Scatter Search for the Feature Selection Problem[⋆]

Félix C. García López, Miguel García Torres[⋆⋆], José A. Moreno Pérez, J. Marcos Moreno Vega

Departamento de E.I.O. and Computación
Escuela Técnica Superior de Ingeniería Informática
University of La Laguna. 38271 La Laguna
Santa Cruz de Tenerife, SPAIN
{fcgarcia,mgarciat,jamoreno,jmmoreno}@ull.es

Abstract. The feature selection problem in the field of classification consists of obtaining a subset of variables to optimally realize the task without taking into account the remainder variables. This work presents how the search for this subset is performed using the Scatter Search metaheuristic and is compared with two traditional strategies in the literature: the Forward Sequential Selection (FSS) and the Backward Sequential Selection (BSS). Promising results were obtained. We use the lazy learning strategy together with the nearest neighbour methodology (NN) also known as Instance-Based Learning Algorithm 1 (IB1).

1 Introduction

Instance-Based Learning (IBL) [1] is an inductive learning algorithm that generalizes tasks by storing available training instances in memory. The purpose of the classification problem, given a set of cases or instances of a known class, is to classify other cases of which only the features or variables are known. A similarity measure is needed between cases [11] for which the set of available features is used. However, not all of the variables that describe the instances provide the same quality of information; some of them do not significantly contribute and can even degrade the information.

The purpose of the feature selection problem is to obtain the subset of these features which optimally carry out the classification task. An appropriate feature selection affects efficiency, because is cheaper to measure only a subset of features, and accuracy, which might be improved by removing irrelevant features. This is of special interest in those problems with a large number of cases and features.

These problems are NP-hard [6]. Thus it is impossible to explore the whole solution space. The metaheuristics provide procedures for finding solutions with

[⋆] This work has been partially supported by the project TIC2002-04242-C03-01, 70% of which are FEDER founds

[⋆⋆] The work of the second author has been partially supported by CajaCanarias grant reasonable efficacy and efficiency. The application of evolutive procedures such as Genetic Algorithms [4] are described in the literature, but, to our knowledge, the Scatter Search (SS) has not been applied. In this work we analyse the SS metaheuristic in this problem, comparing its performance with two traditional strategies: the Forward Sequential Selection (FSS) and the Backward Sequential Selection (BSS).

In the next section we introduce the feature selection problem with a formal description of the problem and the most common solution strategies. In section 3 we describe the main elements of the Scatter Search metaheuristic in the application to this problem. The outcomes of the computational experience are shown in section 4 for several data sets and motivate our conclusions, shown in section 5.

2 The Feature Selection Problem

In the Feature Selection Problem there are two general approaches: the filter and the wrapper strategies. The first approach is characterized by the application of a different methodology in the training and test phases. The wrapper applies the same algorithm in the entire process. In the filter approach, the relevant algorithm FOCUS [3] examines the entire subset of features selecting those that minimize the number of features among those classifying correctly all of the training instances. Another relevant algorithm is *Relief*, which is a random algorithm that assigns a weight to each feature based on the two nearest cases.

The second approach consists of two traditional strategies: the FSS strategy and the BSS strategy. FSS starts with an empty subset of features and iteratively adds the best feature as it improves the solution. BSS starts with the whole set of features and iteratively removes the worst feature while it improves the solution. This last strategy provides less efficient procedures [6].

The formal description of the problem is as follows. Let A be a set of n cases (instances or objects) $A = \{a_i : i = 1, ..., n\}$ each one described by d features (or attributes) $\{X_j\}_{j=1...d}$ where each feature can be nominal or numeric. Each case is described by a vector $a_i = (a_{i1}, \ldots, a_{id})$ and a_{ij} corresponds to the value of feature X_j in the case a_i. Moreover, each case a_i belongs to a class labelled as $a_i(c)$. The purpose of the classification problem is to obtain the label of the instances using only a subset of features. Therefore the objective of our problem is to find the subset of features $S = \{X_{i_j} : j = 1 \ldots m\}$, $m \leq d$ that classifies best.

The classification task using the nearest neighbor approach is carried out by assuming the existence of a set of cases with known class labels (training set) and another set of cases with unknown class labels (test set). The information provided by the training set is used to find the optimal set of features $S = \{X_{i_j} : j = 1, ..., m\}$ to classify. To measure the quality of the classifier (subset of features and classification strategy) we consider the accuracy of the classification as the objective function, f. The associated optimization problem is to find the subset $S \subseteq \{X_j : j = 1, ..., d\}$ such that $f(S)$ is maximized.

Several methods can be considered when measuring accuracy of the algorithms when a set of classified cases is given. A method widely used in the literature is the k-fold cross validation that is described as follows. We consider a total set of cases T divided into k parts of the same size, $T_1, ..., T_k$ and k executions of the algorithm are performed. In the i-th execution the set T_i is taken as the test set and the union of the remainder subsets, $\cup_{j \neq i} T_j$, as training set. In [5] and [2] the method known as "5x2vc" is recommended, which consists in applying the cross validation with $k = 2$ for 5 different orderings of the data base. Such a procedure is also used in the training phase. The IB1 methodology is applied which implies classification by the class of the nearest neighbor [1].

The dissimilarity function between cases applied to determine the nearest neighbor is the distance function HEOM (Heterogenous Euclidean-Overlap Metric) [11]. The distance between two cases a_k and a_m, using the subset of features, $S \subseteq \{X_j : j = 1...d\}$, is defined as

$$d_{HEOM}(a_k, a_m) = \sum_{X_j \in S} d_j^2(a_k, a_m)$$

with

$$d_j(a_k, a_m) = \begin{cases} 1 & \text{if } a_{kj} \text{ or } a_{mj} \text{ are unknown} \\ d_s(a_{kj}, a_{mj}) & \text{if } X_j \text{ is nominal} \\ d_r(a_{kj}, a_{mj}) & \text{if } X_j \text{ is numeric} \end{cases}$$

where d_s is the overlap and d_r is the normalized rectilinear distance that is defined by:

$$d_s(a_{kj}, a_{mj}) = \begin{cases} 1 \text{ if } a_{kj} = a_{mj} \\ 0 \text{ otherwise} \end{cases}$$

and

$$d_r(a_{kj}, a_{mj}) = \frac{|a_{kj} - a_{mj}|}{max_j - min_j}$$

where max_j and min_j are the maximum and minimum values of the feature X_j observed in the training set.

The two traditional hill-climbing strategies, FSS and BSS, improve iteratively the selected subset but they differ in the initial set of features. The first strategy starts with an empty set and at each iteration it adds a feature in case of improvement. The second one starts with the whole set of features and at each iteration removes a feature if there is an improvement. The procedure is as follows:

- *Forward Subset Selection (FSS)*
  1. Initialize the set of features $S = \emptyset$.
  2. For each feature $X_j \notin S$, calculate $f(S \cup \{X_j\})$. Let j^* be such that $f(S \cup \{X_{j^*}\}) = \max_j \{S \cup \{X_j\})\}$ and $S^* = S \cup \{X_{j^*}\}$. If $f(S^*) > f(S)$, take $S = S \cup \{X_{j^*}\}$ and repeat step 2. Otherwise, stop.
- *Backward Subset Selection (BSS)*
  1. Initialize the set of features $S = \{X_j : j = 1, ..., d\}$.

2. For each feature $X_j \in S$, compute $f(S \setminus \{X_j\})$. Let j^* be such that $f(S \setminus \{X_{j^*}\}) = \max_j \{S \setminus \{X_j\})\}$ and $S^* = S \setminus \{X_{j^*}\}$. If $f(S^*) > f(S)$, take $S = S \setminus \{X_{j^*}\}$ and repeat step 2. Otherwise, stop.

3 Scatter Search

The Scatter Search [8] is a recent evolutionary strategy that has been applied successfully to a wide range of optimisation problems. It starts with a population of solutions from which a subset of solutions are selected for the reference set $RefSet$ that evolves by intensification and diversification mechanisms. The solutions of this set are combined to generate new solutions to update the reference set. The combination of the solution in the scatter search is guided and not random, unlike genetic algorithms. Another important difference is that the reference set that evolves in the scatter search is smaller in size than the usual populations in evolutive algorithms.

Figure 1 shows pseudocode of the scatter search.

```
procedure Scatter Search
begin
    GeneratePopulation($InitPop$);
    GenerateReferenceSet($RefSet$);
    repeat
        repeat
            SelectSubSet($SubSet$);
            CombinationMethod($SubSet, S$);
            ImprovingMethod($S$, $S^*$);
        until (StoppingCriterion 1);
        UpdateReferenceSet($RefSet$);
    until (StoppingCriterion 2);
end.
```

Fig. 1. Scatter Search Pseudocode

In order to generate the initial population we use a procedure similar to those used in the constructive phase of the GRASP (Greedy Randomize Adaptive Search Procedure) [12].

The elements of the scatter search in the application to feature selection problem are implemented in the following way:

1. *Generation of the population:*
   The initial population $InitPop$ is constructed by generating solutions from the application of the following procedure until the previously fixed size is reached.

   (a) Do $S = \emptyset$.
   (b) For each feature $X_j \in X \setminus S$, calculate the accuracy with which $S_j = S \cup \{X_j\}$ classifies in the training set using the 5x2 cross validation.
   (c) The restricted candidate list, RCL, to the constructive phase of GRASP is constructed by cardinality and it is compounded by the $|RCL|$ best features according to the value $f(S_j)$.
   (d) Select randomly a feature in RCL. Let X_{j^*} be such feature.
   (e) If by adding X_{j^*} to S it improves the accuracy of the partial solution then do $S = S_{j^*}$ and go to step (b). Otherwise, stop.

2. *Generation of the reference set:*
   The reference set, $RefSet$, is generated in two phases. The first phase includes the $\frac{1}{2}|RefSet|$ solutions of the population with the best values of the objective function f. Then the most diverse solutions with respect to the reference set are added, until the $|RefSet|$ size is reached, by using the following procedure in the second phase:
   (a) Let C be the set of all the features that belongs to any solution of the reference set.
   (b) For each solution S of the population that is not in the reference set, calculate the diversity degree $Div(S, C)$.
   (c) Let S^* be the solution with the highest diversity degree.
   (d) Add S^* to the reference set.
   The diversity degree between a solution S and the set of features C is defined as:
   $$Div(S, C) = |(S \cup C) \setminus (S \cap C)|$$
3. *Selection of the subset:*
   All subsets of two solutions of the reference set are selected (i.e., the inner loop of the pseudocode of Figure 1 is repeated for all the subsets with size 2 (stopping criterion 2)). The combination method is applied to each subset and is described in the next subsection. Note that in the first iteration it is necessary to apply the combination method to all the subsets. However, in later iterations it is only necessary to apply this method to the subsets not considered in previous iterations. This method increases efficiency.
4. *Combination Method:*
   Given two solutions S_1 and S_2, the combination method generates two new solutions, $S_1^{'}$ and $S_2^{'}$, as follows:
   (a) Include in $S_1^{'}$ and $S_2^{'}$ the features that are in S_1 and in S_2; i.e., $S_1^{'} = S_2^{'} = S_1 \cap S_2$. Let $C = (S_1 \cup S_2) \setminus (S_1 \cap S_2)$.
   (b) For each feature $X_j \in C$ calculate $f(S_1^{'} \cup \{X_j\})$ and $f(S_2^{'} \cup \{X_j\})$.
   (c) Let j_1^* and j_2^* be features such that $f(S_1^{'} \cup \{X_{j_1^*}\}) = \max_j\{f(S_1^{'} \cup \{X_j\})\}$ and $f(S_2^{'} \cup \{X_{j_2^*}\}) = \max_j\{f(S_2^{'} \cup \{X_j\})\}$ respectively.
      i. If $f(S_1^{'} \cup \{X_{j_1^*}\}) > f(S_1^{'})$ and $f(S_2^{'} \cup \{X_{j_2^*}\}) > f(S_2^{'})$ update the solution of which the feature X_{j^*} has an associated value of the objective function corresponding to $f = \max\{f(S_1^{'} \cup \{X_{j_1^*}\}), f(S_2^{'} \cup \{X_{j_2^*}\})\}$, do $C = C \setminus X_{j^*}$ and go to step (b). If $f_1^*(S_1^{'} \cup \{X_{j_1^*}\}) = f_2^*(S_2^{'} \cup \{X_{j_2^*}\})$, the feature is added to the partial solution with fewer features.

   ii. If only one of the solutions, S_1' or S_2', improves then add the corresponding feature, do $C = C \setminus X_{j^*}$ and go to step (b).
   iii. Otherwise, stop.

5. *Improving method:*
   We initially applied iteratively the FSS and BSS strategies as long as any improvement took place. However it was observed that it only slightly improved the solutions although a high computational effort was required. Therefore we decided to reject this method.
6. *Update of the reference set:*
   The solution that improves the worst solution in the reference set is then added and replaces the worst one. Therefore, the outer loop in the pseudocode of figure 1 is repeated while the reference set is being updated (stopping criterion 2)

4 Computational Experience

Table 1. Characteristics of the Data Bases considered.

Data Base	Name	Id	Features			$\#Instances$	$\#Classes$
			Total	Nom	Num		
Small	$Iris$	Ir	4	0	4	150	3
	$Echocardiogram$	Ec	9	2	7	132	2
	$Glass$	Gl	9	0	9	214	7
	$Bridges$	Br	11	7	4	108	6
	$Wine$	Wi	13	0	13	178	3
	$Heart(Long - Beach - Va)$	HV	13	7	6	200	2
	$Heart(Hungarian)$	HH	13	7	6	294	2
	$Heart(Cleveland)$	HC	13	7	6	303	2
	Zoo	Zo	16	16	0	90	7
	$SoybeanLarge$	SbL	35	29	6	307	19
	$Sonar$	So	60	0	60	208	2
Average	$PimaIndianDiabetes$	Pm	8	0	8	768	2
	$Vowel$	Vw	10	0	10	528	11
	$CreditScreening$	Cx	15	9	6	690	2
	$Anneal$	An	38	29	9	798	5
Artificials	$Monks - 1$	$Mo1$	6	6	0	432	2
	$Monks - 2$	$Mo2$	6	6	0	432	2
	$Monks - 3$	$Mo3$	6	6	0	432	2

For the computational experience, we considered standard databases from the U.C.I. repository [10]. To study the performance of the algorithms FSS, BSS and SS, we studied real and artificial databases independently. We also studied the small and average ones independently as well for real databases. The reason for doing this was that, for artificial data, the relevance of each feature is known, nevertheless this information is unknown for real databases. Furthermore a low number of instances can affect the undesirable overfitting effects.

Table 1 shows the general characteristics of each database used for the experiments. The name is shown in the second column, and its label in the third one. Additional information is given about the total number of features and the number of nominal and numerical features. Finally the number of instances and the number of classes can be seen in the last two columns.

For the scatter search algorithm an initial population whose number depended on the number of features was considered. The number of solutions was fixed to half of the number of features describing the data. We considered $|LRC| = 3$ for the GRASP strategy to generate the initial population. A number smaller than 3 would make the GRASP strategy very similar to an FSS algorithm, and a larger number would be equivalent to a random initialization. The size of the reference set was set to $\frac{1}{2}|Pop|$, where Pop refers to the population. This $RefSet$ was built with the $\frac{1}{2}|RefSet|$ best solution from Pop and with the same number of most diverse solutions.

Table 2 shows the cross-validation results obtained during the training. A 5x2cv was applied to validate the results and average outcomes are shown for each database. Average standard deviation is also given. Scatter Search found the best solution in most real databases.

The average number of features found for each database is given in Table 3. As we can see, the FSS is the algorithm with greater reduction capacity and the lowest reduction is performed by the BSS.

Table 2. Accuracy and its standardad deviation in training.

DB	$\#FS$	FSS		BSS		SS	
Ir	4	96.09	±0.96	96.14	±0.94	96.08	±1.02
Ec	9	64.39	±4.80	65.61	±3.89	65.21	±4.82
Gl	9	69.88	±4.16	69.97	±3.60	70.49	±4.04
Br	11	59.22	±6.24	58.74	±6.84	60.41	±6.24
Wi	13	96.86	±2.07	97.04	±1.54	97.24	±1.56
HV	13	74.56	±2.92	75.56	±2.74	74.70	±3.83
HH	13	82.61	±3.22	82.15	±3.03	82.84	±3.20
HC	13	79.92	±3.93	80.16	±5.08	80.94	±3.42
Zo	16	91.81	±3.15	90.46	±2.41	93.21	±2.68
SbL	35	81.83	±2.61	81.38	±2.27	83.83	±2.10
So	60	85.04	±3.10	81.79	±3.10	90.40	±1.87
Pm	8	70.66	±1.23	70.53	±1.16	71.39	±0.90
Vw	10	82.07	±1.00	82.09	±1.03	82.14	±1.08
Cx	15	84.94	±1.68	83.22	±2.18	85.28	±1.73
An	38	96.00	±1.48	96.21	±0.86	96.34	±0.69
$Mo1$	6	81.13	±15.96	99.53	±0.35	96.22	±7.77
$Mo2$	6	62.25	±4.11	67.95	±1.83	65.03	±3.45
$Mo3$	6	97.55	±1.86	97.37	±2.17	96.27	±3.00

Outcomes during the test are given in Table 4. These outcomes can be compared with those from the base algorithm IB1. If we compare the results we notice that BSS is clearly different from the other algorithms. In small databases FSS performed better than SS but SS performed better in the most complicated problems (a large number of features). In average problems, SS obtained the best outcomes. This difference increased in the most complicated problems. The BSS algorithm performed best in artificial data.

Table 3. Size of the optimal subset of features found and its standard deviation.

DB	$\#FS$	FSS	BSS	SS
Ir	4	1.90 ±1.10	2.80 ±0.92	2.30 ±1.16
Ec	9	3.10 ±1.45	5.10 ±1.37	3.20 ±0.92
Gl	9	5.30 ±1.42	5.70 ±1.34	4.90 ±1.10
Br	11	3.20 ±1.14	7.60 ±1.71	4.40 ±0.70
Wi	13	5.50 ±1.65	9.2 ±1.40	6.20 ±1.48
HV	13	2.90 ±1.29	9.10 ±1.20	3.10 ±2.03
HH	13	4.60 ±1.96	9.40 ±1.96	4.30 ±1.89
HC	13	4.90 ±1.98	9.20 ±1.32	6.10 ±1.29
Zo	16	6.60 ±2.12	13.90 ±1.45	6.60 ±0.70
SbL	35	17.20 ±2.86	29.60 ±1.78	18.60 ±2.07
So	60	7.20 ±1.87	53.70 ±4.67	11.20 ±2.39
Pm	8	3.40 ±1.27	6.00 ±1.25	3.50 ±1.35
Vw	10	8.50 ±1.18	9.30 ±0.95	8.50 ±1.18
Cx	15	2.20 ±0.79	9.30 ±1.34	2.80 ±1.69
An	38	10.00 ±1.41	27.60 ±2.07	10.80 ±1.75
$Mo1$	6	2.10 ±0.88	3.00 ±0.00	3.20 ±0.42
$Mo2$	6	1.80 ±1.69	5.00 ±0.00	2.80 ±1.93
$Mo3$	6	3.00 ±0.00	3.00 ±0.00	3.00 ±0.00

Finally Table 5 shows the average reduction percentage for each algorithm. SS and FSS performed similarly in average problems and FSS has a higher reduction capacity in the rest of the problems. The BSS has the lowest reduction capacity.

5 Conclusions

The information contained in some features is not relevant enough for the classification problem; therefore the initial feature set can be reduced in order to obtain an optimal subset of features.

Scatter search in small databases is affected by overfitting due to the lack of number of instances; however it performs better if the number of features is large enough. In average data the results are similar except that, in problems with a small number of features, FSS and SS are similar although SS is slightly better.

In artificial data the BSS is the best algorithm because it is the most sensitive to the relevance of each feature among the strategies considered.

SS outcomes are promising. A deeper study of its methods is necessary. Future investigation will consider the application of this algorithm to other learning algorithms and the use of different objective functions.

Table 4. Accuracy and its standardad deviation in testing.

DB	IB1	$\#FS$	FSS	BSS	SS
Ir	94.23	4	95.07 ±1.89	94.80 ±2.22	95.07 ±2.44
Ec	47.88	9	54.55 ±4.23	56.06 ±8.48	52.58 ±5.25
Gl	66.82	9	72.90 ±3.18	70.28 ±5.89	72.90 ±3.15
Br	58.70	11	59.44 ±4.49	58.52 ±7.62	59.26 ±5.45
Wi	94.61	13	94.83 ±2.38	95.28 ±2.94	95.06 ±2.61
HV	72.70	13	69.90 ±4.75	69.40 ±3.89	69.20 ±5.43
HH	78.43	13	80.82 ±3.55	78.23 ±2.42	79.66 ±3.84
HC	75.98	13	73.99 ±3.38	74.06 ±1.81	71.56 ±5.14
Zo	93.65	16	90.89 ±5.33	92.68 ±3.49	90.71 ±4.25
SbL	84.88	35	83.91 ±3.55	84.04 ±4.29	84.76 ±3.25
So	83.17	60	76.64 ±3.24	82.79 ±3.22	79.33 ±2.57
Pm	69.70	8	68.39 ±2.00	67.79 ±1.54	68.49 ±2.78
Vw	95.29	10	94.06 ±1.71	94.63 ±1.90	94.12 ±1.66
Cx	81.54	15	83.77 ±2.34	81.51 ±1.57	84.09 ±2.19
An	93.56	38	95.97 ±2.82	96.35 ±1.73	96.48 ±1.76
$Mo1$	78.34	6	81.91 ±15.97	100.0 ±0.00	97.84 ±6.83
$Mo2$	69.02	6	63.46 ±4.47	70.82 ±3.66	65.62 ±6.17
$Mo3$	81.88	6	97.91 ±1.57	98.20 ±0.83	96.82 ±3.08

Table 5. Capacity of reduction of each algorithm.

DB	$\#FS$	FSS	BSS	SS
Small	11.22	68.16%	20.77%	63.83%
Average	30.25	79.67%	56.86%	79.01%
Artificial	6.00	61.67%	38.33%	50.00%
Total	15.83	67.23%	23.33%	62.98%

References

1. D. W. Aha and D. Kibler and M. K. Albert. Instanced-based learning algorithms. Machine Learning, 6(37–66), 1991.
2. Ethem Alpaydin. Combined 5x2cv f-test for comparing supervised classification learning algorithms. IDIAP-RR 04, IDIAP, 1998.
3. J. R. Anderson and M. Matessa. Explorations of an incremental, bayesian algorithm for categorization. Machine Learning, 9(275–308), 1992.
4. J. Bala, K. Dejong, J. Huang, H. Vafaie, and H. Wechsler. Using learning to facilitate the evolution of features for recognizing visual concepts. Evolutionary Computation, 4(3):297–311, 1996.
5. Thomas G. Dietterich. Approximate statistical test for comparing supervised classification learning algorithms. Neural Computation, 10(7):1895–1923, 1998.
6. Ron Kohavi and George H. John. Wrappers for feature subset selection. Artificial Intelligence, 97(1-2):273–324, 1997.
7. K. Kira and L. Rendell. The feature selection problem: Traditional methods and a new algorithm. Tenth National Conference Conference on Artificial Intelligence (AAAI-92), pages 129–134. MIT, 1992.
8. M. Laguna and R. Martí. Scatter Search: Metodology and Implementations in C. Kluwer Academic Press, 2003.
9. M. Mitchell. An Introduction to Genetic Algorithms. Cambridge, MA: MIT Press, 1996.

10. P. M. Murphy and D. W. Aha. Uci repository of machine learning. Technical report, University of California, Department of Information and Computer Science, 1994.
11. D. R. Wilson and T. R. Matinez. Improved heterogeneous distance functions. Journal of Artificial Intelligence Research, 6:1–34, 1997.
12. M. G.C. Resende and C. C. Ribeiro. Greedy Randomized Adaptive Search Procedures. Handbook of Metaheuristics, F. Glover and G. G. Kochenberger (eds.). Kluwer Academic Publishers, 2003.
13. F. García-López and B. Melián B. and J. A. Moreno-Pérez and J. M. Moreno-Vega. Parallelization of the Scatter Search for the p-median problem. Parallel Computing, 29:575–589, 2003.
14. V. Campos and F. Glover and M. Laguna and R. Martí. An Experimental Evaluation of a Scatter Search for the Linear Ordering Problem. Journal of Global Optimization, 21:397–414, 2001.
15. J. P. Hamiez and J. K. Hao. Scatter Search for Graph Coloring Artificial Evolution, 168–179, 2001.