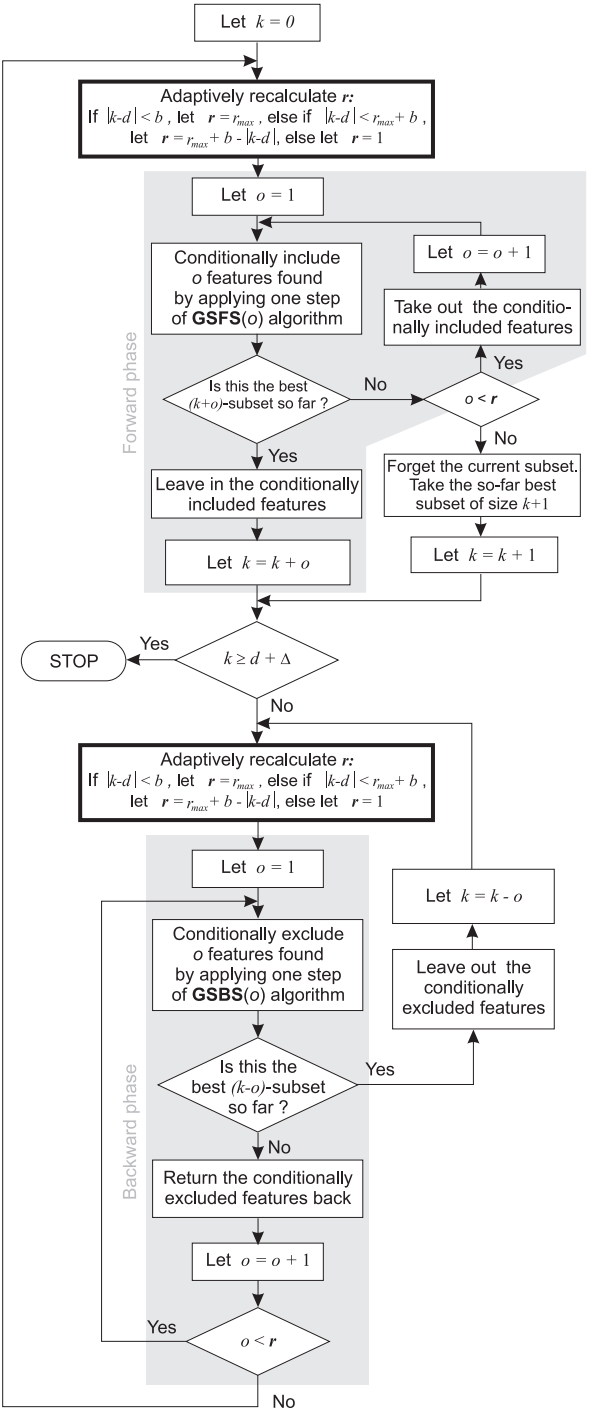

Fig. 3. Simplified flowchart of ASFFS algorithm

Genetic Algorithm [11]

Step 0 (Initialization)
$N \leftarrow$ { Population size}
$\mathcal{P} \leftarrow$ { Initial population with N subsets of Y}
$p_c \leftarrow$ { Crossover probability}
$p_m \leftarrow$ { Mutation probability}
$T \leftarrow$ { Maximum number of generations}
$k \leftarrow 0$
Step 1 (Evolution)
Evaluation of fitness of $\mathcal{P}$
`while (`$k < T$ `and` $\mathcal{P}$ does not converge`) do`
  Breeder Selection
  Crossover with p_c
  Mutation with p_m
  Evaluation of fitness of $\mathcal{P}$
  Replication
  Dispersal
  $k \leftarrow k + 1$

Fig. 4. Simplified flowchart of GA

3.3 Genetic Algorithm

Many studies has been carried out on GAs for feature selection (for example, see [11,12]). In a GA, a feature subset is represented by a binary string with length D, called a *chromosome*, with a zero or one in position i denoting the absence or presence of feature i. Each chromosome is evaluated in its fitness through an optimization function in order to survive to the next generation. A population of chromosomes is maintained and evolved by two operators of crossover and mutation. This algorithm can be regarded as a parallel and randomized algorithm. The simplified flowchart of GA is shown in Fig. 4. GA has four main parameters to be set: the population size N, the maximum number of generations T, the probability of crossover p_c, and the probability of mutations p_m. Based on the results of a study by Kudo and Sklansky [9], we use $N = 2D, T = 50$ and $(p_c, p_m) = (0.8, 0.1)$. In addition, we use $T = 500$ and $N = 20D$ for giving much training to the GA. Two types of initial population of chromosomes are given: 1) Type 1, denoted by $\{1, D-1\}$, in which there are $2D$ feature subsets consisting of all 1-feature subsets and all $(D-1)$-feature subsets, and we add $(N-2D)$ subsets from 2-feature subsets and $(D-2)$-feature subsets; and 2) Type 2, denoted by $[m, M]$ in which N chromosomes are randomly chosen for which the number of features is in this range.

4 Experiments

As the criterion function, we used the leave-one-out correct recognition rate with the 1-NN method. We used $r_{max} = 3$ and $b = 3$ for ASFFS and ASBFS. GAs were carried out in a mode to find the maximum criterion value (corresponding to the objective type O_C in the literature [9]) and repeated three times in each set of parameters.

4.1 Experiment Using Mammogram Data

We tested a mammogram database [9]. The database is a collection of 86 mammograms from 74 cases. Chosen 65 features are 18 features characterizing calcification (number, shape, size, etc.) and 47 texture features (histogram statistics, Gabor wavelet response, edge intensity, etc.). There are two classes of benign and malignant (57 and 29 samples, respectively). Six sets of parameters used in GA are: (N, T, initial population type, p_c, p_m)= $(130, 50, \{1, 64\}, 0.8, 0.1)$, $(1300, 50, \{1, 64\}, 0.8, 0.1)$, $(130, 500, \{1, 64\}, 0.8, 0.1)$, $(130, 50, [10, 14], 0.8, 0.1)$, $(1300, 50, [10, 14], 0.8, 0.1)$, $(130, 500, [10, 14], 0.8, 0.1)$. The results are shown in Figs. 5- 7.

4.2 Experiment Using Sonar Data

Next, we tested the algorithms using sonar data taken from database [13]. The task is to discriminate between sonar signals bounced off a metal cylinder and those bounced off a roughly cylindrical rock using 60 features of which each describes the energy within a particular frequency band, integrated over a certain period of time. The database consists of 111 patterns obtained by bouncing sonar signals off a metal cylinder at various angles and under various conditions and 97 patterns obtained from rocks under similar conditions. As the criterion function, we used the leave-one-out correct recognition rate with the 1-NN method. We used $r_{max} = 3$ and $b = 3$ for ASFFS and ASBFS. Six sets of parameters used in GA are: (N, T, initial population type, p_c, p_m)= $(120, 50, \{1, 59\}, 0.8, 0.1)$, $(1200, 50, \{1, 59\}, 0.8, 0.1)$, $(120, 500, \{1, 59\}, 0.8, 0.1)$, $(120, 50, [18, 22], 0.8, 0.1)$, $(1200, 50, [18, 22], 0.8, 0.1)$, $(120, 500, [18, 22], 0.8, 0.1)$. The results are shown in Figs. 8- 10.

5 Discussion and Conclusion

5.1 SFFS (SBFS) and ASFFS (ASBFS)

In both experiments, we observed the following:

1. In performance, ASFFS (ASBFS) was superior to SFFS (SBFS) overall. However, in some local problems of finding a best feature subset of a certain size, SFFS (SBFS) was sometimes better than ASFFS (SBFS). This means the local optimization strategy does not always give the global maximum.

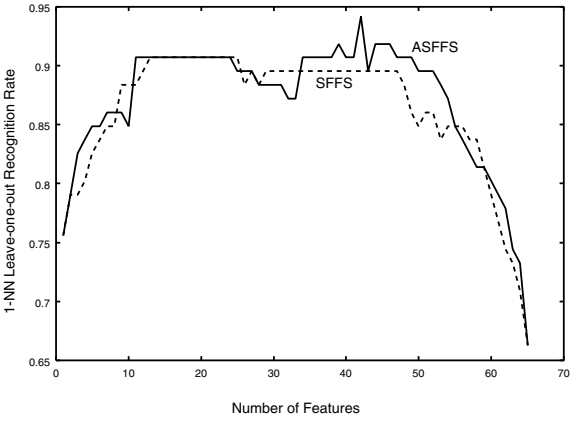

Fig. 5. SFFS vs. ASFFS for mammogram data.

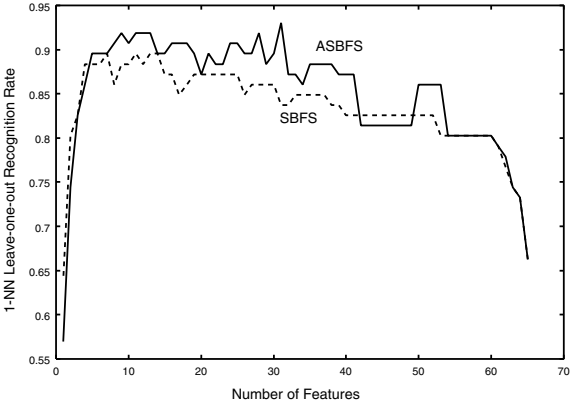

Fig. 6. SBFS vs. ASBFS for mammogram data.

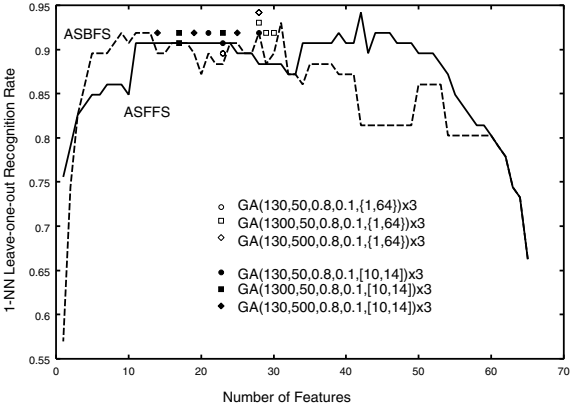

Fig. 7. ASF(B)FS vs. GA for mammogram data.

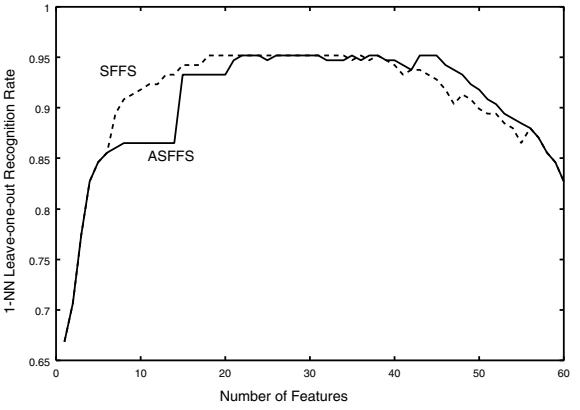

Fig. 8. SFFS vs. ASFFS for sonar data.

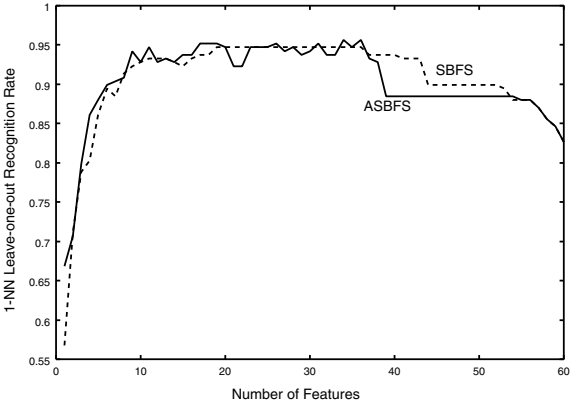

Fig. 9. SBFS vs. ASBFS for sonar data.

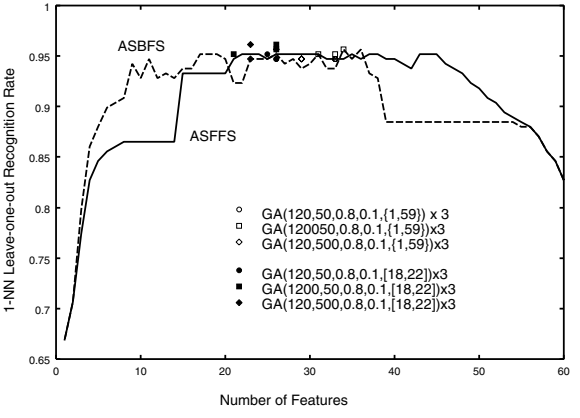

Fig. 10. ASF(B)FS vs. GA for sonar data.

2. ASFFS took about one-thousand times longer than SFFS. Their average evaluation numbers were 164339 and 175, respectively. This ratio is almost the same for ASBFS and SBFS. This difference comes from the fact that ASFFS (ASBFS) actually belongs to the class of generalized algorithms which are obviously more time consuming.

5.2 GA with More Training

For the effectiveness of more training with a larger number of generations or with a larger size of population, we see from Figs. 7 and 10 that

1. An increase in the generation number T (open and closed squares) or an increase in the population size N (open and closed diamonds) leads to better solutions compared with the original settings (open and closed circles). The effectiveness of both enhancements are almost the same.
2. The results of GA with less training are comparable to those of SFFS (SBFS) and are a little inferior to those of ASFFS (ASBFS).

5.3 ASFFS (ASBFS) and GA

From Figs. 7 and 10, we see that

1. In terms of ability to find the maximum value, GA with more training (ten-times larger generation number or population size) and ASFFS (ASBFS) are almost the same. GA requires a few trails to find better solutions but sometimes finds better solutions than those by ASFFS (ASBFS).
2. When ASFFS (ASBFS) is carried out in such a way that it finds feature subsets of every size, it is very time-consuming. This is crucial difference from SFFS (SBFS). This is because ASFFS (ASBFS) is tuned to have more flexibility near a desired number of features d and this flexibility is different depending on the value of d, thus we cannot carry out ASFFS (ASBFS) sequentially in such a way that after finding a solution at size d it finds another solution at size $d + 1$.
3. In terms of speed, GA is faster than is ASFFS (ASBFS). GA required at most 67000 evaluations, while ASFFS (ASBFS) needed about three times that number. However, since the time taken by ASFFS (ASBFS) strongly depends on the size D of the problem and the desired number of features d, for small- and medium-scale problems ASFFS (ASBFS) would be more effective than GA, as was shown in [9].

Based on the results, our recommendations are as follows. If the main priority is time, the user should use SFFS or SBFS for small- and medium-scale problems and GA with less training for large-scale problems. If he or she really wants to achieve the best possible results (of course at the expense of the computational time), ASFFS or ASBFS should be used for small- and medium-scale problems, and GA with more training (for example, ten times generation number or population size) should be used for large-scale problems.

References

1. Kittler, J.: Feature set search algorithms. In: Chen, C. H. (ed.): Pattern Recognition and Signal Processing, Sijthoff and Noordhoff, Alphen aan den Rijn, Netherlands (1978) 41–60
2. Ferri, F. J., Pudil, P., Hatef, M. and Kittler, J.: Comparative study of techniques for large-scale feature selection. In: Gelsema, E. S. and Kanal, L. N. (eds.): Pattern Recognition in Practice IV, Elsevier Science B. V. (1994) 403–413
3. Kudo, M. and Shimbo, M.: Feature selection based on the structural indices of categories. Pattern Recognition **26** (1993) 891–901
4. Pudil, P., Novovičová, J. and Kittler, J.: Floating search methods in feature selection. Pattern Recognition Letters **15** (1994) 1119–1125
5. Pudil, P., Ferri, F. J., Novovičová, J. and Kittler, J.: Floating search methods for feature selection with nonmonotonic criterion functions. In: 12th International Conference on Pattern Recognition (1994) 279–283
6. Novovičová, J., Pudil, P. and Kittler, J.: Divergence based feature selection for mulimodal class densities. IEEE Transactions on Pattern Analysis and Machine Intelligence **18** (1996) 218–223
7. Holz, H. J. and Loew, M. H.: Relative feature importance: A classifier-independent approach to feature selection. In: Gelsema, E. S. and Kanal, L. N. (eds.), Pattern Recognition in Practice IV, Amsterdam: Elsevier (1994) 473–487
8. Jain, A. and Zongkerm, D.: Feature selection: Evaluation, application, and small sample performance. IEEE Trans. Pattern Anal. Machine Intell **19** (1997) 153–157
9. Kudo, M. and Sklansky, J.: Comparison of algorithms that select features for pattern classifiers. Pattern Recognition, **33** (2000) 25–41
10. Somol, P., Pudil, P., Novovičová, J. and Paclík, P.: Adaptive floating search methods in feature selection. Pattern Recognition Letters **20** (1999) 1157–1163
11. Vriesenga, M. R.: Genetic Selection and Neureal Modeling for Designing Pattern Classifier. Doctor thesis, University of California, Irvine (1995)
12. Siedlecki, W. and Sklansky, J.: A note on genetic algorithms for large-scale feature selection. Pattern Recognition Letters **10** (1989) 335–347
13. Murphy, P. M. and Aha, D. W.: UCI Repository of machine learning databases [Machine-readable data repository]. University of California, Irivne, Department of Information amd Computation Science (1996)