ESANN'2001 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 25-27 April 2001, D-Facto public., ISBN 2-930307-01-3, pp. 171-178

Designing Nearest Neighbour Classifiers by the Evolution of a Population of Prototypes

Fernando Fernández and Pedro Isasi

Universidad Carlos III de Madrid.
Avda de la Universidad 30. 28911, Leganés. Madrid (Spain)
ffernand@scalab.uc3m.es isasi@ia.uc3m.es

Abstract. A new evolutionary algorithm to design nearest neightbour classifiers is presented in this paper. Main design topics of this sort of classifiers are the number of prototypes used and their position. This algorithm is based on the evolution of a population of prototypes that try to achieve an equilibrium by finding the right size of the population and the position of each prototype in the environment, solving at the same time both design topics above. A biological point of view is given to explain most of the concepts introduced, as well as the operators used in evolution.

Keywords: Classifier design, nearest neighbour classifiers, evolutionary learning.

1 Introduction

Nearest Neighbour Classifiers are defined as the sort of classifiers that assign to each new unlabeled example, v, the label of the nearest prototype r_i from a set of N different prototypes previously classified [11]. The design of this classifiers is difficult, and rely in the way of defining the number of prototypes needed to achieve a good accuracy, as well as the initial set of prototypes used.

Many discussions about what is the right technique to use can be found in the literature [6]. Some clustering approaches [13, 12, 1] are based in two main steps. The first one is to cluster a set of unlabeled input data to obtain a reduced set of prototypes. The second step is to label these prototypes basing on labeled examples and the nearest neighbour rule.

Neural networks approaches are also very common in the literature, like the LVQ algorithm [5] and the works of other authors with radial basis functions [4]. To find the right number of neurons of the net, two basis approaches can be found. On one hand, some techniques try to introduce or eliminate prototypes (or neurons) while designing the classifier following different heuristics, as the average quantization distortion [13] or the accuracy in the classification [10].

ESANN'2001 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 25-27 April 2001, D-Facto public., ISBN 2-930307-01-3, pp. 171-178

On the other hand, typical approaches try to define first the optimal size of the classifier and after to learn it using this value. Genetic algorithms approaches are typically used to find an initial set of prototypes, as well as its right size, in addition to another technique to achieve local optimization [9]. In [6], genetic algorithms are used, as well as random search to definitively find the right set of prototypes. In [14], an evolutionary approach is presented based in the R^4 rule (recognition, remembrance, reduction and review) to evolve nearest neighbour multilayer perceptrons.

In this work, an evolutionary algorithm is introduced to dinamically define the number of prototypes of the classifier as well as the location of these prototypes basing in a biological description of the problem. Thus, the classifier is defined as a population of animals (prototypes) that must fight to eat vegetables (training examples) that allows them to survive and to find an equilibrium in the environment (optimum number of prototypes). So, the evolution will allow the individuals to locate themselves in the right position, and to be labeled in the right way, achieving the equilibrium only when the right number of prototypes is achieved.

In the next section, the algorithm is explained, as well as the main concepts that are used. Section 3 shows principal experiments performed and a comparision with previous works, while section 4 shows main conclusions achieved and further research.

2 Evolutionary Design of Nearest Neighbour Classifiers (ENNC)

The ENNC algorithm offers an evolutionary point of view to the design of nearest neighbour classifiers. The main advantages of this method is that neither the number of prototypes used, nor an initial set of prototypes are required. The first difference among this algorithm and previous evolutionary approaches is the way of representing the population: in this case, and following the Mitchigan approach, each chromosom represents only one prototype, and not a whole classifier, so the classifier is represented by the whole population. The main concepts can be defined as follow:

Classifier/Population, C. A set of N prototypes $C = \{r_1, \ldots, r_N\}$.

Prototype/Animal, r_i Each prototype is composed by the localization of the prototype in the environment and the class (label) of the prototype.

Region, r_i**.** The environment is divided in a set of N regions. Each animal only eats vegetables in each own region. The region of each animal is defined by the position of the animal and the nearest neighbour rule.

Class/Specie, s_j**.** Both animals and vegetables belogns to a class or specie from the set $S = \{s_1, \ldots, s_L\}$. The goal of an animal r_i of specie s_j is to

ESANN'2001 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 25-27 April 2001, D-Facto public., ISBN 2-930307-01-3, pp. 171-178

eat as vegetables of class s_j as possible and not to eat vegetables of other classes $s_k \neq s_j$.

Pattern/Vegetable, v_r Is each one of the patterns or examples that will be used for training or testing the system. They are considered as vegetables of the biological system.

Quality/Health of a prototype/animal Is a weighted relationship among the local performance of the prototype and the performance of the other prototypes.

Second main difference of this algorithm with previous evolutionary approaches comes from the operators that are used to evolve. In this case, most of the operators are based on heuristics of previous works, and new ones have been incorporated. So the learning phase is an iterative process that execute several operators over each individual. Each of this iteration is called a year in the animals life, and the year is divided in four seasons: spring, summer, fall and winter. In each season, differents operators are executed, and are summarized in table 1.

Season	Operators	Description
Spring	Mutation	Each animal change its own specie to the majoritary specie of vegetables in its region
Summer	Reproduction	The animals reproduce to create animals that eat what they do not want to eat
Fall	Fight and Move	The animals fight against other animals and move to a different position to get more food
Winter	Die	Weak animals die

Table 1: Phases of the algorithm and operators used in each phase

Another important issue is that this division in different iterations allows to use different trainning patterns in each iteration, as well as different test patterns. In this sense, the quantity of patterns used for trainning and for test is defined by the user. The inizialization of the algorithm and the different seasons and operators are explained in the following.

Initializing. To define the initial population, two possibilities have been taken into account. The first one is to start with only one animal, which is the first individual of the population. The second one is to start with one animal for each input pattern. In this sense, the problem of the initial set of prototypes is solved, and both initialization ways will be discussed in the experiments.

Spring. The spring season is the time when the vegetables born. All the animals are placed in its own region, and will recollect all the vegetables in its region. The way to define if a vegetable belongs to an animal or other is based on the nearest neighbour rule.

At the end of spring, each animal knows the quantity of vegetables of each specie that it can eat, so it will become to the specie of the most abundant

ESANN'2001 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 25-27 April 2001, D-Facto public., ISBN 2-930307-01-3, pp. 171-178

specie of vegetables. This operator correspond with the labeling phase of the unsupervised learning approaches[1, 12], but in this case, the supervision is included in each iteration and not only in a posterior phase. This operator is called mutation operator.

Summer. Summer is the season where animals reproduce (second operator). In this case the reproduction is asexual, and an animal only reproduce if it needs another animal that eats what it does not want to eat, so it has a selfish motivation. In a neural network domain, reproduction is equivalent to the insertion of new neurons in the net based on the accuracy of the classifier [4].

So an animal only reproduce if into its region, vegetables of different classes are found. The probability of reproduction is proportional to the difference among the number of vegetables of each class in its region. Newborn animal is located in order to increase the ancestor performance.

Fall. Fall is the time where food starts to scarce, and the animals decide to look for more food. In this sense, fall have two phases. In the first one, animals can fight among them, in order to steal territory to other animals and to get more food (third operator). In the second phase, animals locate themself in an optimum place to spend the winter and to wait the next spring (fifth operator).

1. Fights: An animal can decide to fight with other animals in order to get more food. Fight operator is executed for each animal, and have the following phases:
   (a) To choose a rival by assigning probalities proportionals to the distance to the rest of animals and using a roulette as a selection method.
   (b) Once the rival is selected, the animal have to decide whether to fight or not. The probability of fighting is proportional to the difference in health of both rivals.
   (c) Once the rival has been selected and the animal decides to fight, there are 2 possibilities:
      i. The animals does not belongs to the same specie. In this case, it have not sense to fight, and both animals sign an agreement, in the way that the second one gives the vegetables required to the first one.
      ii. Both animals belongs to the same specie. Animals fight, with a probability of victory proportional to the animals health. The winner steals food to the looser, and if it is allowed to steal all the food to the looser, the looser die.
2. Move: The move operator implies to relocate each animal in the best expected place for spend the winter and wait for the following spring. So each animal decides to move to the centroid of the vegetables of its same class. This operator is based on the Lloyd iteration of the GLA algorithm [7].

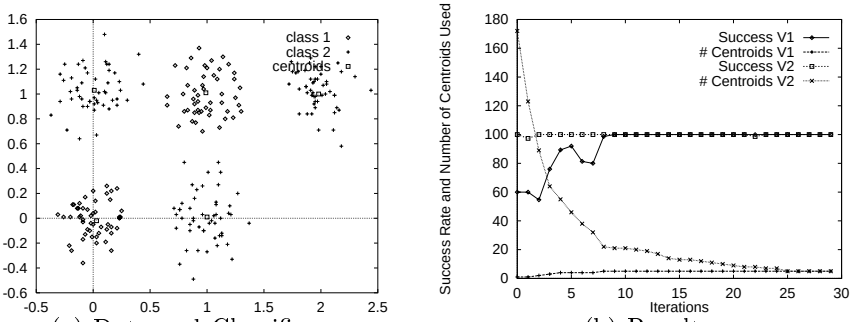
ESANN'2001 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 25-27 April 2001, D-Facto public., ISBN 2-930307-01-3, pp. 171-178

Winter. In winter, weak animals die. Probability to die is 1 minus the double of the health. Then, healthfull animals will survive with probability of 1, while weak animals with health of less than 0.5 might die. In the neural network bibliography, an extense documentation about which neurons to select in order to simplify the network structure can be found [4, 13]. At the end of this season, all vegetable disappear.

3 Experiments and Results

3.1 Simple Gaussian Distributed Data

In this experiment, two different classes are defined following the distributions shown in figure 1(a). We have applied the ENNC learning algorith in its two versions. First, starting with only 1 centroid (population of size 1), and second using as centroids as the number of input patterns. In both cases, the 30% of the data is used for testing following a cross validation scheme. The results of both approaches are shown in figure 1(b).



(a) Data and Classifier (b) Results

Figure 1: (a) Data with Simple Gaussian Distributions and the Centroids of a classifier obtained (b)Success obtained and prototypes used for each Iteration.

We can see how in the first case, the number of centroids is increased to five. In the second case, the number of centroids is also reduced to five, so both versions achieve the same number of centroids. Furthermore, a 100% of succes is obtained in both cases. Note, that the first vesion achieves the objective faster than the second, and only needs 8 iterations to get better results. For the second version, 25 iterations are need. This delay respect the first version is due to the high number of centroids to eliminate. An example of the classifier found is given in figure1(a), showing the centroids located in the mean of the distributions.

ESANN'2001 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 25-27 April 2001, D-Facto public., ISBN 2-930307-01-3, pp. 171-178

3.2 Iris Data Set

Iris Data Set from UCI Machine Learning Repository [1] [3] is used in the second experiment. This dataset consits of 150 samples of three classes, where each class has 50 examples. The dimension of the feature space is 4. In this case, and for comparision reasons, the whole data set was used for training and for testing.

The results of aplying both versions of the ENNC algorithm are shown in figure 2(a), where the number of prototypes and the success achieved are shown. Two experiments have been perfored, one initializing the population with only one prototype (version 1) and another initializing the population with as prototypes as number of input examples (version 2). Both versions converges to classifiers of 3 or 5 centroids, osciling among them without to decide which one is better. 5 centroid classifier achieves the best results, 98% of success, while 3 centroids classifier achieve a 91% of success. As in previous experiment, convergence of the ENNC algorithm is faster when an initial classifier of only one centroid is used instead to use as initial classifier the whole data set, and in both cases, less than 150 iterations were needed to find the better results. The decission about what classifier is better, the one of 3 prototypes and 91% of accuracy, or the one of 5 prototypes and a 98% of accuracy, is a user decision.

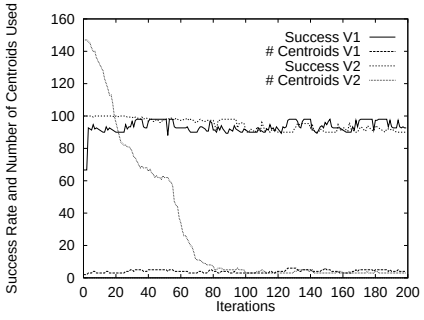
These results are compared with the ones presented in [2, 6, 8], and are summarized in table 2(b), where the number of prototypes and the misclassifications are shown. For the ENNC algorithm, different results are extracted from different moments of the population evolution for both inizializing ways. We can see how ENNC algorithm improves the results of MFCM-3, LVQ and GLVQ-F, but can not achieve the results of the improved PNN, that has one misclassification with only 3 prototypes.

4 Conclusions and Further Research

The ENNC algorithm for the design of nearest neighbour classifiers has been exposed in this work, and some experimental results in well-known domains, as well as their comparisions with different works from the literature have been shown. In this sense, good results have been achieved, improving the results of most of the algorithms in most of the domains. Anyway, the real improvement of ENNC is not only the capability to achieve good results, but the facility these results are achieved. Most of the algorithms need a predefined number of prototypes to use, and the initial location of these prototypes. In this algorithm, no initial conditions must be introduced, given that both version of the algorithm converges to the same solutions. Furthermore, most of the methods found in the literature to solve this problem are based in how to generate these initial parameters in order to locally optimize with another technique, i.e. they use a known technique and try to improve it. In our

[1] http://www.ics.uci.edu/~mlearn/MLRepository.html

ESANN'2001 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 25-27 April 2001, D-Facto public., ISBN 2-930307-01-3, pp. 171-178



(a) Evolution of the populations

Algorithm	Prot.	Mis.
MFCM-3	7	8
LVQ	7	3
LVQ	3	17
GLVQ-F	8	3
DR	5	3
DR	3	10
Improved PNN	3	1
ENNC (version 1)	5	3
ENNC (version 1)	4	9
ENNC (version 1)	3	11
ENNC (version 2)	82	0
ENNC (version 2)	5	3
ENNC (version 2)	3	11

(b) Comparisions

Figure 2: (a)Success obtained and prototypes used for each Iteration. (b)Comparisions with previous works

case, all the goals are solved and integrated in the algorithm itself, and no additional technique is needed to solve any other problem.

However, there are different aspects that have not been handle yet. The main one is about convergence. We have shown how the algorithm does not converge totaly in complex domains, neither to a defined number of centroids, nor to a defined success. This is due to the ability of the algorithm to escape from a local minimum. For instance, the algorithm is able to escape from a situation of a 100% of success in order to find less size classifiers, and vice versa, it is able to escape from a small size classifier introducing new centroids in order to improve the accuracy. There are several mechanisms that could be used to decide when to stop learning. The single one is to let the algorithm run a long number of iterations, and after, review the results. The problem of this approach is that, at first, the user does not know the number of iterations that the algorithm will need to achieve good results. Another way is to define several user goals, so the algorithm works until it achieves these goals. The problem of this approach is that maybe the algorithm might find better solutions in a low time. Thus, an extensive work about when to stop the algorithm is required.

Acknoledgments

The author would like to thank Inés Galván and Ana M^{a} Iglesias for their main contributions on this work. This research has been partially supported by a grant of the Ministry of Education, Culture and Sport of Spain.

ESANN'2001 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 25-27 April 2001, D-Facto public., ISBN 2-930307-01-3, pp. 171-178

References

[1] Sergio Bermejo and Joan Cabestany. A batch learning algorithm vector quantization algorithm for nearest neighbour classification. *Neural Processing Letters*, 11:173–184, 2000.

[2] James C. Bezdek, Thomas R. Rechherzer, Gek Sok Lim, and Yianni Attikiouzel. Multiple-prototype classifier design. *IEEE Transactions on Systems, Man and Cybernetics*, 28(1):67–79, February 1998.

[3] C. L. Blake and C. J. Merz. UCI repository of machine learning databases, 1998.

[4] Bernd Fritzke. Growing cell structures -a self-organizing network for unsupervised and supervised learning. *Neural Networks*, 7(9):1441–1460, 1994.

[5] Teuvo Kohonen. *Self-Organization and Associative Memory.* Springer, Berlin, Heidelberg, 1984. 3rd ed. 1989.

[6] Ludmila I. Kuncheva and James C. Bezdek. Nearest prototype classification: Clustering, genetic algorithms, or random search? *IEEE Transactions on Systems, Man and Cybernetics*, 28(1):160–164, February 1998.

[7] S. P. Lloyd. Least squares quantization in pcm. In *IEEE Transactions on Information Theory*, number 28 in IT, pages 127–135, March 1982.

[8] K. Z. Mao, K.-C. Tan, and W. Ser. Probabilistic neural-network structure determination for pattern classification. *IEEE Transactions on Neural Networks*, 11(4):1009–1016, July 2000.

[9] J. J. Merelo, A. Prieto, and F. Morán. Optimization of classifiers using genetic algorithms. In Patel Honavar, editor, *Advances in Evolutionary Synthesis of Neural Systems.* MIT press, 1998.

[10] J. C. Pérez and Enrique Vidal. Constructive design of LVQ and DSM classifiers. In J. Mira, J. Cabestany, and A. Prieto, editors, *New Trends in Neural Computation*, volume 686 of *Lecture Notes in Computer Science.* Springer Verlag, 1993.

[11] Peter E. Hart Richard O. Duda. *Pattern Classification and Scene Analysis.* John Wiley And Sons, 1973.

[12] Nikhil R.Pal, James C. Bezdek, and Eric C.-K. Tsao. Generalized clustering networks and kohonen's self-organizing scheme. *IEEE Transactions on Neural Networks*, 4(4):1993, July 1993.

[13] M. Russo and G. Patanè. ELBG implementation. *International Journal of Knowledge based Intelligent Engineering Systems*, 2(4):94–109, April 2000.

[14] Q. Zhao and T. Higuchi. Evolutionary learning of nearest neighbour MLP. *IEEE Transactions on Neural Networks*, 7(3):762–767, 1996.