Incorporating Knowledge in Evolutionary Prototype Selection

Salvador García[1], José Ramón Cano[2], and Francisco Herrera[1]

[1] University of Granada, Department of Computer Science and Artificial Intelligence, E.T.S.I. Informática, 18071 Granada, Spain
{salvagl, herrera}@decsai.ugr.es
[2] University of Jaén, Department of Computer Science, 23700 Linares, Jaén, Spain
jrcano@ujaen.es

Abstract. Evolutionary algorithms has been recently used for prototype selection showing good results. An important problem in prototype selection consist in increasing the size of data sets. This problem can be harmful in evolutionary algorithms by deteriorating the convergence and increasing the time complexity. In this paper, we offer a preliminary proposal to solve these drawbacks. We propose an evolutionary algorithm that incorporates knowledge about the prototype selection problem. This study includes a comparison between our proposal and other evolutionary and non-evolutionary prototype selection algorithms. The results show that incorporating knowledge improves the performance of evolutionary algorithms and considerably reduces time execution.

1 Introduction

Most machine learning methods use all examples from the training data set. However, data sets may contain noisy examples, that make the performance worse of these methods, or they may contain great amount of examples, increasing the complexity of computation. This fact is important especially for algorithms such as the k-nearest neighbors (k-NN) [1]. Nearest neighbor classification is one of the most well known classification methods in the literature. In its standard formulation, all training patterns are used as reference patterns for classifying new patterns.

Instance selection (IS) is a data reduction process applied as preprocessing in data sets which are used as inputs for learning algorithms [2]. We consider data as stored in a flat file and described by terms called attributes or features. Each line in the file consists of attribute-values and forms an instance. By selecting instances, we reduce the number of rows in the data set. When we use the selected instances for direct classification with k-NN, then the IS process is called Prototype Selection (PS).

Various approaches were proposed in order to carry out PS process in the literature, see [3] and [4] for review. Evolutionary Algorithms (EAs) have been used to solve the PS problem with promising results [5,6]. These papers show that EAs outperform the non-evolutionary ones obtaining better instance reduction

E. Corchado et al. (Eds.): IDEAL 2006, LNCS 4224, pp. 1358–1366, 2006.

rates and higher classification accuracy. However, the increasing of the size of data is always present in PS. The Scaling Up problem produces excessive storage requirement, increases times complexity and affects to generalization accuracy.

When we use EAs for selecting prototypes (we call it as Evolutionary Prototype Selection (EPS)), we have to add to these drawbacks the ones produced by the chromosomes size associated to the representation of the PS solution. Large chromosomes size increases the storage requirement and time execution and reduces significantly the convergence capabilities of the algorithm. A way of avoid the drawbacks of this problem can be seen in [7], where data sets stratification is used.

In order to improve the capacity of convergence and reduce time execution on EPS, we propose an evolutionary model that incorporates knowledge through the local improvement of chromosomes based on removing prototypes and adapting chromosome evolution. The aim of this paper is to present our proposal model and compare it with others PS algorithms studied in the literature. To address this, we have carried out experiments with increasing complexity and size of data sets.

To achieve this objective, this contribution is set out as follows. Section 2 summarizes the main features of EPS. In Section 3, we explain how to incorporate knowledge in EPS. Section 4, describes the methodology used in the experiments and analyzes the results obtained. Finally, in Section 5, we point out our conclusion.

2 Evolutionary Prototype Selection

EAs [8] are stochastic search methods that mimic the metaphor of natural biological evolution. All EAs rely on the concept of *population* of individuals (representing search points in the space of potential solutions to a given problem), which undergo probabilistic operators such as *mutation*, *selection* and *recombination*. The *fitness* of an individual reflects its objective function value with respect to particular objective function to be optimized. The mutation operator introduces innovation into the population, the recombination operator performs an information exchange between individuals from a population and the selection operator imposes a driving force on the evolution process by preferring better individuals to survive and reproduce.

PS problem can be considered as a search problem in which EAs can be applied. To accomplish this, we take into account two important issues: the specification of the representation of the solutions and the definition of the fitness function.

- *Representation:* Let us assume a data set denoted TR with n instances. The search space associated is constituted by all the subsets of TR. This is accomplished by using a binary representation. A chromosome consists of n genes (one for each instance in TR) with two possible states: 0 and 1. If the gene is 1, its associated instance is included in the subset of TR represented by the chromosome. If it is 0, this does not occur.

– *Fitness Function:* Let S be a subset of instances of TR to evaluate and be coded by a chromosome. We define a fitness function that combines two values: the classification rate ($clas\_rat$) associated with S and the percentage of reduction ($perc\_red$) of instances of S with regards to TR.

$$Fitness(S) = \alpha \cdot clas\_rat + (1 - \alpha) \cdot perc\_red. \quad (1)$$

The 1-NN classifier is used for measuring the classification rate, $clas\_rat$, associated with S. It denotes the percentage of correctly classified objects from TR using only S to find the nearest neighbor. For each object y in S, the nearest neighbor is searched for amongst those in the set $S \setminus \{y\}$. Whereas, $perc\_red$ is defined as

$$perc\_red = 100 \cdot \frac{|TR| - |S|}{|TR|}. \quad (2)$$

The objective of the EAs is to maximize the fitness function defined, i.e., maximize the classification rate and minimize the number of instances obtained.

Considering this issues, four models of EAs have been studied as EPS [6]. The first two are the classical Genetic Algorithm (GA) models [9]; the generational one and the steady-state one. The third one, heterogeneous recombinations and cataclysmic mutation (CHC), is a classical model that introduces different features to obtain a tradeoff between exploration and exploitation [10], and the fourth one, PBIL [11], is a specific EA approach designed for binary spaces.

Our proposal of EA is a steady-state model with the following characteristics:

– The *fitness function* is calculated by the number of instances correctly classified, without obtain the reduction rate.
– Selection mechanism used is binary tournament.
– As genetic operators we use a crossover operator that randomly replace 20% of first parent's bits with second parent's bits and vice versa, and standard mutation of bit representation of chromosomes.
– Our proposal will use a replacement of the worst individuals of the population in all cases.

3 Incorporating Knowledge in Prototype Selection

Incorporation of knowledge can help to improve the behavior of an algorithm for a determined problem. We have designed a Local Search (LS) procedure based on knowledge on the PS problem that will be applied to improve individuals of a population of an EA. LS that incorporates knowledge procedure is an iterative process that tries to enhance the accuracy classification of a chromosome representation by using 1-NN method and to reduce the number of instances selected in a solution.

To achieve this double objective, it considers neighborhood solutions with $m-1$ instances selected, being m equal to the number of instances selected in a current solution (all positions with value 1 in the chromosome). In other words, a neighbor is obtained by changing 1 to 0 in a gene. In this way, the number of instances represented in a chromosome after the optimization always will be less than or equal to the number of instances of the original chromosome.

Now we describe the local search. It has a standard behavior when we improve the fitness, and a strategy for dealing with the problem against premature convergence in the second half of the run. In the following, we describe them:

- *Standard behavior*: It starts from an initial assignment (a recently generated offspring) and iteratively try to improve the current assignment by local changes. If in the neighborhood of the current assignment, a better assignment is found, it replaces the current assignment and it continues from the new one. The selection of a neighbor is made randomly without repetition among all solutions that belongs to the neighborhood. In order to consider an assignment better than the current one, the accuracy of classification must be better than or equal to the previous one, but in this last case, the number of instances selected must be less than current assignment. The procedure stops when there are not solutions considered better than the current one in its neighborhood.
- *Avoiding premature convergence*: When the search process advances, a tendency of the population to premature convergence toward a certain area of the search space takes place. A local optimization promotes this behavior when it considers solutions with better classification accuracy. In order to prevent this conduct, LS proposed will accept worse solutions in the neighborhood provided two conditions are carried out: the difference of fitness between current and neighbor solution will be not greater than one unit and a certain number of evaluations of EA in the execution have been reached (we consider overcome the half of total number of them).

By using this strategy of local optimization of a chromosome, we can distinguish between *Total evaluation* and *Partial evaluation*.

- *Total Evaluation*: It consists in a standard evaluation of performance of a chromosome in EPS, that bears to compute the nearest neighbor of each instance belongs to subset selected and take the account of the instances classified correctly.
- *Partial Evaluation*: It can take place when it accomplish on a neighbor solution of a current already evaluated and differs only in a bit position, which have changed from value 1 to 0. If a total evaluation counts as one evaluation in terms of taking account of number of evaluations for the stop condition, a partial evaluation counts as:

$$\frac{N_{nu}}{|TR|} \tag{3}$$

where N_{nu} is the number of neighbors updated when a determined instance is removed by LS procedure and $|TR|$ is the size of the original set of instances (also is the size of the chromosome).

If a structure $U = \{u_1, u_2, ..., u_n\}$ is defined, where $u_i/i = 1, ..., n$ represents the identifier of the nearest instance to the instance i, considering only instance subset selected by the chromosome, a reduction of time complexity can be achieved. A partial evaluation can take advantage of U and of the divisible nature of the PS problem when instances are removed. Note that if instances are added (changes from 0 to 1 are allowed), the update of U is neither partial nor an efficient process because all neighbors have to been computed again. In this sense, the PS problem have characteristics of divisible nature.

An example is illustrated in figure 1, where a chromosome of 13 instances is considered. LS procedure removes the instance number 3. Once removed, the instance number 3 can not appears into U structure as nearest neighbor of another instance. U must be updated at this moment obtaining the new nearest neighbors for the instances that had instance number 3 as nearest neighbor. Then, a relative fitness with respect original chromosome fitness is calculated (instances 1, 5, 11 and 13).

Class	Instances
A	{1,2,3,4,5,6,7}
B	{8,9,10,11,12,13}

	Current Solution	→	Neighbor Solution
Representation	0110110100010	→	01**0**0110100010
U structure	$\{3, 5, 8, 8, 3, 2, 6, 2, 8, 8, 3, 2, 3\}$	→	$\{\mathbf{12}, 5, 8, 8, \mathbf{2}, 2, 6, 2, 8, 8, \mathbf{8}, 2, \mathbf{8}\}$
Fitness	{1,1,0,0,1,1,1,0,1,1,0,0,0}	→	{-1,*,*,*,0,*,*,*,*,*,+1,*,+1}
	7	→	$7 - 1 + 1 + 1$
Partial evaluation account: $\frac{N_{nu}}{\lvert T \rvert} = \frac{4}{13}$			
Neighbor Fitness: 8			

Fig. 1. Example of a move in LS procedure and a partial evaluation

4 Experiments and Results

In this section, the behavior of the EPS proposed is analyzed using 13 data sets taken from the UCI Machine Learning Database Repository [12] and compared with others non-evolutionary PS algorithms, such as ENN [13], CNN [14], RNN [15], IB3 [16], DROP3 [3] and RMHC [17] (brief descriptions can be found in [3]). The main characteristics of these data sets are summarized in Table 1.

The data sets considered are partitioned using the *ten fold cross-validation (10-fcv)* procedure. Whether either small or medium data sets are evaluated, the parameters used are the same, see Table 2. The scale of size of data sets and parameters for the algorithms follow the instructions given in [6].

Tables 3 and 4 show us the average of the results offered by each algorithm for small data sets and medium data sets, respectively. Each column shows:

– The first column shows the name of the algorithm. Our proposal of Incorporating Knowledge in a Genetic Algorithm for PS problem will be labeled by *IKGA* and will be followed by the number of evaluations executed to reach the stop condition. It has been executed considering 5000 and 10000 evaluations in order to check its behavior.

Table 1. A brief summary of the experimental data sets

Name	N. Instances	N. Features.	N. Classes.	Size
Bupa	345	7	2	small
Cleveland	297	13	5	small
Glass	294	9	7	small
Iris	150	4	3	small
Led7Digit	500	7	10	small
Lymphography	148	18	4	small
Monks	432	6	2	small
Pima	768	8	2	small
Wine	178	13	3	small
Wisconsin	683	9	2	small
Pen-Based	10992	16	10	medium
Satimage	6435	36	7	medium
Thyroid	7200	21	3	medium

Table 2. Parameters considered for the algorithms

Algorithm	Parameters
CHC	$Pop = 50, Eval = 10000, \alpha = 0.5$
IB3	$Acept.Level = 0.9, DropLevel = 0.7$
IKGA	$Pop = 10, Eval = [5000\|10000], p_m = 0.01, p_c = 1$
PBIL	$LR = 0.1, Mut_{shift} = 0.05, p_m = 0.02, Pop = 50$ $Negative_{LR} = 0.075, Eval = 10000$
RMHC	$S = 90\%, Eval = 10000$

- The second column contains the average execution time associated to each algorithm. The algorithms have been run in a Pentium 4, 3 GHz, 1 Gb RAM.
- The third column shows the average reduction percentage from the initial training sets.
- Fourth and Fifth columns contains the training accuracy when using the training set selected S_i from the initial set TR_i and the test accuracy of S_i over the test data set TS_i, respectively.

In the third, fourth and fifth columns, the best result per column are shown in bold.

By studying the tables 3 and 4, two drawbacks can be appreciated:

- The evaluation of just the mean classification accuracy over all the data sets hides important information due to a better/worse behavior of an algorithm associated to a determined data set.
- Each data set represents a different classification problem and different data sets have many different degrees of difficulty.

To avoid these drawbacks, we have included a second type of table accomplishing a statistical comparison of methods over multiple data sets. Demšar [18] recommends a set of simple, safe and robust non-parametric tests for statistical comparisons of classifiers. One of them is Wilcoxon Signed-Ranks Test [19,20]. Table 5 collects results of applying Wilcoxon test between our proposed methods and rest of PS algorithm studied in this paper over the 13 data sets considered. This table is divided in three parts: In the first part, we accomplish Wilcoxon test by using as performance measure only the reduction of the training set; in the second part, the measure of performance used is the accuracy

Table 3. Average results for small data sets

Algorithm	Time(s)	Reduction	Ac. Train	Ac. Test
1NN	0.02	-	73.52	72.52
CHC	8.67	97.45	79.91	74.14
CNN	0.03	65.70	64.76	68.35
DROP3	0.17	83.53	73.50	67.90
ENN	0.02	25.21	77.68	73.44
IB3	0.02	69.17	64.17	69.90
PBIL	31.05	93.99	81.55	73.96
IKGA 5000 Ev.	6.20	98.43	74.77	**76.37**
IKGA 10000 Ev.	13.49	**98.46**	74.71	75.97
RMHC	17.50	90.18	**83.80**	74.98
RNN	4.63	92.43	74.59	73.37

Table 4. Average results for medium data sets

Algorithm	Time(s)	Reduction	Ac. Train	Ac. Test
1NN	31.56	-	94.19	94.18
CHC	8945.91	**99.55**	93.02	92.45
CNN	2.31	88.80	88.86	90.58
DROP3	186.82	94.62	89.17	87.71
ENN	10.78	5.88	95.11	**94.41**
IB3	5.48	73.02	91.87	92.80
PBIL	51165.29	83.82	94.79	93.80
IKGA 5000 Ev.	2394.07	99.29	94.84	93.81
IKGA 10000 Ev.	4996.26	99.34	94.95	93.74
RMHC	8098.15	90.00	**96.35**	94.06
RNN	23822.91	96.57	94.22	92.81

Table 5. Wilcoxon test

Reduction Performance										
	IKGA 5000 Ev.	IKGA 10000 Ev.	CHC	CNN	DROP3	ENN	IB3	PBIL	RMHC	RNN
IKGA 5000 Ev.	=	=	+	+	+	+	+	+	+	+
IKGA 10000 Ev.	=	=	+	+	+	+	+	+	+	+
Accuracy in Test Performance										
	IKGA 5000 Ev.	IKGA 10000 Ev.	CHC	CNN	DROP3	ENN	IB3	PBIL	RMHC	RNN
IKGA 5000 Ev.	=	=	+	+	+	=	+	=	=	=
IKGA 10000 Ev.	=	=	+	+	+	=	+	=	=	=
Accuracy in Test $*$ 0.5 + *Reduction* $*$ 0.5										
	IKGA 5000 Ev.	IKGA 10000 Ev.	CHC	CNN	DROP3	ENN	IB3	PBIL	RMHC	RNN
IKGA 5000 Ev.	=	=	+	+	+	+	+	+	+	+
IKGA 10000 Ev.	=	=	+	+	+	+	+	+	+	+

classification in test set; in third part, a combination of reduction an classification accuracy is used for performance measure. This combination corresponds to $0.5 \cdot clas\_rat + 0.5 \cdot perc\_red$. Each part of this table contains two rows, representing our proposed methods, and N columns where N is the number of algorithms considered in this study. In each one of the cells can appear two symbols: + or =. They represent that the algorithm situated in that row outperforms (+) or is similar (=) in performance that the algorithm which appear in the column (Table 5).

The following analysis seeing this results can be made:

- IKGA presents the best reduction and test accuracy rates in Table 3.
- In Table 4, IKGA offers a good test accuracy rate and maintains a high reduction rate.

- When the problem scales up to medium data sets, IKGA avoids premature convergence observed in CHC and reach similar test accuracy than PBIL, but provides more reduction rate.
- Time execution over small and medium data sets decreases considerately by using IKGA with respect the remaining EAs.
- IKGA executed with 5000 evaluations and 10000 evaluations have similar behavior. This indicates that this algorithm carries out a good trade-off between exploitation and exploration over the search space.
- In Table 5, IKGA outperforms all methods considering that both objectives (reduction and test accuracy) have the same importance. Furthermore, Wilcoxon accuracy test considers it equal to classical algorithms such as ENN, RNN or RMHC, but these algorithms don't reach its reduction rate.

5 Concluding Remarks

In this paper, we have presented an Evolutionary Algorithm that incorporates knowledge on the Prototype Selection problem. The results shows that incorporating knowledge can obtain an improvement on accuracy and a better reduction of data. Furthermore, a decrement of time complexity is got with respect others Evolutionary Prototype Selection algorithms studied in the literature.

Acknowledgement. This work was supported by TIN2005-08386-C05-01 and TIN2005-08386-C05-03.

References

1. Cover, T.M., Hart, P.E.: Nearest neighbor pattern classification. IEEE Transactions on Information Theory **13** (1967) 21–27
2. Liu, H., Motoda, H.: On issues of instance selection. Data Min. Knowl. Discov. **6** (2002) 115–130
3. Wilson, D.R., Martinez, T.R.: Reduction techniques for instance-based learning algorithms. Machine Learning **38** (2000) 257–286
4. Grochowski, M., Jankowski, N.: Comparison of instance selection algorithms II. Results and comments. In: ICAISC. (2004) 580–585
5. Ishibuchi, H., Nakashima, T.: Evolution of reference sets in nearest neighbor classification. In: SEAL'98: Selected papers from the Second Asia-Pacific Conference on Simulated Evolution and Learning on Simulated Evolution and Learning, London, UK, Springer-Verlag (1999) 82–89
6. Cano, J.R., Herrera, F., Lozano, M.: Using evolutionary algorithms as instance selection for data reduction in KDD: An experimental study. IEEE Transactions on Evolutionary Computation **7** (2003) 561–575
7. Cano, J.R., Herrera, F., Lozano, M.: Stratification for scaling up evolutionary prototype selection. Pattern Recogn. Lett. **26** (2005) 953–963
8. Eiben, A.E., Smith J.E.: Introduction to Evolutionary Computing. SpringerVerlag (2003)
9. Goldberg, D.E.: Genetic Algorithms in Search, Optimization, and Machine Learning. Addison Wesley (1989)

10. Eshelman, L.J.: The CHC adaptative search algorithm: How to safe search when engaging in nontraditional genetic recombination. In: FOGA. (1990) 265–283
11. Baluja, S.: Population-based incremental learning: A method for integrating genetic search based function optimization and competitive learning. Technical report, Pittsburgh, PA, USA (1994)
12. Newman, D.J., Hettich, S., Merz, C.B.: UCI repository of machine learning databases (1998)
13. Wilson, D.L.: Asymptotic properties of nearest neighbor rules using edited data. IEEE Transactions on Systems, Man and Cybernetics **2** (1972) 408–421
14. Hart, P.E.: The condensed nearest neighbour rule. IEEE Transactions on Information Theory **18** (1968) 515–516
15. Gates, G.W.: The reduced nearest neighbour rule. IEEE Transactions on Information Theory **18** (1972) 431–433
16. Aha, D.W., Kibler, D., Albert, M.K.: Instance-based learning algorithms. Machine Learning **7** (1991) 37–66
17. Skalak, D.B.: Prototype and feature selection by sampling and random mutation hill climbing algorithms. In: ICML. (1994) 293–301
18. Demšar, J: Statistical comparisons of classifiers over multiple data sets. Journal of Machine Learning Research **7** (2006) 1–30
19. Wilcoxon, F.: Individual comparisons by rankings methods. Biometrics **1** (1945) 80–83
20. Sheskin, D.J.: Handbook of Parametric and Nonparametric Statistical Procedures. CRC Press (1997)