GRASP for Instance Selection in medical data sets

Alfonso Fernández[1], Abraham Duarte[2], Rosa Hernández[3] and Ángel Sánchez[4]

Abstract Medical data sets consist of a huge amount of data organized in instances, where each one contains several attributes. The quality of the models obtained from a database strongly depends on the information previously stored on it. For this reason, these data sets must be preprocessed in order to have fairly information about patients. Data sets are preprocessed reducing the amount of data. For this task, we propose a GRASP algorithm with two different improvement strategies based on Tabu Search and Variable Neighborhood Search. Our procedure is able to widely reduce the original data keeping the most relevant information. Experimental results show how our GRASP is able to outperform the state of the art methods.

1 Introduction

Almost every day, medical staff diagnoses whether a patient has a disease or not in order to apply the pertinent treatment. The diagnosis is based on different analyses/tests performed to the patient and the expertise of the doctors. This process could be eased if medical staff were able to find common patterns with other patients that have suffered the same disease. In this way, if it is compared the medical record of the current patient with other patients previously treated, doctors could infer a diagnosis based of the history. However, it means to deal with massive amounts of information and collections of data. A very active line of research focuses on scaling down data, where the main problem is how to select the relevant data. This task is carried out in the data preprocessing phase in a Knowledge Discovery in Databases (KDD) process. The goal of this area consists of withdrawing relevant information from databases using a systematic and detailed analysis of the data [5]. This information is used in Data Mining, DM, to create models useful for science, engineering or economy [8]. As it is reported in the literature, DM models are very dependent on the quality of the stored data. Therefore, the first phase KDD is the preprocessing the original data whose main target is improving the "quality" of data.
In Data Mining there are several preprocessing techniques. Among them, we can highlight *Data Reduction*, *Data Cleaning*, *Data Integration* and *Data Transformation*. The reader is referred to [1] to find detailed descriptions of these strategies.

[1]Departamento de Ciencias de la Computación, URJC, a1fonso.fernandez@urjc.es
[2]Departamento de Ciencias de la Computación, URJC, abraham.duarte@urjc.es
[3]Departamento de Ciencias de la Computación, URJC, rm.hernandez@alumnos.urjc.es
[4]Departamento de Ciencias de la Computación, URJC, angel.sanchez@urjc.es

In this work, we focus on Data Reduction (DR). It can be achieved in many ways. Specifically, in the literature we can find DR based on selecting features, making the feature-values discrete and selecting instances. This paper is devoted to Instance Selection (IS) as DR mechanism [11]. IS consists of reducing the number of rows in a data set where each row represents an instance. There are several IS strategies. Among them we can highlight sampling, boosting, prototype selection, and active learning. We will study IS from the prototype selection (PS) perspective, called IS-PS.

There are several papers in the literature that have studied this topic. Most of them are based on Evolutionary Strategies. The first relevant paper, presented by Kuncheva [10], is an Evolutionary Algorithm for PS. In [1] is presented PBIL as the first combination of Genetic Algorithm and Competitive Learning designed for searches in binary spaces. Eshelman presented in [3] CHC, considered a reference in the Genetic Algorithm field because it introduces a diversity mechanism to obtain a good balance between intensification and diversification in the search process. Cano et al introduced in [2] a Memetic Algorithm, MA, that solves the scalability problem in prototype selection. Finally, in [6] is presented the most recent work in the context of IS-PS. It is an improved MA, called SSMA, which outperforms previous approaches.

In this work, we have designed a procedure based on several metaheuristics, to preprocess medical data sets. The goal of our procedure consists of reducing the set of original data obtaining a subset of data that fairly represents the original set.

2 Preprocessing in KDD

In general, data sets are arranged on a table, *OriginalTable*, where each row corresponds to an *Instance* and each column to an attribute. Each instance is characterized by a set of attributes and classified in a determined class (according to the values of their corresponding attributes). IS techniques construct a smaller table, *ReducedTable*, selecting the smallest set of instances that enable a given algorithm to predict the class of a query instance with the same (or higher) accuracy as the original set. This reduction improves both space and time complexities of subsequent DM strategies. It is important to remark that removing instances does not necessarily lead to a degradation of the results. This behavior could be explained taking into account that some data with noise or repeated be deleted by removing instances.

The main goal of our proposal consists of constructing the *ReducedTable* $\subset$ *OriginalTable* with the most representative instances and the larger capacity of classifying new instances. In order to classify an instance in the corresponding class, we will use the Nearest Neighborhood (1-NN strategy) as customary. In order to classify a new instance using *ReducedTable*, we compute the distances from this new instance to the rest of instances in *ReducedTable*. Finally, the new instance is assigned to the same class of the nearest neighbor one. As in previous works, we use

the Euclidean distance defined in an *n*-dimensional space, where *n* represents the number of attributes [2].
To determine the quality of the IS-PS technique we define a *fitness* function that combines two values: the classification performance (%*Clas*) and the percentage of reduction (%*Red*). This function is a tradeoff between the ability of *ReducedTable* to classify instances and the reduction in the data performed for IS-PS. In mathematical terms, the *fitness* function *f* is:

$$f = \alpha*(\%Clas) + (1-\alpha)*(\%Red) \qquad (2.1)$$

The 1-NN classifier is used for measuring the classification rate, %*Clas*, and denotes the percentage of correctly classified instances and %*Red*, is defined as:

$$\%Red = 100* (|OriginalTable| - |ReducedTable|)/|OriginalTable| \qquad (2.2)$$

where |*OriginalTable*| is the number of instances in the original table and |*ReducedTable*| is the number of instances in the reduced table. The objective of the proposed algorithm is to maximize the *fitness* function. As a consequence, it is maximized the classification rate and minimized the resulting number of instances. The value of α ranges from 0 to 1. It measures the emphasis given to precision (percentage of classification) and reduction (percentage of reduction). In this work, we set α to 0.5, balancing precision and reduction.

3 GRASP for IS-PS

The GRASP methodology was developed in the late 1980s, and the acronym was coined by Feo and Resende in 1995[4]. Each iteration consists of constructing a trial solution and then applying an improvement procedure to find a local optimum (i.e., the final solution for that iteration). The construction phase is iterative, randomized greedy, and adaptive. In this section we describe our adaptation of the GRASP methodology to IS-PS.

3.1 Constructive algorithm

The construction phase is greedy, randomized and adaptive. It is greedy because the addition of each element is guided by a greedy function. It is randomized because a random selection takes place and the information provided by the greedy function is used in combination with this random element. Note that the randomness in GRASP allows multiple iterations obtaining different solutions. Finally, it is adaptive because the element chosen at any iteration in a construction is a function of those previously chosen (updating relevant information from one construction step to the next).
In the context of IS-PS, the constructive algorithm starts by computing the center of gravity of a class. The center, *s_center*(X), of a set of elements belonging to class $X = \{s_i : i \in I\}$ is defined as:

$$s\_center(X) = \frac{\sum_{i \in I} s_i}{|X|} \qquad (3.1)$$

where *I* represent the set of different attributes. *s_center*(*X*) is “virtual” instance where the value of each attribute is computed as the average of the attributes of every instance that belongs to *X*. In order to have a “real” instance instead of a “virtual” instance, we select from the original table the nearest instance to each virtual instance. Therefore, if the data set has *m* classes, we compute *m* centers of gravity (one for each class), and we initialize *ReducedTable* with these *m* real instances. One time we have this table; we can compute the *fitness* as defined above. Obviously, we will have the largest possible percentage of reduction but the percentage of classification is worst.

To simplify the notation, we call *Sel* as the set of instances in *ReducedTable* and *Unsel* as the set of instances in *OriginalTable – ReducedTable*. *Sel* contains the set of selected instances and *Unsel* contains the set of unselected instances.

The GRASP constructive procedure improves %*Class* by adding new instances to *Sel* one at a time. In order to do so, it is computed the *fitness f(v)*, for each instance $v \in$ *Unsel* if instance *v* were included in *Sel.* Notice that the larger *f*(*v*) the better the improvement. This is the greedy part of the algorithm.

All the instances in *Unsel* with a *fitness* value strictly positive are candidates to be included in *Sel.* We call them as Candidate List (*CL*),

$$CL = \{v \in Unsel \,/\, f(v) > 0\} \qquad (3.2)$$

We define the Restricted Candidate List (*RCL*), as the set of elements with larger *f*(*v*) values. In mathematical terms:

$$RCL = \{v \in CL \,/\, f(v) \geq f_{th}\} \qquad (3.3)$$

where f_{th} is a threshold computed as a percentage β between the maximum, f_{max}, and minimum, f_{min}), values of the instances in *Unsel*:

$$f_{th} = f_{min} + \beta(f_{max} - f_{min}) \qquad (3.4)$$

where $f_{max} = max\ f(v)\ f_{min} = min\ f(v)$ with $v \in CL$. If β =1, the algorithm is completely greedy. On the other hand, if β = 0 the algorithm is purely random. To favor the “diversification” of the procedure, an instance is randomly selected at each iteration to be included in *Sel*. This is the random part of the algorithm. The inclusion of the new instance in *Sel* yields to a modification of the computation of *f*(*v*) and then, to a new *RCL*. This is the adaptive part of the algorithm.

The constructive process is maintained, including instances into *Sel*, until no further improvement in the *fitness* is obtained. Then it stops and returns the constructed solution.

3.2 Local Search

The second phase of our solving method is an improvement procedure. Specifically, we propose a local search, LS, based on removing/adding instances. It means that solutions reachable from the incumbent one are those constructed by removing or adding one instance to *Sel*. Specifically, LS starts by removing instances from *Sel* until no further improvement in the *fitness* value is obtained. Then, the local search resort to add new instances, selecting in each iteration an instance in *Unsel* able to improve the current *fitness*. LS performs insertion movements while the *fitness* value increases. LS keeps removing/ adding instances until no further improvement is possible.

3.3 Tabu Search

Tabu Search, TS is a metaheuristic that guides a local search procedure to explore the solution space beyond local optimality [7]. One of the main components of TS is its use of adaptive memory, which creates more flexible search behavior.
The structure of a neighborhood in TS goes beyond that used in local search by embracing the types of moves used in constructive and destructive processes (where the foundations for such moves are accordingly called constructive neighborhoods and destructive neighborhoods). We can implement memory structures to favor (or avoid) the inclusion of certain elements in the solution previously identified as attractive (or unattractive). Such expanded uses of the neighborhood concept reinforce a fundamental perspective of TS, which is to define neighborhoods in dynamic ways that can include serial or simultaneous consideration of multiple types of moves.
We propose a TS strategy based on LS. In destructive neighborhoods, TS selects the instance x in *Sel* with the lowest contribution, $f(x)$, to the *fitness* function. It means that we would obtain the best possible *fitness* removing x. On the other hand, in constructive neighborhoods, TS selects the instance y in *Unsel* which were able to improve upon the *fitness* value. Every instance involved in a movement becomes tabu for *Tenure* iterations. As it is customary in TS, we permit non-improving moves that deteriorate the objective value. TS stops after *MaxIterations* without improving the best found solution.

3.4 Variable Neighborhood Search

The Variable Neighborhood Search (VNS), proposed by Hansen and Mladenovíc [9], is a metaheuristic whose basic idea is a systematic change of neighborhood within a local search. Each step in VNS has three major phases: neighbor generation, local search and jump. Unlike to other metaheuristics, VNS allows changes of the neighborhood structure during the search. VNS explores increasingly neighborhoods of the current best found solution. The basic idea is to change the neighborhood structure when the local search is trapped on a local minimum.
Let N_k, $k = 1,\ldots, k_{max}$ be a set of predefined neighborhood structures and let $N_k(x)$ be the set of solutions in the kth-order neighborhood of a solution x. Specifically the kth-order neighborhood is defined by all solutions that can be derived from the current one by selecting k instances and transferring each instance from *Sel* to *Unsel* and vice versa.

Our VNS strategy is based on LS, but it generalizes the constructive/destructive neighborhoods. Specifically, starting from a solution *x*, VNS select *k* instances at random (instead of 1) to be added to/removed from *Sel* obtaining a solution *y*. After that, it applies LS. If the new improved solution has a better *fitness* than the original one, then *k* is set to 1 and the search jumps to the new solution. Otherwise, *k* = *k*+1 and VNS tries again to add/remove *k* instances at random. The procedure stops when *k* reaches a maximum value.

5 Experimental Results

All experiments were conducted on a personal computer with a Pentium IV Core 2 Duo 2.27 GHz with 2 GB RAM. We coded all the procedures in Java and the number of iterations of the GRASP algorithm was set to 20.
In order to evaluate the behavior of the algorithms applied in different size data sets, we have carried out a number of experiments increasing complexity and size of data sets. We have selected seven test sets, which cover a wide size range, as we can see in Table 1. They are available at http://archive.ics.uci.edu/ml/.

Table 1 Medical datasets

Name	Instances	Attributes	Classes
Lymphography	148	18	4
Cleveland	303	14	2
Bupa	345	7	2
Wisconsin	683	9	2
Pima	768	8	2
Splice	6435	36	3
Thyroid	7200	21	3

For example, *Cleveland* data set contains information about patients with or without cardiovascular problems. Some of its attributes are the number of cigarettes smoked a day, the age of the patient or if an antecessor of the family of the patient suffered from similar problems. In *Wisconsin*, the data are gathered from the analysis of different samples of lung tissue of patients that could suffer from lung cancer. In this case, the attributes correspond to measurements of cell morphology such as area, radius or perimeter. *Splice* junctions are points on a DNA sequence at which “superfluous” DNA is removed during the process of protein creation in higher organisms. This problem consists of three subtasks: recognizing exon/intron boundaries (referred to as EI sites), recognizing intron/exon boundaries (IE sites) or none of them.
Table 2 reports the average results for the test sets introduced in Table 1. All the results were computed using a 10-fold cross validation scheme showing the average values. In the first column, CPU Time, we report the average running time. In the second column, %*Class*, is represented the average percentage of classification

with its corresponding standard deviation. Finally, in the third column, %*Red*, shows the average percentage of reduction.

Table 2 Results for all the medical datasets

Name	GRASP_TS			GRASP_VNS		
	CPU Time	%Class	%Red	CPU Time	%Class	%Red
Lymphography	6.18	91.67 ± 1.19	90.69 ± 1.35	0.89	90.32 ± 1.09	92.19 ± 0.81
Cleveland	30.9	69.18 ± 1.59	90.42 ± 1.27	7.96	65.74 ± 1.45	93.19 ± 0.83
Bupa	37.8	85.76 ± 1.12	91.40 ± 1.20	5.93	79.97 ±1.46	96.01 ± 0.73
Wisconsin	105.5	98.15 ± 0.28	99.38 ± 0.18	4.75	97.82 ± 0.31	99.61 ± 0.08
Pima	172.2	81.48 ± 1.12	97.11 ± 0.68	45.5	80.93 ± 0.70	98.44 ± 0.21
Splice	12384	92.30 ± 0.46	95.53 ± 0.26	3418	91.05 ± 0.46	96.35 ± 0.26
Thyroid	7695	95.11 ± 0.22	99.50 ± 0.10	3067	94.96 ± 0.12	99.64 ± 0.09

Table 2 shows the merit of the proposed procedures. Our GRASP implementations, GRASP_TS and GRASP_VNS, consistently produce high quality solutions. GRASP_TS marginally improves GRASP_VNS in terms of %Class, while GRASP_VNS improves GRASP_TS in terms of %Red. Summarizing, the behavior of both procedures is quite similar. The robustness of the method fact can be observed in the small value of the standard deviation. Regarding the percentage of reduction, as an example, in the test set Wisconsin, the final *ReducedTable* contains only two instances (one for each class) classifying correctly on average 97.65% of the instances. On the other hand, it is important to remark that our approach is able to reduce the set of instances from thousands (i.e Thyroid) to tens (1%), with a CPU time considerably larger.
Having determined the quality of our algorithm, we compare our GRASP algorithms with the best method identified in previous studies [6]. We employ in each experiment not only the same test sets but also the conditions and evaluation criteria found in the respective papers. Tables 3 shows the *fitness* value for both procedures executed over the whole test set.

Table 3 Results for medium medical datasets

Name	GRASP_TS	GRASP_VNS	SSMA
Lymphography	91.18 ± 0.50	**91.26 ± 0.41**	75.23 ± 1.47
Cleveland	79.80 ± 0.30	79.46 ± 0.45	**80.68 ± 0.82**
Bupa	**88.58 ± 0.38**	87.99 ± 0.54	85.78 ± 1.07
Wisconsin	**98.76 ± 0.14**	98.71 ± 0.15	98.52 ± 0.11
Pima	89.29 ± 0.38	89.68 ± 0.33	**89.77 ± 0.48**
Splice	**93.91 ± 0.14**	93.70 ± 0.14	75.23 ± 1.47
Thyroid	**97.30 ± 0.09**	**97.30 ± 0.07**	97.08 ± 0.11
Avg. Value	91.26 ± 0.28	91.16 ± 0.29	86.04 ± 0.79

The best values of the *fitness* are bolded. The results in Tables 3 indicate that GRASP is capable of finding high quality solutions, defeating to SSMA in 5 (out of 7) test sets. Regarding the average *fitness* GRASP compares favorably to SSMA in both improvement methods (TS and VNS).

6 Conclusions

We have described the development and implementation of a GRASP algorithm for IS–PS problem. We propose a new constructive procedure based on the computation of the center of gravity. We have also described a new improvement method based on two different types of movements: exchanges and add/remove instances. Additionally, we have proposed two advanced improvement strategies based on the TS and VNS methodologies. We are able to produce a method that reaches good quality solutions on previously reported problems. Our algorithm is compared to state-of-the-art methods and the outcome of our experiments seems quite conclusive in regard to the merit of the procedure that we propose.

Acknowledgments

This research has been partially supported by the Ministerio de Ciencia e Innov. (Ref. TIN2008-06890-C02-02), and by the Comunidad de Madrid–URJC (Ref. URJC-CM-2008-CET-3731).

References

[1]S. Baluja. Population-based incremental learning, Carnegie Mellon Univ. Pittsburgh, PA, CMU-CS-94-163 (1994).

[2]J. R. Cano, F. Herrera and M. Lozano, Using Evolutionary Algorithms as Instance Selection for Data Reduction in KDD: An Experimental Study, IEEE Transactions on Evolutionary Computation 7 (2003) 561-575.

[3] L.J. Eshelman, The adaptive search algorithm: How to have safe search when engaging in non-traditional genetic recombination, in *Foundations of Genetic Algorithms-1*, Morgan-Kauffman (1991) 265-283.

[4] T. A. Feo and M. Resende, Greedy adaptive search procedures. Journal of Global Optimization 6 (1995) 109-133.

[5] A.A. Freitas, Data Mining and Knowledge Discovery with Evolutionary Algorithms. Springer (2002).

[6] S. García, J.R. Cano and F. Herrera, A memetic algorithm for evolutionary prototype selection: A scaling up approach, Pattern Recognition 41 (2008) 2693-2709.

[7] F.Glover and M. Laguna. Tabu Search. Kluwer Academic Publishers, Boston (1997).

[8] J. Han and M. Kamber, Data Mining: Concepts and Techniques, Morgan Kaufmann (2006).

[9] P. Hansen and N. Mladenovìc. Variable Neightborhood Search. In: Glover, F., Kochenberger, G. (Eds.), Handbook of Metaheuristics, Kluwer (2003), 145-184.

[10] L.I. Kuncheva. Editing for the k-nearest neighbors rule by a genetic algorithm. Pattern Recognition Letters 16 (1995) 809-814.

[11] H. Liu and H. Motoda. On issues of instance selection. Data Mining and Knowledge Discovery 6 (2002),115-130.