Global and Partial Function Approximation with Evolutionary Algorithms

Carlos Kavka, Patricia Roggero
LIDIC
Departamento de Informática
Universidad Nacional de San Luis
Ejército de los Andes 950
(5700) San Luis - Argentina

Marc Schoenauer
C.M.A.P.
Centre de Mathématiques Appliqués
Ecole Polytechnique
91128 Palaiseau Cedex - France

Keywords: Evolutionary computation, Neural networks, Voronoi diagrams

Abstract

We present an evolutionary algorithm that evolves a population of local approximators in order to fit an unknown function. The evolutionary algorithm performs a simultaneous learning of simple local approximators together with the regions in which the local approximators are applied. By combining these simple local approximators, the domain is in fact partitioned in the Voronoi diagram that has as centers, the center points of the region in which each local approximator is efficient and useful. Both continuous and non-continuous approaches are considered. The algorithm seems promising in order to develop good neural networks for function approximation.

1 Introduction

Function approximation can be carried by global or local methods. The back propagation algorithm and the RBF algorithm are examples of a global and a local learning algorithm respectively [5].

The standard back propagation algorithm, used in the neural networks community, is an example of a global approximation method. The algorithm tries to find a set of parameters (weights) that are combined in a global function defined in terms of sigmoid (and may be linear) functions. The performance of the algorithm is acceptable, except when the function is not continuous.

The RBF neural network consists of a set of units that performs a local based function approximation, making the method faster and adequate for non continuous function approximation. However, the fact that the domain has to be partitioned in predefined regions, randomly, or by using non supervised techniques, reduces the applicability of the algorithm.

In this work, we propose the use of an evolutionary algorithm that can learn simultaneously the partition of the domain, and the parameters of the local approximator used in each region.

Each local approximator consists in a vector that represents a point in the input space and a set of parameters. The output that corresponds to a specific input pattern is calculated by a function based on the parameters of the local approximator with the central point that is the nearest to the input pattern. By following this approach, the input space is partitioned in Voronoi regions, where the local approximator vectors correspond to the central points of these regions. The local approximators can represent any kind of local function. In this work we are considering both linear and RBF local approximators. Two strategies to implement continuity on region boundaries are considered.

The function to be approximated is represented by a set of N patterns $p_1, p_2, ...,p_N$, where each pattern p_i is composed of an input pattern $\eta_i =< x_{i1}, x_{i2}, ..., x_{id} >$with dimension d and an output value y_i. The patterns are re-generated every generation in order to guarantee good generalization ability.

2 The Voronoi diagram

Let $P = \{p_1, p_2, ..., p_n\}$ be a set of n distinct points in the plane; these points are the sites.We define the Voronoi diagram of P as the subdivision of the plane into n cells, one for each site in P, with the property that a point q lies in the cells corresponding to the site p_i if and only if $distance(q, p_i) < distance(q, p_j)$ for each $p_j \in P$ with $j \neq i$ [2]. We denote the Voronoi diagram of P with $Vor(P)$. The cell that corresponds to a site p_i is denoted $v(p_i)$. The figure 1 presents an example of the Voronoi diagram corresponding to a given set of points.

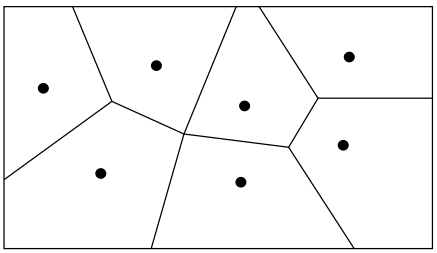

Figure 1: An example of a Voronoi diagram

3 The evolutionary algorithm

The algorithm is a standard evolutionary algorithm with a population that is modified by crossover and mutation operations. Selection of the individual is done with tournament and elitism is used. The representation of the individuals and the operators are described in the next subsections.

3.1 Individuals

Each individual consists in a list of variable length of local approximators. Each local approximator consists in two vectors, the central point $c_k =< c_{k1}, c_{k2}, ..., c_{kd} >$ and the parameters $a_k =< a_{k1}, a_{k2}, ..., a_{km} >$ that are used to represent the function defined by the local approximator k, where d corresponds to the dimension of the input patterns and m corresponds to the number of parameters. The figure 2 presents an individual with the second local approximator shown in detail.

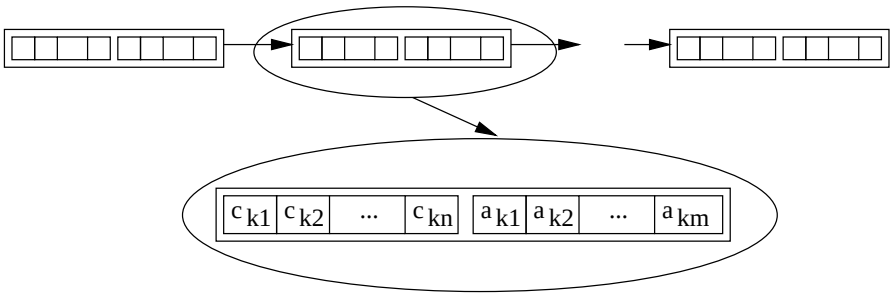

Figure 2: Individual with a detail on the second local approximator

The local approximator k of an individual is selected based on the distance between the input pattern $< x_1, x_2, ..., x_d >$ and the center point defined by the individual, as follows:

$$select\, min_k = \sum_{i=1}^{d} (x_i - c_{ki})^2$$

where $select\, min_k$ means that k is the index of the local approximator selected.

3.2 The local approximator function

The function computed by the local approximator can be any kind of function, as it was explained in section 1. In this work, we have restricted to linear and RBF local approximators.

Two strategies to implement continuity on cell boundaries are considered. The first one, called *continuity zero*, makes the function to take the value 0 in the boundaries. The second one, called *continuity L*, uses a global function to compute the values in the boundaries.

A linear local approximator has $d + 1$ parameters. The output o produced by a linear individual for an input pattern $\eta =< x_1, x_2, ..., x_d >$is calculated as the linear combination of the inputs multiplied by the parameters of the nearest local approximator, as follows:

$$o(\eta) = \sum_{i=1}^{d} x_i * a_{ki} + a_{k(d+1)}$$

An RBF approximator has $d+1$ parameters. The output o produced by an RBF individual is calculated as usual, with the $d+1$ parameter being considered as the output weight:

$$o(\eta) = \sum_{i=1}^{d} \exp(-(x_i - a_{ki})) * a_{k(d+1)}$$

The first approach to implement continuity is the continuity 0, or vanishing approach, where the values of the function tend to 0 near the boundaries. The output value is computed as follows:

$$o(\eta) = A_i * \psi(\frac{dist(\eta, \text{boundary})}{dist(\eta_i, \text{boundary})})$$

where η is the input pattern, η_i is the center of the cell to which the point η belongs to, A_i is the output of the local approximator at η, ψ is a function that controls the effect of the distance of η to η_i (linear in our experiments), $dist()$ is a function that computes the Euclidean distance between two points, and boundary is the nearest point to η in the boundary of the cell.

The effect of this output function is to produce the output of the local approximator at the center of the Voronoi cell, and approach the output to 0 in the boundaries by following a function ψ.

The second approach to implement continuity is the continuity L, where a separate global function provides the values in the boundaries. The effect of the output function is to produce the output of the local approximator at the center of the Voronoi cell, approaching the output of the global function at the boundaries.

The output function with continuity L is defined as follows:

$$o(\eta) = A_i * \psi(\frac{dist(\eta, \text{boundary})}{dist(\eta_i, \text{boundary})}) + L(\eta)$$

where L is a linear approximator that fulfills the following restrictions:

- $L(\eta_i) = V_i$
- L is linear
- L is continuous in the domain

and V_i is the real value that the function gets at the center of the Voronoi cell.

L is the global function that provides the values in the cell boundaries. The value of L can be computed based on the values of the three nearest Voronoi regions.

3.3 Crossover

The Voronoi crossover defined in [3] is used. This crossover interchanges subsets of Voronoi cells. A line segment is selected randomly on both parents, and the children are formed with the cells from both parents that lie in opposite sides of the hyperplanes. The figure 3 shows an example of the application of the Voronoi crossover.

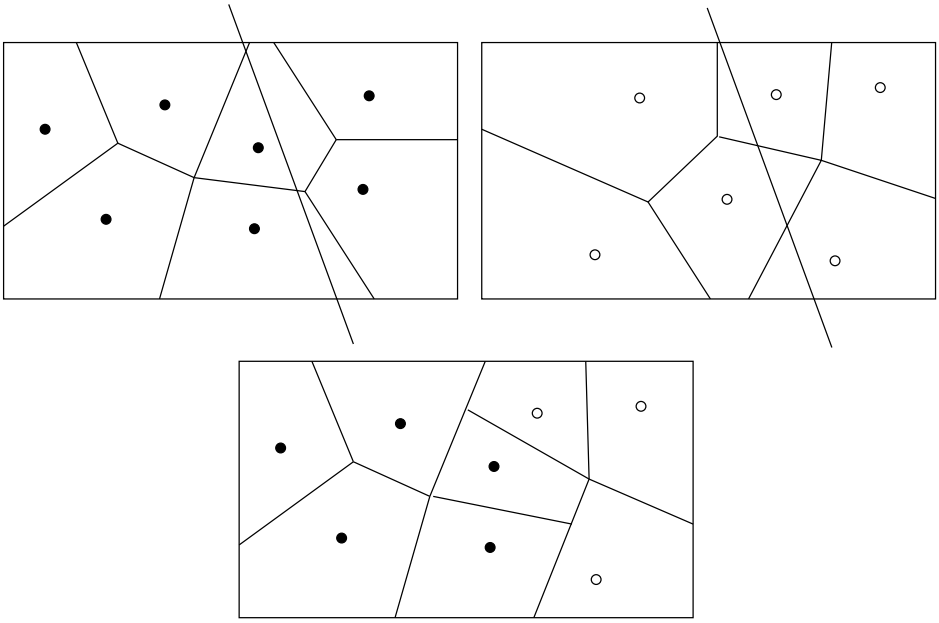

Figure 3: An example of the application of the Voronoi crossover

3.4 Mutation

Three kinds of mutation operators are used:

- *local approximator modification*: its effect is a slight modification of the centers and/or the parameters of a local approximator. The objective of this operator is to move in the input space the area covered by the local approximator or to make a minor change in the function used in this Voronoi region.
- *local approximator creation*: its effect is the addition of a new local approximator with random center and parameters. The objective of this operator is the creation of new local approximators in areas not covered before and the increment of the number of local approximators that belongs to an individual.
- *local approximator deletion*: its effect is the deletion of a local approximator. The objective of this operator is the elimination of local approximators that are useless in an individual.

3.5 Fitness evaluation

The fitness of an individual is computed as the squared error between the output o and the desired output y on all patterns.

$$fitness = \sum_{p=0}^{N} (y_p - o_p)^2$$

The evolutionary algorithm tries to obtain the individual with the lowest fitness, i. e., the individual with the smallest error.

The patterns are randomly generated in the domain of the target function for each generation of the evolutionary algorithm in order to increase the generalization ability of the approximation.

4 Experiments

Three experiments were considered. The first problem (called "add&sub") is a combination of two linear functions defined in different areas of the domain. The second problem (called "add&sub&prod") is a combination of three linear and one nonlinear function. The third problem ("prod&prod") is defined as a combination of two nonlinear functions. They are presented in figure 4.

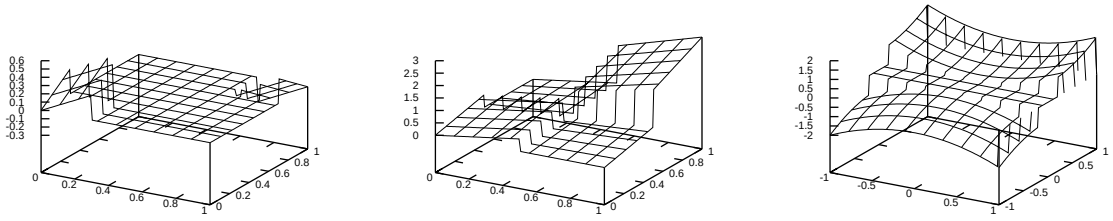

Figure 4: target functions

The experimental runs were based on a population of 100 individuals. Tournament selection was used in all cases. The probability of the three kinds of mutation and crossover were kept fixed for all experiments in experimentally selected good values: 0.01 for mutations and 0.65 for crossover.

The results are presented in 12 graphics for each experiment, and corresponds to typical runs. The first set of six graphics corresponds to the experiment run with linear approximators. The second set to the experiment run with RBF approximators. The first column of each set of graphics corresponds to the experiment run with no continuity, the second column to the experiment run with continuity 0 and the third to the experiment run with continuity L. The first row of each set of graphics corresponds to the performance of the approximation evaluated on a set of 100 random patterns. The original patterns are shown with a sign "+" and the approximation with a sign "X". The second row shows the error plotted against the number of generations. Please note that the range of the axes in the plots can be different due to the fact that in some cases more generations are necessary to get good results. Please note that the plot of the error against the number of generations seems unstable is some cases. This is due to the fact that a new set of patterns is generated for every generation of the evolutionary algorithm.

In the first experiment (figures 5 and 6), all approaches provide a good solution for the problem. This is expected, due to the fact that the problem is linear and simple. Continuity seems important by considering the evolution of the error against the number of generations. It also helps the RBF approximators to approach linear functions with non linear RBF outputs.

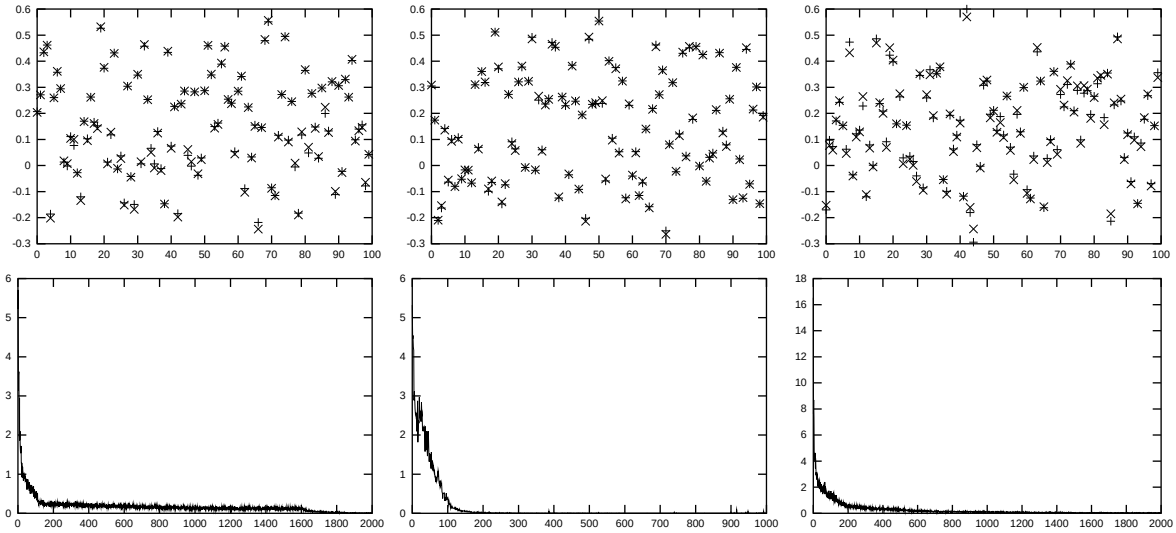

Figure 5: Performance and evolution of the experiment 1 with linear approximators with (a) no continuity (b) continuity 0 and (c) continuity L

When RBF is used without continuity, the algorithm needs more generations to start to get good quality solutions.

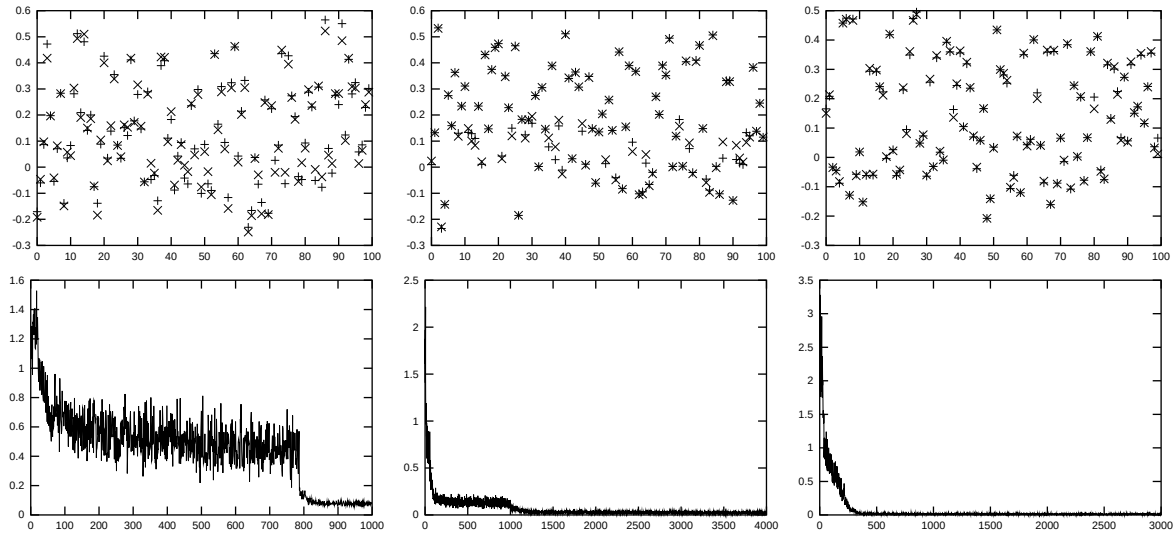

Figure 6: Performance and evolution of the experiment 1 with RBF approximators with (a) no continuity (b) continuity 0 and (c) continuity L

In the second experiment (figures 7 and 8), the solutions obtained by RBF local approximators are better. This is expected, due to the fact that the problem is a combination of linear functions and a non linear function. Both continuity 0 and L helps RBF approximators to produce high quality solutions.

In the third experiment (figures 9 and 10), the solutions obtained by RBF local approxi-

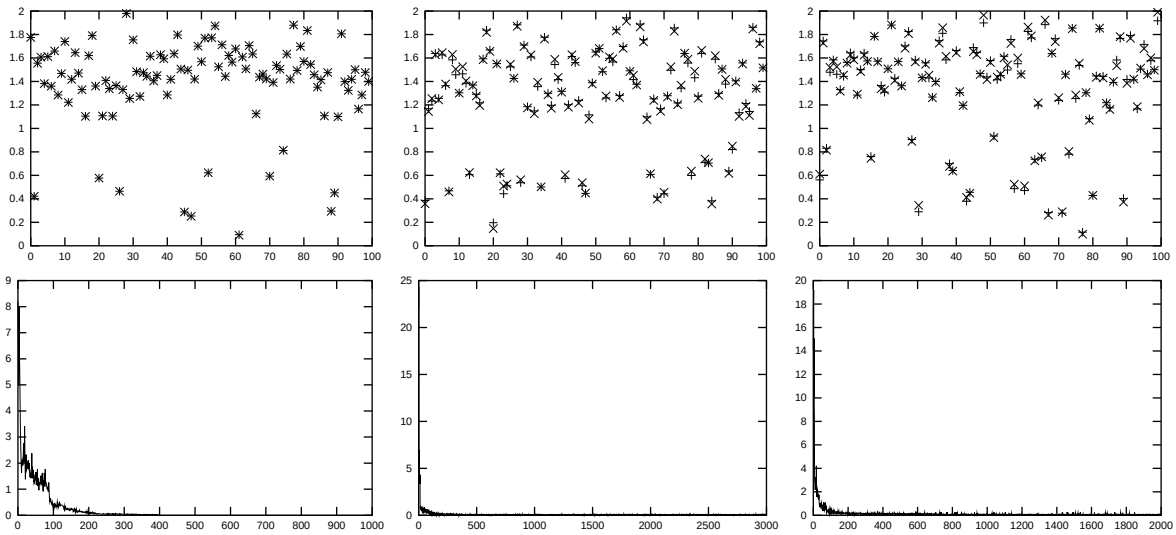

Figure 7: Performance and evolution of the experiment 2 with linear approximators with (a) no continuity (b) continuity 0 and (c) continuity L

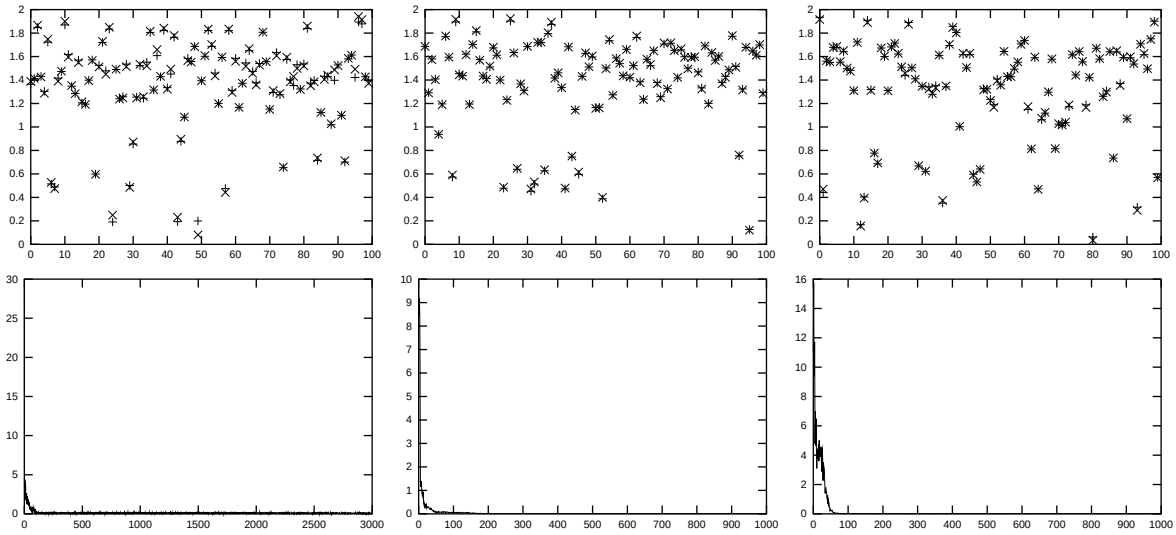

Figure 8: Performance and evolution of the experiment 2 with RBF approximators with (a) no continuity (b) continuity 0 and (c) continuity L

mators are better. This is expected, due to the fact that the problem is a combination of no linear functions. It can be seen that this problem can also be approached with linear functions by using continuity.

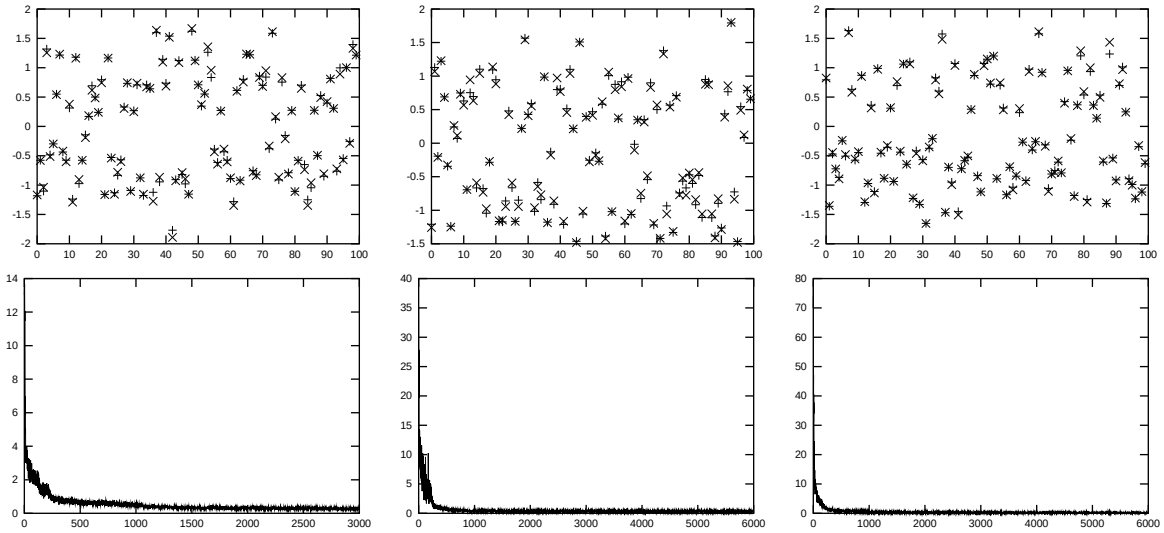

Figure 9: Performance and evolution of the experiment 3 with linear approximators with (a) no continuity (b) continuity 0 and (c) continuity L

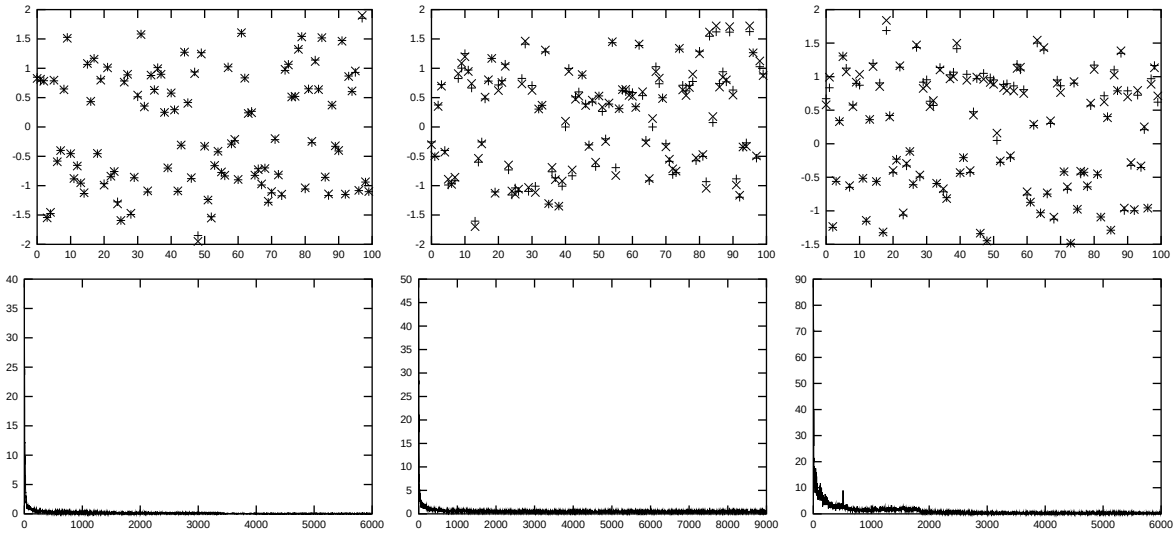

Figure 10: Performance and evolution of the experiment 3 with RBF approximators with (a) no continuity (b) continuity 0 and (c) continuity L

5 Conclusions

In this work, we have proposed an evolutionary algorithm that can be used to fit functions with a set of local approximators. The evolutionary algorithm evolves the local approximators and the region of the domain in which they are used. The domain is partitioned by following a Voronoi approach. The algorithm impose no restrictions on the kind of local approximators used. Continuous and non continuous approaches are considered.

Experiments with linear and RBF local approximators show that both kind of approximators

provide good performance. Of course, linear approximators are better suited to approach linear functions and RBF to approach non linear functions. However, when they are combined in a continuous way, they can approach successfully any kind of surface.

The approach seems promising in order to fit continuous and non continuous functions. New problems are under study now. Individuals with different kinds of local approximators are also being considered.

6 Acknowledgments

We thank Lic. Gloria Simonetti, from the Department of Mathematics in the Universidad Nacional de San Luis, Argentina, for helping us to define mathematical procedures to implement the approximators. We thank also Prof. Raul Gallard, Head of the LIDIC, for his continuous support.

References

[1] Tomas Back, David Foguel and Zbigniew Michalewicz (Eds.) (1997-2000). *Handbook of Evolutionary Computation.* Oxford University Press.

[2] Mark de Berg, Mark van Kreveld, Mark Overmars and Otfried Schwarzkopf (1997). *Computational Geometry, Algorithms and Applications.* Springer.

[3] Marc Schoenauer, Francois Jouve and Leila Kallel (1997). *Dasgupta and Z. Michalewicz Ed. Identification of Mechanical Inclusions. Evolutionary Algorithms in Engineering Applications.* Springer Verlag.

[4] Marc Schoenauer, Leila Kallel and Francois Jouve (1996). *Mechanical Inclusions Identification by Evolutionary Computation.* Revue Européenne des éléments finis. Vol 5(5-6).

[5] Simon Haykin (1994) . *Neural Networks, A Comprehensive Foundation.* Macmillian College Publishing Company.

[6] Carlos Kavka and Patricia Roggero (2001). *Evolucion de Redes Neuronales Locales.* Workshop de Investigadores en Ciencias de la Computación. San Luis, Argentina.

[7] Emile Fiesler and Russell Beale (Eds.) (1997). *Handbook of Neural Computation.* Institute of Physics Publishing.

[8] Earl Swokowski (1987). *Cálculo con geometría analítica.* PWS Publishing Co.