2009 Ninth International Conference on Intelligent Systems Design and Applications

# Improved Accuracy Rates of a Prototype Based Classifier Using Evolutionary Computation

Gustavo Recio, Yago Saez and Pedro Isasi
Department of Computer Science, Carlos III University, Madrid, Spain
Evolutionary Algorithms, Neural Networks and Artificial Intelligence
(EVANNAI) grecio@inf.uc3m.es; ysaez@inf.uc3m.es; isasi@ia.uc3m.es

## Abstract

*Prototype based classifiers allow to determine the class of a new example based on a reduced set of prototypes instead of using a large set of known samples. By doing this, the computational time gets substantially decreased as the initial set is replaced by a reduced one and hence the classification requires less computations to estimate nearest neighbours. In most simple classification problems the samples associated to each class are in general gathered in a particular region of the euclidean space defined by their characteristic features.*

*In these particular problems prototype classifiers reach their best performance. Unfortunately, not all classification problems have their samples distributed in this way and therefore improvements are needed in order to reach acceptable classification accuracy rates. This work proposes a nearest prototype classifier that uses evolutionary computation techniques to increase the classification accuracy. A genetic algorithm was used to evolve the spatial location of each prototype resulting in a better distribution of prototypes which are able to obtain larger classification accuracy rates.*

## 1. Introduction

Nearest prototype classifiers [1] allow to determine the class of a new example on the basis of a set of previously classified prototypes. The way of obtaining this set of prototypes is based on selecting them from an original set of labelled samples, or by replacing the original set by a different and reduced one [2]. Learning vector quantisation algorithms [3] are a family of algorithms focused on nearest prototype classification. They are based on a the dynamical computation of a set of prototypes in order to minimise the classification error. Several approaches are based on this model [4]. Some other approaches are based on local weighting in nearest prototype classification [5]. The way they do this is by providing the prototypes with their continuous weight vector. Thus, the algorithm follows a prototype-specific weighting approach, instead of a typical instance-specifying one [6]. Other approaches are based on the so called evolutionary nearest prototype classification algorithm [7] which is a nearest prototype approach that follows an evolutionary process to compute a correct number of prototypes. Prototype classification has also been applied to handwriting recognition in a method that employs a learning process to determine both the number and the location of prototypes [8].

Next section describes in detail the hybrid method that combines a prototype based classifier and evolutionary computation techniques. Section 2.1 deals with the generation of prototypes from the databases whereas Section 2.2 describes the implementation of the genetic algorithm used to evolve the spatial location of the prototypes. The databases used in this work are described in Section 3. In Section 4, the experiments carried out in this research are explained and followed by the main conclusion and future work that arose as a result of this work.

## 2. Hybrid Classifier

In this paper a hybrid classification technique that uses a prototype selection mechanism to generate the seed of a genetic algorithm is presented. The algorithm can be divided in two different stages, in the first one the number of prototypes is specified and a first classification is done over the training set of instances. As this is not usually enough to obtain accurate classifiers over the validation set of instances, a second stage where the prototypes are reallocated to a different spatial location is needed. The first stage, which will be explained in next section, consists of a prototype based classifier, this classifier uses the euclidean distance to identify nearest neighbours, it also sets the number of prototypes and places them in a particular spatial location.

978-0-7695-3872-3/09 $26.00 © 2009 IEEE
DOI 10.1109/ISDA.2009.172

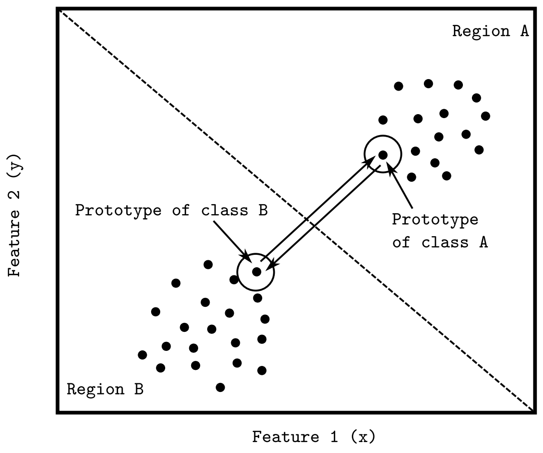

**Figure 1. Two dimensional artificial domain for illustration of prototype classification.**

The second stage is described in Section 2.2 and uses the prototype set previously obtained as the seed of a genetic algorithm which evolves the spatial location of the prototypes and aims at a better classification accuracy. The genetic algorithm itself would be no better than the hybrid technique as typical genetic algorithms use random initial populations wereas this technique uses as initial population the prototype set generated in the first stage of the algorithm.

## 2.1. Selection of Prototypes

Reducing the number of computations at the time of sorting unknown instances is the motivation behind prototype classifiers. In this manner, each prototype replaces a set of instances of the same class as their prototype. Figure 1 represents an artificial two-dimensional domain where instances of two different classes, A and B, are distributed in two separate regions of the euclidean space which is defined by the characteristic features or attributes of the instances. Using the euclidean distance as the metric to find nearest neighbours, it is clear that each set of instances in regions A and B can be replaced by a single prototype associated with a particular class (circled instances). Then it is possible to sort all instances in Figure 1 by choosing which of the two prototypes is closest to each instance.

Many prototypes can be chosen, however, this is an important decision which will affect the final accuracy of the classifier. The first stage of the proposed algorithm consists of a method for choosing such prototypes. That is implemented by computing for each instance its nearest neighbour of a different class using as metric the euclidean distance defined by their normalised attributes. Then, every couple of instances that satisfy being the nearest neighbour of each other are selected as prototypes. This can be clearly understood by looking at Figure 1, every instance of class A has an associated nearest neighbour of class B, however, there are only two instances that satisfy being the nearest neighbour of each other and only these are taken to build up the prototype set. This bi-univocal relation is represented by two parallel arrows with opposite directions whereas the prototypes are represented by circled instances.

This method provides a very large accuracy when handling simple and linear domains like the one represented in Figure 1. However, for more complicated domains it does not perform so well by itself and thus it needs further improvements which were achieved by including new prototypes in a second classifying round. It was noticed that the instances that were not correctly classified could be grouped according with the prototype they were confused with, i.e. instances of class B which distance to prototype A is smaller than their distance to prototype B (they are missclassed or their prototype is confused).

Within each group, the instance which distance to the missleading prototype is smaller will be selected as a new prototype. This process is repeated until 100% accuracy is obtained over the training set. Unfortunately, doing this some of the prototypes would be classifying just one instance and the classifier would be specialised in a particular training set. Also, during the second stage of the algorithm, i.e. the evolution of prototypes using a genetic algorithm, the fitness of individuals consists of the classification accuracy obtained by that set of prototypes over the training set, which can not be larger than 100%. Then, in order to evolve the spatial location of the prototypes some of the selected prototypes must be ignored. The removing policy uses a certain threshold to eliminate prototypes that do not classify a minimum number of instances. In this way the accuracy over the training set would be less than 100% and therefore a larger accuracy can be achieved by evolving the spatial location of the prototypes at the same time that the classifier would not be specialised. The selection of prototypes is summarised in the flowchart of Figure 2.

The proposed method, of course, can be extended for the $n$-dimensional case where each instance is defined by $n$ different attributes. Also, these basic principles can be applied to domains with $k$ classes. This is the general case for the actual datasets used through this work. Once the prototype set is obtained its spatial distribution must be optimised to obtain a larger classification accuracy.

## 2.2. Reallocation of Prototypes

The optimisation of the spatial location of each prototype can be done using a genetic algorithm. First, the genotype of this algorithm must be defined from the initial distribution of prototypes or phenotype. Take as an exam-

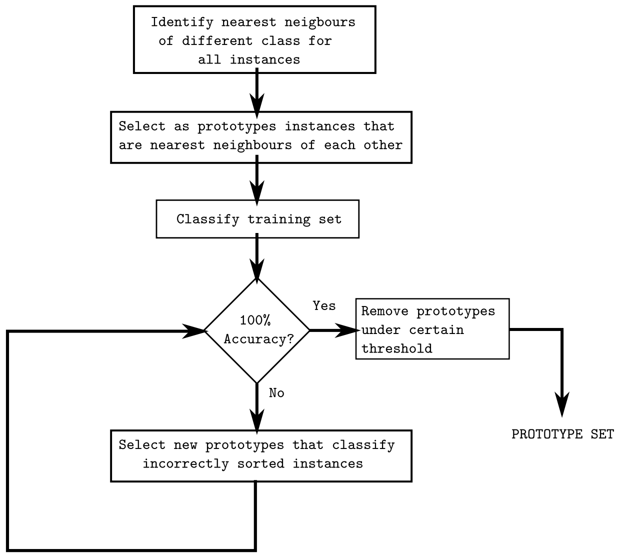
Identify nearest neigbours of different class for all instances
Select as prototypes instances that are nearest neighbours of each other
Classify training set
100% Accuracy?
Yes
No
Remove prototypes under certain threshold
PROTOTYPE SET
Select new prototypes that classify incorrectly sorted instances

**Figure 2. Flowchart for selection of prototypes.**

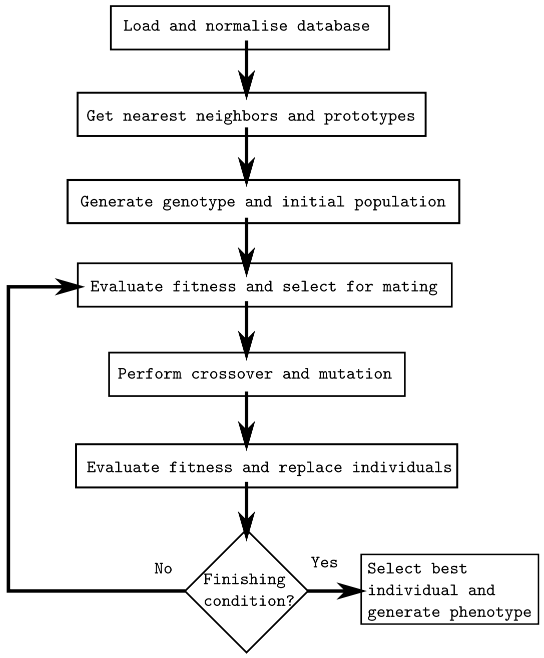
Load and normalise database
Get nearest neighbors and prototypes
Generate genotype and initial population
Evaluate fitness and select for mating
Perform crossover and mutation
Evaluate fitness and replace individuals
Finishing condition?
No
Yes
Select best individual and generate phenotype

**Figure 3. Flowchart of the algorithm.**

ple a two-dimensional domain with two possible classes (i.e. instances with two attributes that belong to one of two classes). Assume that through the above process six prototypes of each class have been selected. Then, there are 12 prototypes with 2 attributes each. This phenotype can be represented by a single chromosome with 24 genes. Therefore, the genotype representation for the proposed genetic algorithm depends on the above prototype selection process, through the number of prototypes, and on the actual domain, through the number of attributes that define each instance. The genetic algorithm will modify the spatial location of each prototype in order to find the optimal set of prototypes.

The evaluation of fitness for each individual can be done by counting the number of instances that it is able to classify with no error. As in every evolutionary algorithm, individuals with the best fitness have more chances of being selected for reproduction and propagate their genes through future generations. The genetic algorithm is linked to the prototype classifier through the initial population which is based on the prototype set. This initial population contains the prototype set itself, versions of the prototype set modified through a process of mutation and individuals generated at random.

The initial population is then evolved through a typical genetic algorithm process with fitness proportional selection, one point crossover, uniform mutation, and an elitist replacement strategy to obtain further and improved generations. The entire process is summarised in Figure 3. The classifier and the genetic algorithm work together towards achieving the same aim, i.e. every time the genetic algorithm needs to evaluate the fitness of an individual, its phenotype is obtained and used by the classifier to compute the fitness. The process normally stops when there is no further improvement for a number of generations, otherwise, it continues for a given number of generations.

## 3. Data Sets

Artificial data sets were created for the evaluation of the algorithm performance at the developing stage. These were bi-dimensional domains for easy graphical representation and understanding. Once the algorithm was fully developed, its performance was tested over actual data set form the UCI repository [9]. Table 1 shows a summary of the characteristic of the chosen domains. These were chosen to create a diverse set of databases with respect to the number of attributes and classes as well as to have balanced and not balanced databases with respect to the frequency of the classes.

The databases in Table 1 were first normalised and cleared off from unknown attribute values and then randomly ordered and separated into two sub-sets, one of them containing the 90% of the instances of the original database for training purposes, and the other with the remaining 10% for validation of the algorithm performance over unknown instance values. The training set was used by the classifier and the genetic algorithm to create the models and the

**Table 1. Actual domains from the UCI repository to be used in the experiments.**

| Domain | Instances | Atributes | Classes | Class Distribution |
|---|---|---|---|---|
| Balance scale | 625 | 4 | 3 | 288/49/288 |
| Glass | 214 | 9 | 6 | 70/76/17/13/9/29 |
| Thyroid | 215 | 5 | 3 | 150/35/30 |
| Wisconsin | 699 | 6 | 2 | 458/241 |

validation set was only used to validate the accuracy of the model which has never seen the instances within the validation set.

## 4. Experiments

Several experiments were carried out to validate the accuracy of the method. These consisted in two main parts. First a simple nearest prototype classifier that uses the euclidean distance as metric was generated as described in Section 2.1. The accuracy of this initial classifier was tested over the training and validation sets for the databases shown in Table 1. Then, each initial prototype classifier was evolved as described in Section 2.2 in order to re-distribute the prototypes within the euclidean space.

The initial population of the genetic algorithm consisted in 100 individuals, one of them was an exact copy of the prototype set generated by the classifier in the previous stage, 49 individuals were created from this prototype set through a process of mutation with a mutation rate of 5% at every gene, the remaining 50 individuals were generated at random to explore the search space. The fitness of an individual, or set of prototypes, was evaluated as the number of instances that it classifies with no error. The selection mechanism that selects parents for reproduction used fitness proportional probabilities. The genetic algorithm followed an elitist replacement strategy keeping the 10 best individuals of each generation. To create the 80% of the remaining 90 individuals, i.e. 72 individuals, one point crossover was performed, whereas the rest, i.e. 18 individuals, came from mutation of their parents. The mutation mechanism added a random number from a zero mean Gaussian distribution to each entry of the parent vector. The amount of mutation, which was proportional to the standard deviation of the distribution, was decreased at each new generation. This standard deviation started at one with the first generation and decreased to zero at the final step.

The evolution process for each database shown in Table 1 is presented in Figures 4 to 7. In the figures, the continuous line represents the fitness of the best individual at every generation whereas the dashed line represents the mean fitness of all individuals at every generation. It can be easily

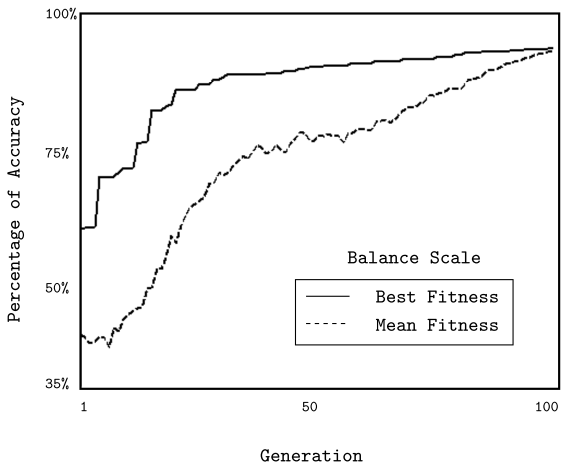

**Figure 4. Evolution of prototypes for the Balance Scale database.**

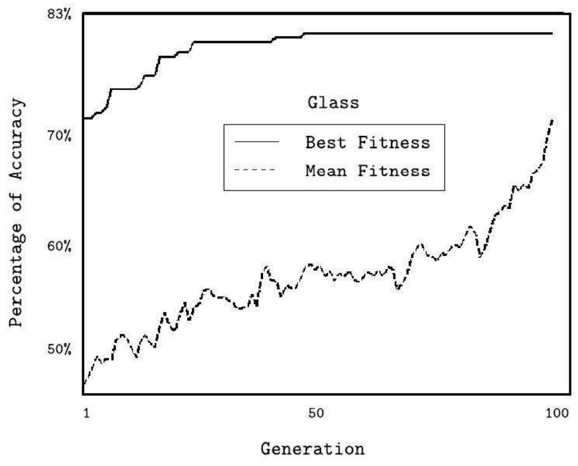

**Figure 5. Evolution of prototypes for the Glass database.**

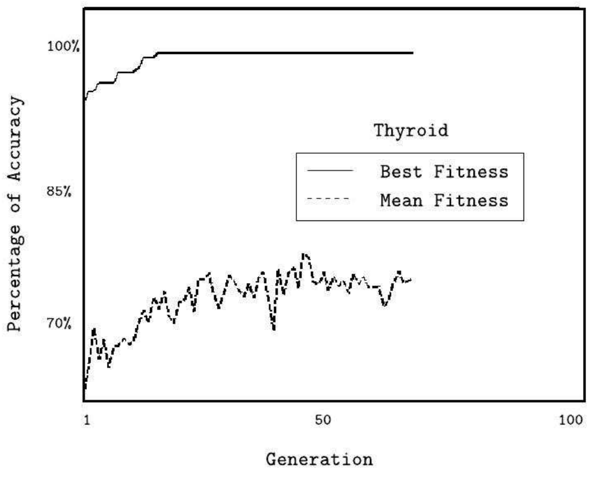

**Figure 6. Evolution of prototypes for the Thyroid database.**

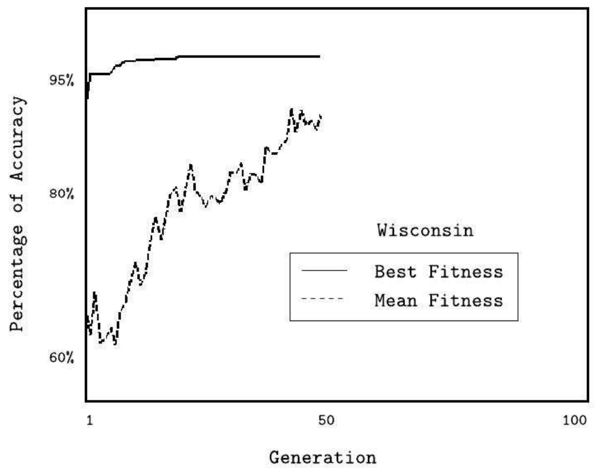

**Figure 7. Evolution of prototypes for the Wisconsin database.**

**Table 2. Classification accuracy rates of the algorithm over training and validation sets of the databases shown in Table 1.**

| Accuracy \ Data Set | BalanceScale | Glass | Thyroid | Wisconsin |
|---|---|---|---|---|
| Training Set Before Evolu. | 57.55% | 71.50% | 86.60% | 81.40% |
| Training Set After Evolu. | 92.18% | 80.83% | 98.45% | 97.62% |
| Valid. Set Before Evolu. | 41.94% | 66.67% | 85.71% | 75.71% |
| Valid. Set After Evolu. | 83.87% | 71.43% | 95.24% | 97.14% |

seen an improvement in the percentage of instances that the classifier is able to sort correctly as the generations go on. Note that some of the experiments terminated before reaching the maximum number of generations which was set to 100. The algorithm may stop if no further improvement is registered, there are a stall time limit, a stall generation limit and a fitness function tolerance limit that allow the algorithm to stop before reaching the maximum number of generations.

Table 2 shows the classification accuracy percentages of the proposed algorithm for each of the databases in Table 1 over the training and validation sets. For illustration, it also shows percentages of accuracy before and after the evolution stage of the algorithm. This is linked with Figures 4 to 7, the initial and final values of the best fitness plot match up with the accuracy rate values before and after the evolution stage over the training set. The results shown in Table 2 demonstrate that the initial set of prototypes is much less accurate than the evolved version of it over the validation set of the different databases, therefore, it can be said that in general the hybrid technique performs better than the classifier on its own. For instance the classifier itself has an accuracy rate of 41.94% when classifying the Balance Scale database, after the hybridisation the accuracy of the method has increased up to 83.87%, which is the most significant increase. It is worth mentioning the high accuracy rates of 95.24% and 97.14% obtained with this method when classifying the Thyroid and Wisconsin databases respectively.

The performance of different classification methods over the four databases uses in this work is summarised in Table 3. It can be seen that the results obtained with the proposed method of evolved prototypes outperforms the classification accuracy rates of most traditional methods. Also, when no improvement was observed with respect to these methods the accuracy of the evolved prototypes was still quite high.

**Table 3. Comparative results of different classification methods over the databases shown in Table 1.**

| Algorithm \ Data Set | BalanceScale | Glass | Thyroid | Wisconsin |
|---|---|---|---|---|
| IBK K=1 | 82.42% | 71.99% | 97.21% | 95.14% |
| IBK K=3 | 80.26% | 70.58% | 93.46% | 96.42% |
| LVQ | 85.10% | 62.56% | 91.03% | 95.88% |
| J48 (C4.5) | 77.82% | 72.86% | 92.06% | 94.70% |
| Evolved Prototypes | 83.87% | 71.43% | 95.24% | 97.14% |

## 5. Conclusion

In many real classification problems it is usual to find domains that may be described by a few prototypes instead of using the whole set of instances. This results in a quicker classification as the classification task requires a smaller number of comparisons. In this paper a new method that combines nearest prototype classification and genetics has been presented. At a first stage, the nearest prototype classifier generates the prototypes whose spatial location will be evolved at a second stage through the use of a genetic algorithm. A number of experiments were carried out over artificial and real domains in order to validate the algorithm. From the results of the experiments, it has been proved effective to use genetics in order to improve the classification accuracy of the initial set of prototypes. The classification accuracy has been improved in all of the actual domains. In two domains the accuracy of this method is over 95% and in a third one it is almost 84% which demonstrates the ability of the algorithm to classify real data.

One of the disadvantages of this method is that the number of prototypes is fixed by the classifier at the initial stage and it does not evolve with the second stage, the genetic algorithm. Allowing the genetic algorithm to increase the number of prototypes may improve the classification accuracy. Thus, this is considered by the authors as the main future line of research in this topic.

## References

[1] J. C. Bezdek and L. I. Kuncheva. Nearest neighbour classifier designs: An experimental study. *Int. J. Intell. Syst.*, 16:1445–1473, 2001.

[2] L. I. Kuncheva and J. C. Bezdek. Nearest prototype classification: Clustering, genetic algorithms, or random search? *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, 28:160–164, 1998.

[3] T. Kohonen. *Self-organization and associative memory*. Springer-Verlag. Berlin, Germany, 3rd edition, 1989.

[4] S. Seo and K. Obermayer. Soft learning vector quantization. *Neural Comput.*, 15:1589–1604, 2003.

[5] F. Fernandez and P. Isasi. Local feature weighting in nearest prototype classification. *Neural Networks, IEEE Transactions on*, 19:40–53, 2008.

[6] D. Aha and R. Goldstone. Concept learning and flexible learning. *Proc. 14th Annu. Conf. Cogn. Sci. Soc.*, pages 534–539, 1992.

[7] F. Fernandez and P. Isasi. Evolutionary design of nearest prototype classifiers. *J. Heuristics*, 10:431–454, 2004.

[8] C. Chou, C. Lin, Y. Liu, and F. Chang. A prototype classification method and its use in a hybrid solution for multiclass pattern recognition. *J. Pattern Recognition*, 39:624–634, 2006.

[9] A. Asuncion and D Newman. UCI machine learning repository. 2007.