Evolutionary Multi-Objective Approach for Prototype Generation and Feature Selection

Alejandro Rosales-Pérez[1], Jesus A. Gonzalez[1], Carlos A. Coello Coello[2], Carlos A. Reyes-Garcia[1], and Hugo Jair Escalante[1]

[1] Instituto Nacional de Astrofísica, Óptica y Electrónica (INAOE), Computer Science Department, Tonantzintla, Puebla, Mexico
{arosales,hugojair,jagonzalez,kargaxxi}@inaoep.mx
[2] CINVESTAV-IPN, Computer Science Department, Mexico City, Mexico
ccoello@cs.cinvestav.mx

Abstract. This paper introduces EMOPG+FS, a novel approach to prototype generation and feature selection that explicitly minimizes the classification error rate, the number of prototypes, and the number of features. Under EMOPG+FS, prototypes are initialized from a subset of training instances, whose positions are adjusted through a multi-objective evolutionary algorithm. The optimization process aims to find a set of suitable solutions that represent the best possible trade-offs among the considered criteria. Besides this, we also propose a strategy for selecting a single solution from the several that are generated during the multi-objective optimization process.We assess the performance of our proposed EMOPG+FS using a suite of benchmark data sets and we compare its results with respect to those obtained by other evolutionary and non-evolutionary techniques. Our experimental results indicate that our proposed approach is able to achieve highly competitive results.

1 Introduction

The k-nearest neighbor (k-NN) classifier is one of the most well-known methods for pattern classification. Its popularity relies on its simplicity and high performance. This method belongs to the lazy learning algorithms family, which implies that a training phase is not performed. Instead, the standard k-NN stores the entire training set and, for classifying a test instance, it performs as many similarity computations as samples are available in the training set. These are major concerns when using the k-NN classifier on large data sets.

To overcome the aforementioned shortcomings, a number of techniques have been proposed aiming to reduce the number of instances in the original training set, while trying to maintain a good classification performance. These techniques can be grouped into two major approaches: prototype selection methods [5], which attempt to select a representative subset of samples from the training set, and prototype generation (PG) methods [10], whose goal is to generate a small set of artificial prototypes to replace the original training set. In this work, we focus on PG because prototype selection can be seen as a special case of PG.

Among the studies that have approached the PG problem, those based on bio-inspired optimization have gained popularity in recent years [1–4,6,8–10]. These generally seek at optimizing a criterion related to the classification performance of the generated prototypes. In few cases an instance reduction term is also considered in an aggregated single criterion. In spite of the satisfactory results that have been reported with these methods, we believe that the optimization of a single objective may not be the best option for PG, since classification performance and training set reduction could be in conflict with each other. Moreover, most of the existing studies only focused on instance reduction, while dimensionality reduction is not taken into account. To the best of our knowledge, the PG has not been dealt with, considering dimensionality reduction at the same time in a multi-objective approach. These issues can be treated in a natural fashion as a multi-objective optimization problem that explicitly deals with these three goals: preserving a good classification performance, reducing the prototype set size, and reducing the number of features.

This paper introduces EMOPG+FS: an evolutionary multi-objective approach that seeks to reduce the number of training samples, through the generation of prototypes, and the number of features, by means of selecting a subset of them, while preserving a good classification performance. EMOPG+FS adopts a positioning adjustment approach for generating the prototypes, i.e., each pattern is represented as a point in the feature space, whose position is modified by an optimization process. To this end, we use PAES (Pareto Archived Evolution Strategy) [7] as our multi-objective optimization technique. The main contribution of this paper is to advance the state of the art in the area, by simultaneously addressing the prototype generation and the feature selection problems as a multi-objective one, with the aim of optimizing in a single run, the three aforementioned criteria. We report experimental results in a suite of data sets used for benchmarking PG methods and compare the performance of our proposed approach to that of alternative techniques [6, 10]. Experimental results give evidence of the suitability of EMOPG+FS for the problem at hand.

The remainder of this paper is organized as follows. Section 2 explains in detail our proposal, EMOPG+FS. Section 3 reports experimental results. Finally, Section 4 presents our conclusions, and outlines future work.

2 EMOPG+FS: Evolutionary Multi-Objective Prototype Generation and Feature Selection

In this work, the PG and FS tasks are treated as a multi-objective optimization problem where the three considered objectives are: (1) the minimization of the 1-NN classification error in the training set when using the prototypes, (2) the reduction in the number of prototypes, and (3) the reduction in the number of features. Moreover, we constrain the set of feasible solutions ($\mathcal{X}$) to be formed by all possible sets of prototypes that have at least one prototype per class and are described by at least one feature.

Algorithm 1 EMOPG+FS

Require: $\mathbf{X}$: training set,
N: maximum number of prototypes,
k : number of nearest neighbors,
IC: number of instances competing in a tournament,
MOEA's parameters
Ensure: A set of prototypes
1: Let $\mathbf{N} = [n_1, \ldots, n_m]$ be the number of instances for each class, such that $\sum_{i=1}^{m} n_i = N$ for m classes
{Weight each instance in the training set based on its k nearest neighbors from other classes}
2: **for** each instance $\mathbf{x}_i \in \mathbf{X}$ **do**
3: Find the k nearest neighbor from other classes
4: Compute the weight of the instance $\mathbf{x}_i$ using equation (1)
5: **end for**
{Construct an initial set of N prototypes giving preference to border instances}
6: **for** each class $c_i \in \mathbf{C}$ **do**
7: **while** the cardinality of prototypes from $c_i < n_i$ **do**
8: Choose randomly IC prototypes from $\mathbf{X}$ that belong to c_i
9: Add to the set the prototype with the highest weight among the IC prototypes
10: **end while**
11: **end for**
12: Apply a multi-objective evolutionary algorithm for adjusting the positions of the prototypes
13: Select a single solution from the resulting non-dominated front based on some preference

For generating the prototypes, we start with a subset of training samples, which are represented as points in a multi-dimensional space. Their positions are adjusted through an optimization process. The initial training samples are selected according to a weighting scheme that takes into account their discriminative power. The outcome of the multi-objective optimization process is a set of solutions (i.e., sets of generated prototypes each with a subset of selected features) that correspond to the best possible trade-offs among the objectives (i.e., no objective can be improved without worsening another). A single solution must be chosen from this set in order to use it in the classification task. In this regard, we propose a strategy to select a single solution. Algorithm 1 describes the proposed approach, and it is detailed next.

Weighting Instances: We first assign a weight to each training instance in order to know its discriminative capabilities. The main idea is that instances that are closer to others from different classes should have a higher weight than those that are farther away, since the instances closest to the borders are expected to be the most difficult to classify, and they would provide more information that can allow us to discriminate among classes. This score is somewhat related to assumptions in SVM classifiers.

The weighting procedure is described in steps 2 to 5 of Algorithm 1. Let $\mathcal{N} = \{\mathbf{x}_1, \ldots, \mathbf{x}_k\}$ be the set of k nearest neighbors from other classes of an instance $\mathbf{x}_i$, the weight of this instance is computed as:

$$w(\mathbf{x}_i) = \frac{1}{k} \sum_{\mathbf{x}_j \in \mathcal{N}}^{k} \frac{1}{\|\mathbf{x}_i - \mathbf{x}_j\|} \tag{1}$$

where $\|\mathbf{x}_i - \mathbf{x}_j\|$ is the Euclidean norm between $\mathbf{x}_i$ and $\mathbf{x}_j$. This weight is used to generate the initial prototypes as is described next.

Constructing an Initial Set of Prototypes: The second step of our proposal is to generate an initial set of prototypes through the selection of samples from the training set. The procedure for choosing an initial set of N prototypes is described in steps 6 to 11 in Algorithm 1. We determine the maximum number of prototypes for each class, which is done in a stratified fashion, aiming to preserve, in the prototypes set, the original proportions of examples from each class as in the training set. For each class, we select IC prototypes at random, and the one with the highest weight is added to the initial set of prototypes. This process is repeated until the maximum number of prototypes is reached. One should note that by proceeding in this manner, we guarantee that there is at least one prototype per class in the set of initial prototypes.

Adjusting the Positions of Prototypes: We adopted a positioning adjustment approach to generate the prototypes. This is done through an evolutionary process. We expect that the initial set of prototypes is a reasonably good solution, which should be improved by a local search engine that takes into consideration the three objectives in our formulation. One should note that under this assumption, our method can be seen as a post-processing step that can be applied after any other PG method. To this aim, we used the (1+1)-PAES [7] for solving this problem. A description of PAES is given in [7]. Next, we explain the application of PAES to adjust the positions of the initial set of prototypes.

Representation: As we previously stated, we want to adjust the position of prototypes in the feature space. To achieve this task, the prototypes are encoded in an $N \times d$ dimensional vector, where N is the maximum allowable number of prototypes and d is the dimensionality of each prototype in the feature space. We should recall that our objectives include the reduction in the number of prototypes and the number of features. Therefore, the proposed encoding scheme should also consider a mechanism to deal with these two criteria. With this in mind, each potential solution to the problem (i.e., a set of prototypes) is represented in an $N \times (d+1) + d$ dimensional vector as follows:

$$\mathbf{x}^{(i)} = \left[f_1^1, \ldots, f_1^d, \ldots, f_N^1, \ldots, f_N^d, b_1, \ldots, b_N, b_{N+1}, \ldots, b_{N+d}\right] \tag{2}$$

where $f_i^j \in \mathbb{R}$ represents the j^{th} feature value of the i^{th} prototype, and $b_i \in \{0, 1\}$ is a variable that indicates whether the corresponding prototype/feature is considered or not.

One should note that the class label is not encoded in the adopted representation. This is because the initial set of prototypes is chosen from the training set.

Therefore, each sample has a class label, which is defined *a priori*, and remains unchanged during the evolutionary search.

In (1+1)-PAES, mutation is the only evolutionary operator used for creating an offspring from a parent. The adopted representation is a mixed-encoding, involving both real and binary variables. Hence, we propose to mutate both real and binary variables independently. Therefore, the individual is decomposed in two parts: the real part and the binary part. For each part, an *ad-hoc* mutation operator is applied. For the real-numbers part, we used polynomial-based mutation, and bit-flip mutation was adopted for the binary part.

Fitness Functions: In order to evaluate how good an individual is, we need to assess it with the considered optimization criteria. We consider three objectives, related to the classification performance and reductions in the number of prototypes and in the number of features. The first objective, $f_1(\mathbf{x})$, is assessed through a fitness function that accounts for the error incurred by the prototypes when used with a 1-NN rule to classify the training set. The second objective, $f_2(\mathbf{x})$, is captured by a fitness function indicating the relative reduction rate attained by a specific individual, and the third objective, $f_3(\mathbf{x})$, takes into account the reduction in the number of features. One should note the constraint that the set of prototypes must have at least one prototype for each class and at least one feature. This constraint is handled in a straightforward fashion following a penalty function approach, i.e., we penalize the solutions that do not satisfy these constraints.

Thus, the goal of PAES is to search the space of prototypes aiming to simultaneously optimize the stated criteria. PAES returns a set of non-dominated solutions found during the search, from a single one that represents the set of prototypes is chosen.

Selecting a Single Solution: PAES returns a set of non-dominated solutions, which is expected to be an approximation to the true Pareto optimal set. Each of these non-dominated solutions represents a set of prototypes to be used as a reduced data set for the 1NN classifier. Hence, it is desirable to choose a single solution from that set. For doing so, we first define what an ideal solution to the problem would be. An ideal solution would not cause errors to classify the samples, it would have one prototype per class, and it would have the minimum number of features, i.e., $z_{ideal} = \left[0, {}^{m}/_{N}, {}^{1}/_{d}\right]$, for a problem with m classes. We adopted a compromise programming approach to choose a solution, and the Tchebycheff metric as the distance measure. Thus, the solution is chosen through the following expression:

$$S^* = \underset{\mathbf{x}}{\operatorname{argmin}} \left[\max \left\{ f_1(\mathbf{x}), \left| f_2(\mathbf{x}) - {}^{m}/_{N} \right|, \left| f_3(\mathbf{x}) - {}^{1}/_{d} \right| \right\}\right] \tag{3}$$

3 Experiments and Results

For our experiments, we used 59 data sets taken from the KEEL repository,[3] which were also used in the comparative study of PG methods performed by

[3] These data sets are available at http://sci2s.ugr.es/keel/datasets.php

Table 1. Results obtained by EMOPG+FS in terms of classification performance and reduction rate averaged over different data sets for all, for the small and for the large data sets. It also shows the results obtained by other PG methods. The best results are shown in **boldface**.

Method	**Accuracy**		
	All	Small	Large
1-NN	74.79 ± 18.48	72.45 ± 16.08	79.73 ± 22.24
AMPSO	70.66 ± 17.68	69.03 ± 15.92	74.10 ± 20.97
GENN	$\mathbf{77.47 \pm 17.71}$	$\mathbf{75.64 \pm 15.45}$	81.33 ± 21.70
LVQTC	70.05 ± 18.74	69.81 ± 17.44	70.56 ± 21.74
MSE	73.78 ± 17.64	72.37 ± 14.81	76.74 ± 22.69
PSCSA	66.90 ± 19.68	66.82 ± 18.74	67.07 ± 22.05
PSO	76.62 ± 16.39	75.01 ± 14.09	79.99 ± 20.44
SGPFGP	74.64 ± 17.48	71.97 ± 15.76	80.25 ± 19.93
EMOPG+FS	76.70 ± 16.89	74.26 ± 14.70	$\mathbf{81.82 \pm 20.24}$

Triguero et al. [10]. These data sets are divided according to the number of samples, in small data sets (less than 2000 samples) and large data sets (2000 and more samples). Each of these data sets were previously partitioned into 10 training/test subsets by means of a 10 fold cross validation procedure. For each data set, we applied our proposal (EMOPG+FS) 10 times: one for each of the training partitions, in order to generate a prototypes set, whose performance is assessed by using the test set. This leads to a total of 590 experiments performed for PG. For assessing the performance of the PG methods we considered the three criteria: test-set accuracy, training-set reduction, and dimensionality reduction. We compared the experimental results obtained by our proposed method with those obtained by other evolutionary and non-evolutionary methods for PG reported in [10], by SGPFGP [6], and by the 1-NN classifier. It is worth mentioning that SGPFGP is a recent approach for dealing simultaneously both training-set reduction and dimensionality reduction via a single-objective genetic programming, in which the criterion to optimize is the accuracy performance.

Table 1 shows the mean and standard deviation obtained by EMOPG+FS and those obtained by the reference studies. The results are shown separately, in terms of test-set accuracy, when considering: all the data sets (59 data sets), only small data sets (40 data sets), and only large data sets (19 data sets). From this table, we can see that GENN, PSO, and EMOPG+FS get a little better accuracy-performance than the one obtained by the 1-NN classifier for all cases. GENN further reached the best accuracy performance for both all and the small data sets. With respect to the large data sets, the performances of GENN SGPFGP, and EMOPG+FS are very close. It is also remarkable that PSCSA showed the worst performance among the considered methods for all cases.

Table 2 shows the results in terms of the reduction rates, both in prototypes and features, attained by each method. We can observe that PSCSA obtained the best prototype set size reduction rate in all cases. EMOPG+FS is the third best method for both, all and the small data sets, and the fifth one for the large

Table 2. Results obtained by EMOPG+FS in terms of the reduction rate in the number of prototypes and the number of features averaged over different data sets for all, for the small and for the large data sets. It also shows the results obtained by other PG methods. The best results are shown in **boldface**.

Method	Reduction Rate					
	Prototypes			**Features**		
	All	Small	Large	All	Small	Large
AMPSO	95.49 ± 1.86	94.39 ± 0.99	97.97 ± 0.09	00.00 ± 0.00	00.00 ± 0.00	00.00 ± 0.00
GENN	17.70 ± 14.93	19.91 ± 14.48	15.76 ± 19.92	00.00 ± 0.00	00.00 ± 0.00	00.00 ± 0.00
LVQTC	96.87 ± 3.13	95.61 ± 2.96	99.75 ± 0.16	00.00 ± 0.00	00.00 ± 0.00	00.00 ± 0.00
MSE	96.54 ± 4.66	95.30 ± 5.10	99.36 ± 0.73	00.00 ± 0.00	00.00 ± 0.00	00.00 ± 0.00
PSCSA	$\mathbf{99.00 \pm 1.37}$	$\mathbf{98.60 \pm 1.47}$	$\mathbf{99.88 \pm 0.17}$	00.00 ± 0.00	00.00 ± 0.00	00.00 ± 0.00
PSO	95.90 ± 1.57	94.97 ± 0.83	97.99 ± 0.08	00.00 ± 0.00	00.00 ± 0.00	00.00 ± 0.00
SGPFGP	98.72 ± 1.23	98.39 ± 1.37	99.43 ± 0.09	42.24 ± 8.27	41.45 ± 12.76	42.62 ± 5.13
EMOPG+FS	97.68 ± 1.35	97.32 ± 1.49	98.42 ± 0.46	$\mathbf{56.80 \pm 23.16}$	$\mathbf{60.47 \pm 19.14}$	$\mathbf{49.09 \pm 29.00}$

data sets. The worst method in terms of training-set reduction was GENN. Furthermore, most of the compared methods do not consider the dimensionality reduction. The only one exception is SGPFGP, which attained a dimensionality reduction around a 40%, while in EMOPG+FS is around a 50%. This reduction evidently would represent computational-savings in the similarity computation.

We performed a Friedman test to know if there is a statistical significant difference among GENN, PSCSA, PSO, 1-NN, SGPFGP, and EMPOG+FS. We applied it with a 95% of confidence and the Bonferroni-Dunn test is used as a post-hoc test. We summarize the results obtained by these tests as follows:

- In terms of accuracy performance for all, small and large data sets, there is not a statistically significant difference between EMOPG+FS, GENN, PSO, and 1-NN. EMOPG+FS performs significantly better than all of these methods in terms of prototype set size reduction.
- EMOPG+FS significantly outperforms PSCSA in all, small, and large data sets in terms of accuracy performance, but it is statistically worst in terms of the prototype set size reduction.
- SGPFGP performs significantly better than EMOPG+FS in training set reduction, but EMOPG+FS significantly outperformed it in terms of dimensionality reduction. Moreover, EMPOG+FS is statistically superior than SGPFGP in accuracy performance for all and small data sets.

Summarizing, GENN obtained very good results in terms of accuracy. However, it had the worst reduction rate. On the other hand, PSCSA is the best with respect to the prototype set size reduction, but its performance on accuracy is the worst among the considered methods. Moreover, most of the methods considered for comparison contemplates dealing with the dimensionality reduction problem. The only one that takes into consideration the dimensionality reduction is SGPFGP, which is a single-objective approach. EMOPG+FS offers a more balanced trade-off between the considered objectives than any other of the considered methods, while it takes into account the dimensionality reduction task in a natural fashion via a multi- objective approach. Therefore, EMOPG+FS is

able to reduce the number of prototype samples as well as the number of features without significant over-fitting.

4 Conclusions and Future Work

We presented EMOPG+FS, an evolutionary multi-objective approach for dealing jointly with the prototype generation (PG) and the feature selection problems, while keeping a good accuracy performance. Experimental results showed that our proposal is able to obtain prototypes and to reduce the dimensionality of the data set, without significantly degrading the performance of k-NN. Besides this, the performance of EMOPG+FS over different data sets from different domains gives evidence of the suitability of using it as a general method for this task.

Our future work involves extending our proposal to deal with the user's preferences during the optimization process. We would also like to evaluate the performance of EMOPG+FS using different values of k for the k-NN classifier. Studying the impact of the evolutionary parameters on the quality of the solutions generated by EMOPG and testing EMOPG+FS on very large scale data sets are other potential paths for future research.

References

1. Cervantes, A., Galvan, I., Isasi, P.: AMPSO: a new particle swarm method for nn classification. IEEE Trans. Sys. Man Cy. B 39(5), 1082–1091 (2009)
2. Chen, J.H., Chen, H.M., Ho, S.Y.: Design of nearest neighbor classifiers: multi-objective approach. Int. J. Approx. Reason. 40(1-2), 3–22 (2005)
3. Escalante, H.J., Marín-Castro, M., Morales-Reyes, A., Graff, M., Rosales-Pérez, A., Montes-y Gómez, M., Reyes-Garcia, C.A., Gonzalez, J.A.: MOPG: A Multi-Objective Evolutionary Algorithm for Prototype Generation (2014), submitted.
4. Escalante, H.J., Mendoza, K.M., Graff, M., Morales-Reyes, A.: Genetic programming of prototypes for pattern classification. In: Proc. of IbPRIA 2013. LNCS, vol. 7887, pp. 100–107. Springer (2013)
5. García, S., Derrac, J., Cano, J.R., Herrera, F.: Prototype selection for nearest neighbor classification: Taxonomy and empirical study. IEEE Trans. Pattern Anal. Mach. Intell. 34(3), 417–435 (2012)
6. García-Limón, M., Escalante, H.J., Morales, E., Morales-Reyes, A.: Simultaneous Generation of Prototypes and Features through Genetic Programming. In: Proc. of GECCO. pp. 517–524 (2014)
7. Knowles, J., Corne, D.: Approximating the nondominated front using the pareto archived evolution strategy. Evol. Comput. 8(2), 149–172 (2000)
8. Rosales-Pérez, A., Escalante, H.J., Coello Coello, C.A., Gonzalez, J.A., Reyes-Garcia, C.A.: An Evolutionary Multi-Objective Approach for Prototype Generation. In: Proc. of IEEE WCCI. pp. 1100–1007 (2014)
9. Triguero, I., Garcia, S., Herrera, F.: Differential evolution for optimizing the positioning of prototypes in nearest neighbor classification. Pattern Recogn. 44, 901–916 (2011)
10. Triguero, I., Derrac, J., Garcia, S., Herrera, F.: A taxonomy and experimental study on prototype generation for nearest neighbor classification. IEEE Trans. Syst. Man Cy. C 42(1), 86–100 (2012)