Evolutionary Approach to Overcome Initialization Parameters in Classification Problems

P. Isasi,F. Fernandez

Computer Science department
Universidad Carlos III de Madrid

Abstract The design of nearest neighbour classifiers is very dependent from some crucial parameters involved in learning, like the number of prototypes to use, the initial localization of these prototypes, and a smoothing parameter. These parameters have to be found by a trial and error process or by some automatic methods. In this work, an evolutionary approach based on Nearest Neighbour Classifier (ENNC), is described. Main property of this algorithm is that it does not require any of the above mentioned parameters. The algorithm is based on the evolution of a set of prototypes that can execute several operators in order to increase their quality in a local sense, and emerging a high classification accuracy for the whole classifier.

1 INTRODUCTION

Nearest Neighbour Classifiers are defined as the sort of classifiers that assign to each new unlabelled example, $v_r \in V = \{v_1, \ldots, v_M\}$, the label of the nearest prototype, r_i, from a set, $C = \{r_1, \ldots, r_N\}$, of N prototypes previously classified [DH73]. The set of different labels or classes is $S = \{s_1, \ldots, s_L\}$.

The design of these 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 and, usually, a smoothing parameter. Many discussions about what is the right technique to use can be found in the literature. Some approaches based on clustering techniques are divided in two main steps. The first one is to cluster a set of unlabelled input data to obtain a reduced set of prototypes, for instance, with the LBG algorithm. The second step is to classify these prototypes basing on previosly labelled examples and the nearest neighbour rule. Although this approach produces good results, it is obvious that to introduce information about the classification performance in the location of the prototypes seems to be needed to achieve a higher performance.

Neural networks approaches are also very common in the literature, like the LVQ algorithm [Koh84] and the works with radial basis functions. Some techniques try to introduce or to eliminate prototypes (or neurons) while designing the classifier following different heuristics, as the average quantization distortion or the accuracy in the classification. Other approaches try to define first the optimal size of the classifier and then to learn it using the previous value.

In this work, an evolutionary approach called Evolutionary Nearest Neighbour Classifier (ENNC), is introduced to dynamically define the number of prototypes of the classifier as well as the location of these prototypes.

2 THE ENNC ALGORITHM

2.1 Architecture

The system can be represented by a bidimensional matrix, where each row is associated to a region $r_i \in C$, and each column is associated to a class $s_j \in S$. Each position (i, j) of the matrix is a structure that contains features about the set of training examples that belongs to the region r_i and to the class s_j. These sets are called Region-Class Sets, and notated by V_{ij}. Furthermore, information about the elements that belong to a defined class (set VS_j), and that belong to a defined region (set VR_i) are stored. Last, some information can be obtained from the set V. Note that the structure does not store the whole region, class, and region-class sets, but only some features about them, as their size, or their centroid, that will be defined below. For all the sets, its membership function is given too.

Figure 1 shows the architecture used to keep all this information. Each of those nodes stores the features about the sets that represent. This structure is dynamic, in the sense that if any set is empty, the node is not created, solving some memory requirements in complex domains.

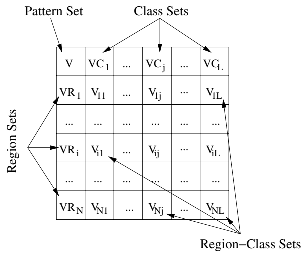

Figure1. ENNC Architecture

From this architecture some important features are computed:

- $regions_{s_j}$ is the number of regions which prototype class is s_j.
- $expectation_{s_j}$ is the number of patterns that any prototype r_i of the class s_j is expected to correctly classify. This number is computed in a relative way, by the relationship among the number of patterns and the number of regions of each class s_j:

$$expectation_{s_j} = \frac{||VS_j||}{regions_{s_j}} \tag{1}$$

- $location_{r_i}$ is the location of the prototype r_i.
- $class_{r_i}$ is the class of the prototype r_i.
- $accuracy_{r_i}$ is the classification accuracy of the prototype. It is calculated as shown in equation 2, being $s_i = class_{r_i}$.

$$accuracy_{r_i} = \frac{||V_{is_i}||}{||VR_i||} \tag{2}$$

– $apportation_{r_i}$ is a relationship among the number of patterns correctly classified by this prototype and the number of patterns that it is suppose to classify:

$$apportation_{r_i} = \frac{||V_{is_i}||}{\frac{expectation_{s_i}}{2}} \tag{3}$$

– $quality_{r_i}$ is the quality of the prototype, calculated from equation 4. This value is a relationship among the accuracy of the prototypes, and its contribution to the classification of the whole input pattern set.

$$quality_{r_i} = max(1, accuracy_{r_i} * apportation_{r_i}) \tag{4}$$

– $neighbours(r_i)$ is the set of regions that has a common border with the region r_i.

2.2 The Algorithm

The learning phase is an iterative process where the prototypes can execute several operators, once the features defined in section 2.1 have been computed. This operators are heuristics that allow the prototypes to change their location, to introduce new prototypes, etc. The algorithm begins with an initialization where the classifier is composed by only one prototype. After, the classifier evolves by executing, in a loop, all the designed operators, described below. These operators take advantage of some information gathered at the beginning of the loop.

Initialization One relevant feature of our method is the absolute elimination of initial conditions. These initials conditions are usually summarized in three: the number of prototypes, the initial set of prototypes and a smoothing parameter. The ENNC algorithm allows to learn without those parameters, given that the initial number of prototypes is always one, the initial location of that prototype is not relevant, and there are no learning parameters.

Mutation Operator The goal of this operator is to label each prototype with the most populate class in each region. Once the features are obtained, each prototype knows the number of patterns of each class located in its region, so the prototype changes, if needed, and becomes to the same class of the most abundant class of patterns in its region.

This way to get the main class is typically used when unsupervised learning is applied to supervised classification. In these works, the algorithms typically generate a set of clusters that takes into account only the distribution of the data. In a second phase, the clusters are labelled to the most populate class in the cluster. However, in this work, the supervision is included in each iteration of the algorithm, and not only in "a posteriori" phase. The formulation of this operator is defined in equation 5.

$$\forall r_i \in C, class_{r_i} = \arg_j \max ||V_{ij}|| \tag{5}$$

where $||V_{ij}||$ is the size of the set V_{ij}.

Reproduction Operator The goal of this operator is to introduce new prototypes in the classifier. The insertion of new prototypes is a decision that is taken by each prototype, in such a way that all of them contains in its region as much patterns as possible of the same class. The regions with patterns belonging to different classes, can create new regions containing the patterns of a different class to the class of the prototype.

Each prototype, r_i, of the class s_j, executes a roulette proportional to the number of elements of the set $V_{ij'}$ to which it represents. The possible results of the roulette are two. On on hand, if the resulting set $V_{ij'}$ is the set V_{ij}, i.e. $j = j'$, no reproduction is executed. On the other hand, if the resulting set $V_{ij'}$ is not the set V_{ij}, i.e. $j \neq j'$, the reproduction is executed, and a new region $r_{i'}$ is created to contain the patterns in $V_{ij'}$, that is renamed to be $V_{i'j'}$.

Fight Operator This operator provides the prototype the capability of getting patterns from other regions. Formally, this operator allows a prototype, r_i, to modify its sets V_{ij} from the sets $V_{i'j}$ of another prototype $r_{i'}$, for $i \neq i'$. This operator takes place in several steps:

1. Choose the prototype $r_{i'}$ against to fight proportionally to the difference between its quality and the quality of a neighbor r_i.
2. Decide whether to fight or not proportionally to the distance of their qualities:

$$P_{fight}(r_i, r_{i'}) = |quality_{r_i} - quality_{r_{i'}}| \tag{6}$$

3. If prototype r_i decides to fight with prototype $r_{i'}$, there are two possibilities. Given $class_{r_i} = s_i$, and $class_{r_{i'}} = s_{i'}$:
   - If $s_i \neq s_{i'}$ (cooperation). Both prototypes belongs to different classes. In this case, the prototype $r_{i'}$ will give to the prototype r_i the patterns of the class s_i.
   - If $s_i = s_{i'}$ (competition). In this case, patterns can be transfered from set $V_{i's_i}$ to the set V_{is_i}, or vice versa, depending on who wins the fight. The amount of patterns that are transferred depends on a probability proportional to the qualities of both prototypes.

Move Operator The move operation implies to relocate each prototype in the best expected place. So each prototype, r_i, decides to move to the centroid of the set V_{ij}, being $class_{r_i} = j$, i.e. it takes as centroid the centroid of the set V_{ij} of its class, as shown in equation 7.

$$location_{r_i} = centroid_{V_{is_i}} \tag{7}$$

where $s_i = class_{r_i}$.

This operation based in the second step of Lloyd iteration allows to make a local optimization of the classifier, increasing its performance.

Die Operator The probability for a prototype of being eliminated is 1 minus the double of the quality, as defined in equation 8. In that case, successful prototypes will survive with probability of 1, while useless prototypes with quality less than 0.5 might die. A wide range of heuristics about how to reduce the number of prototypes in a classifier can be found in the bibliography.

$$P_{die}(r_i) = \begin{cases} 0, \text{ when } quality_{r_i} > 0.5 \\ 1 - 2 * quality_{r_i}, \text{ when } quality_{r_i} \leq 0.5 \end{cases} \quad (8)$$

3 Experiments

This section shows the experiments performed over two different domains, where the ENNC algorithm is compared with other classification algorithms, as C4.5 [Qui93], decission rules [FW98], Naive Bayes [DH73], and IBK [AK91], for values of $k = 1$ and $k = 3$, executed in WEKA [WF00].

3.1 Spiral Data Set

Spiral data set is typically used in the bibliography as a challenge data set, where two interlazed spirals must be correctly classified, as it is shown in figure 2. Examples in the same spiral belong to the same class, and there are 500 examples in each spiral. From the whole set, 900 examples was used for learning, and 100 for test.

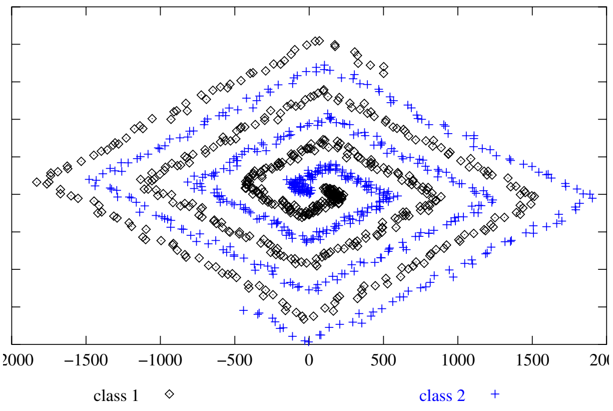

Figure2. Spiral Data Set

Given that the ENNC algorithm has an important stochastic component, the experiment has been executed 20 times, each of them of 300 iterations. The results are summarized in Table 1 where, for each execution, the iteration when the winner classifier was evolved is shown, as well as the number of prototypes of the classifier and the classification error for training and test sets. The table shows that the 99% of success is typically achieved on training, while success on test set ranges from 94% to 98%, what means a maximum difference of 4 errors over the test set. The number of prototypes of the classifiers achieved in each iteration is close to 80.

Iteration	Prototypes	Training (%)	Test (%)	Iteration	Prototypes	Training (%)	Test (%)
289	82	99,33	97,0	281	73	98,9	96,0
158	75	99,0	96,0	300	84	99,44	97,0
174	81	99,11	95,0	123	84	99,56	96,0
267	81	99,33	96,0	81	79	99,33	94,0
282	81	99,44	97,0	260	82	99,44	96,0
260	78	99,11	97,0	131	77	98,67	95,0
170	79	99,56	96,0	81	81	99,33	98,0
182	82	99,33	97,0	97	79	99,33	96,0
120	79	98,67	95,0	299	78	99,56	96,0
291	84	99,78	97,0	179	75	99,0	97,0

Table1. Results of different executions of the ENNC algorithm over spiral data set.

Table 2 shows comparative results of different classifiers over this domain, showing the best and the lowest results of the ENNC algorithm over the 20 executions. The same training and test sets was used with all the classifier systems. Results for C4.5, decission rules (PART) and Naive Bayes are poor, and they are far from the solutions obtained by ENNC. However, IBK achieves the best results of a 100% of success as is expected in a domain with the examples of different classes very separated, as shown in Figure 2.

C4.5	PART	Naive Bayes	IBK ($k = 1$)	IBK ($k = 3$)	ENNC (best)	ENNC (worst)
62	56	50	100	100	98	94

Table2. Comparative results over the spiral data set.

Figure 3 shows the evolution of one execution of the ENNC algorithm, with the number of prototypes and success obtained in each iteration over the training and the test sets. Figure shows that in only 25 iterations, the number of prototypes grows up to 64, obtaining a success of the 91,34% over the training set, and of the 85% over the test set. However, the number of prototypes still grows up to the range 70-80, when the search of better solutions go on, obtaining the best solution in iteration 176, with a success of 94% over the test set.

3.2 Uniformly Distributed Data

In this experiment, the ENNC algorithm has been executed over the data set shown in the Figure 4, as defined in [Bur91]. As in that work, data set contains 2000 instances of each class, using 500 for training ant 1500 for test.

As in previous case, the ENNC algorithm is executed 20 times, with a mamimum length of 300 iterations. The algorithm achieves a solution of only 2 prototypes in 16 of the 20 executions, achieving a 98.6% iver the training and test sets. Other executions achieve better results for classifiers with more prototypes, arriving up to the 98.73% with 12 prototypes. Table 3 shows some comparative results with previous approaches, as LVQ [Koh84] and LVQ-PNN [Bur91] of different sizes, and a different version of

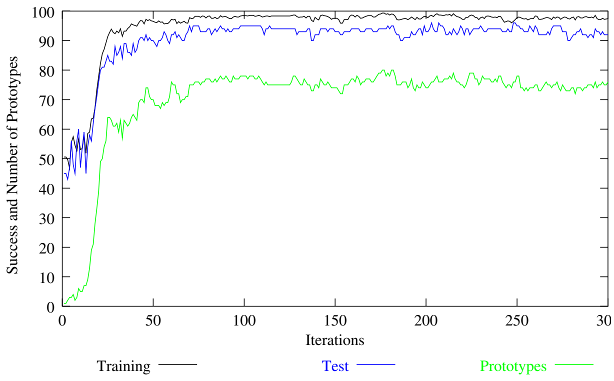

Figure3. Evolution of one execution of the ENNC algorithm over spiral data set.

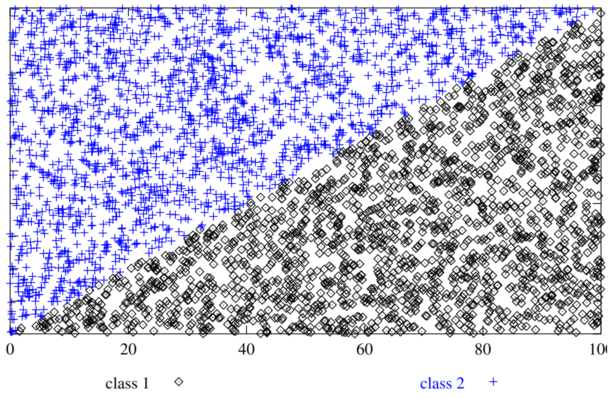

Figure4. Uniformly Distributed Data.

PNN [MTS00] that automatically defines the number of neurons. Furthermore, C4.5, PART, Naive Bayes and IBK (experimented with the default values defined in WEKA) are compared too.

The table shows that the worst result obtained by ENNC for two prototypes (98.6%) is very similar to the best solutions obtained with other different approaches, while the best solution obtained, with a 98.93% of success, is better than the rest of solutions reported.

4 CONCLUSIONS

The algorithm presented in this work is an evolutionary approach to solve the problem of finding a set of prototypes that are able to correctly classify the examples of a domain, following a 1-nearest neighbour approach. The main advantage of this method are, on

Algorithm	Data	Success
LVQ	10 prototypes	96.13
LVQ	100 prototypes	97.77
LVQ-PNN	10 Neurons	96.99
LVQ-PNN	100 Neurons	98.17
PNN (Mao)	8 Neurons	98.85
C4.5	-	96.97
PART	-	96.6
Naive Bayes	-	96.77
IBK	$k = 1$	98.7
IBK	$k = 3$	98.93
ENNC (best)	2 Prototypes	98.93
ENNC (worst)	2 Prototypes	98.6

Table3. Comparative results over Uniformly Distributed Data.

one hand, that it is able to achieve a high accuracy in most of the domains where it has been tested, even compared with other techniques from the literature. On the other hand, the achievement of these good results is done without the user defines the initial conditions for learning.

References

[AK91] D. Aha and K. Kibler. Instance-based learning algorithms. *Machine Learning*, 6:37–66, 1991.

[Bur91] Pietro Burrascano. Learning vector quantization for the probabilistic neural network. *IEEE Transactions on Neural Networks*, 2(4):458–461, July 1991.

[DH73] Richard O. Duda and Peter E. Hart. *Pattern Classification and Scene Analysis*. John Wiley And Sons, 1973.

[FW98] Eibe Frank and Ian H. Witten. Generating accurate rule sets without global optimization. In *Proceedings of the Fifteenth International Conference on Machine Learnin*, 1998.

[Koh84] Teuvo Kohonen. *Self-Organization and Associative Memory*. Springer, Berlin, Heidelberg, 1984. 3rd ed. 1989.

[MTS00] 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.

[Qui93] J. Ross Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993.

[WF00] I. H. Witten and E. Frank. *Data Mining. Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufmann, 2000.