Evolutionary Training of Morphological Neural Networks

Roberto C. Piñeiro Colón
Dr. Jorge L. Ortiz

Electrical and Computer Engineering Department
University of Puerto Rico, Mayagüez Campus
Mayagüez, Puerto Rico 00681
roberto.pineiro@ece.uprm.edu
jortiz@ece.uprm.edu

Abstract

This article describes the use of genetic algorithms as a training tool for morphological neural networks described by Ritter [Ritter96]. Morphological neural networks are a new type of neural networks that replace classical operations of multiplication and addition by addition and maximum or minimum operations. The method used to train the network only one possible implementation, other implementations can be used.

1. Introduction

Since a morphological neural network needs to be trained, various methods have been proposed to train it. One of them was the algorithm proposed by Ritter in his article [Ritter99] in addition to a hybrid method proposed by Lima [Lima01] who combines genetic algorithms with a gradient technique.

This paper describes one possible implementation of a genetic algorithm used to train a morphological neural network. The encoding of the solution into a chromosome, the crossover and mutation are explained and an example of a possible run is given.

2. Genetic Algorithms

Genetic algorithm [Fogel94] and [Fogel97] is an optimization technique based on Darwin's theory about evolution. In order to use genetic algorithms the solution must be represented as a chromosome. The encoding of the solution into a chromosome is the most important step for the genetic algorithm, since different encoding can affect the performance of the system and the time needed reach the solution. A population of possible solutions represented by organisms is created, and then genetic operators such as crossover and mutation are applied to evolve the population in order to find the best solution.

3. Morphological Neural Networks

Morphological Neural Networks (MNN) are a new type of neural network described by Ritter [Ritter96], [Ritter98] and [Ritter99]. These types of neural networks replace the classical operations of multiplication and addition by addition and maximum or minimum operations. The maximum and minimum operations allow to perform a nonlinear operation before the application of the activation or transfer function. MNN utilize algebraic lattice operations structure [Ritter96] known as semi-ring $(\Re_{\pm\infty},\vee,\wedge,+,+')$, different from traditional neural networks that are based on the algebraic structure known as ring $(R,+,\times)$. The operations ? and ? denote binary operations of minimum and maximum, respectively. The algebra of matrices over $\Re_{\pm\infty}$ provides for an elegant way to express the total input effect on a morphological neural network layer.

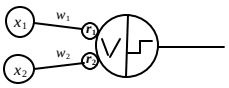

Figure 1. Example of a morphological perceptron.

4. Evolving Training

The usage of genetic algorithms as a learning tool for a morphological neural network was introduced by Lima [Lima01]. He used the evolving training to improve a gradient technique.

The proposed algorithm identifies the weights for all the defined class in order to train a morphological neural network. In general the algorithm can train the network for two classes at the same time. In order to train the network for multiples classes the test patterns that do not belong to the class that is going to be trained must be regrouped into a temporary class. If a system with *m* classes is going to be trained, then the weights

for a class C_i must be identified, where $0 < i < m$, therefore a temporary class C_t must be used. The class C_t contains all the test patterns from all the classes without including the test patterns from class C_i.

In a two dimensional space the training of the system could be visualized as the search for groups of patterns that can be enclosed by rectangular regions. The borders of these rectangular regions are the decision boundaries.

4.1 Encoding of the Organism into a Chromosome

The way the problem is encoded into the chromosome affects the performance of the system. There are different ways to encode a problem into a chromosome. The binary encoding is the most common way to encode the data into a chromosome. In this case the encoding used is order-based encoding.

The chromosome is built of two parts: the first part is built by the index of each test pattern that belongs to the class C_i and the second part contains the way these test patterns are grouped. Assuming the class C_i contains n test patterns, then the chromosome has n+1 entries. Each of the first n entries contains a unique value between 0 and n-1 that represent a particular test pattern. The values for the entries can not be repeated.

For example, given the class $C_0 = \{p_0 = (-2,4), p_1 = (1,4), p_2 = (2,4), p_3 = (2,5), p_4 = (-1,2), p_5 = (-2,1), p_6 = (-1,1)\}$, $C_1 = \{p_7 = (0,1), p_8 = (0,2), p_9 = (0, 3)\}$, y $C_2 = \{p_{10} = (1,2), p_{11} = (2,2), p_{12} = (3,2), p_{13} = (3,3)\}$. In the fist iteration, lets C_t be equal to the class C_0. The remaining test patterns from the classes C_1 y C_2 have to be regrouped into the temporary class $C_t = \{(0,1), (0,2), (0, 3), (1,2), (2,2), (3,2), (3,3)\}$. The index of each of the test patterns from the class C_i is placed on the first n entries of the chromosome.

Figure 1 shows an example of a chromosome generated randomly. The indexes of the test patterns from the class C_i are encoded within the chromosome in the first n entries and the order was randomly selected. At the end of the chromosome a value d is randomly generated. The meaning of d is discussed in the next paragraph. See Figure 2.

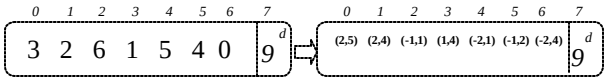

Figure 2. Randomly generated values encoded into the chromosome

The way the groups are established is specified in the second part of the chromosome. At the end of the chromosome a binary number of n-1 bits, where n is the number of test patterns contained in the class C_t, known as d, where d(j) represent the bit at the position n-j. The value of d represents the subgroups in the class C_i. The test pattern in the entry k of the chromosome, where $0 \leq k < n$-1, is in the same groups as the test pattern in the entry k+1 if and only if d(k) = 0. In case d = 0, this means all the test patterns belongs to the same group. See figure 3.

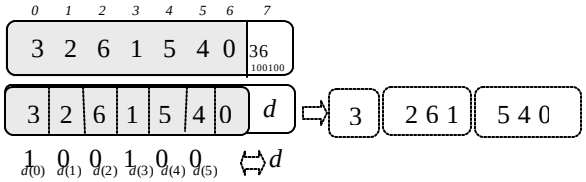

Figure 3. Representation of the second part of the chromosome.

For each group of test patterns, a rectangular region that includes all the test patterns is defined. The lower left corner and the upper right corner of these rectangular regions represent the weights to be used in the configuration of the morphological neural network. In the case one of the dimensions is equal to zero then it must be expanded in a way that the region includes all test patterns of the group but without including any test pattern from the class C_t.

4.2 Crossover

The crossover between two organisms produces two new offspring that have characteristics from the two parents. After the organisms are evaluated two parents are selected randomly. The genetic information from one parent is combined to the genetic information from the other parent to create a child that has characteristics from both parents.

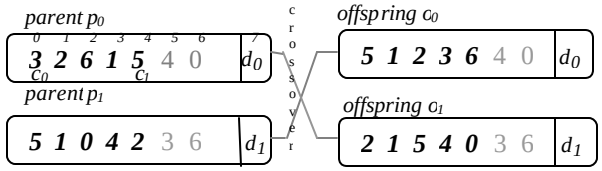

Figure 4. Crossover of two organisms.

This is a possible approach of the implementation of a crossover and it is based on the order-based crossover. Figure 4 shows the two parents before the crossover, parent p_0 and p_1 and the offspring o_0 and o_1. For the crossover used in this algorithm two indexes must be selected randomly, index c_0 and index c_1, where $0 \leq c_0 < n$-1 and $c_0 < c_1 \leq n$-1. In order to apply the crossover to the parent p_0 over the parent p_1 all the elements in the entries between c_0 and c_1 are reordered according the way they appear on parent p_1. Finally the value of d_0 is copied from parent p_0 to offspring o_0. This process produces offspring o_0. In order to apply the crossover to the parent p_1 over the parent p_0 all the elements in the entries between c_0 and c_1 are reordered according the

way they appear on parent p_0. Finally the value of d_1 is copied from parent p_1 to offspring o_1. This process produces offspring o_1. In this implementation of the crossover the groups do not change, they are copied from the parents to the offspring.

4.3 Mutation of the Chromosome

Mutation is the process where the genetic information of an organism is changed in order to add some diversity to the population, adding some characteristics that could improve for the population.

The mutation used in the implementation of this algorithm is random binary mutation and this mutation is applied only in the last part of the chromosome, that is, the part where the groups of test patterns are specified. Each one of the bits in the value *d* is changed according to a probability of mutation *p*M. In the figure 5 the original chromosome is on the left side, at the bottom is the binary representation of the *d* before the mutation. After the mutation is applied the *d* value has changed, resulting in a change of the subdivisions.

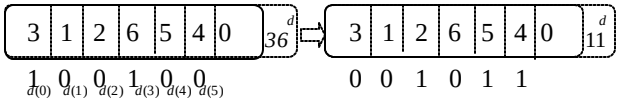

Figure 5. Mutation of the genetic information of an organism

The mutation in this example, see Figure 6, change how the groups of test patterns are composed. This is done by changing the number of groups or by adding elements to a group or removing it from a group.

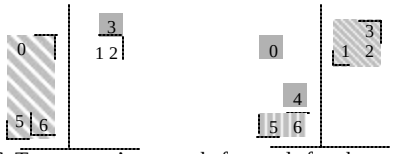

Figure 6. Test pattern's groups before and after the mutation

4.4 Fitness Function

Each organism must be evaluated according to the characteristics it has. Only the organisms that have the desired characteristics can mate other organisms and transmit their own characteristics to the offspring.

When an organism is evaluated, two factors are very important: the number of test patterns that classify incorrectly, known as *K* and the number of groups represented by *L*. To smaller number of patterns classified incorrectly, better it is the evaluation. To smaller number of groups, better it is the evaluation. The fitness function is defined as follow: $f = (K+1)^2 * L$.

4.5 Selection of the Parents

Not all the organisms in the population are the able to transmit their characteristics to the new generation. A selection process is used to allow organisms who have higher fitness to transmit their characteristics with higher probability than the ones who have a lower fitness.

The selection process used in this genetic algorithm was roulette wheel selection. In the roulette wheel selection the probability an organism is selected to mate another organism is equal to the fitness of the organism divided by the total fitness all the organisms.

5. Example

Let define given the class C_0 = {(-1,4), (1,4), (1,3), (0,4)}, C_1 = {(-1,3), (0,2), (0,3), (-2,4)}, C_2 = {(1,2), (2,2), (2,3), (2,4)}. See figure 7. Let's use the notation $C_i(n)$ is the test pattern of the class C_i at the index n.

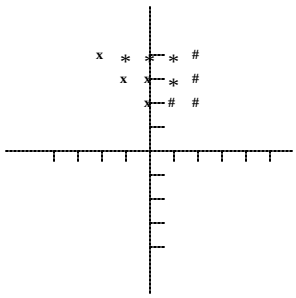

Figure 7. Test patterns for the example used in this section, where * represent class C_0, x represents class C_1 and # represents class C_2.

In the first iteration i=0 then $C_i = C_0$ and C_t = {(-1,3), (0,2), (0,3), (-2,4), (1,2), (2,2), (2,3), (2,4)}
The initial random population of chromosomes could be:

$CR_0 = [3,1,2,0,0]$, $CR_1 = [0,1,3,2,6]$
$CR_2 = [1,3,0,2,3]$, $CR_3 = [2,0,3,1,7]$

First, the chromosomes must be evaluated using $f = (K+1)^2 * L$:

$f(CR_0) = 9$, $f(CR_1) = 8$, $f(CR_2) = 18$, $f(CR_3) = 4$,

Crossover and mutation:
The crossover of the parent CR_3 and CR_1, with the index points $c_0 = 0$, $c_1 = 2$:

crossover after mutation
$CR_3 = [2,0,3,1,7] \Rightarrow O_0 = [0,3,2,1,7] \Rightarrow O_0 = [0,3,2,1,6]$
$CR_1 = [0,1,3,2,6] \Rightarrow O_1 = [0,3,1,2,6] \Rightarrow O_1 = [0,3,1,2,6]$

The crossover of the parent CR_3 and CR_0, with the index points $c_0 = 1$, $c_1 = 2$:

crossover after mutation
$CR_3 = [2,0,3,1,7] \Rightarrow O_2 = [2,3,0,1,7] \Rightarrow O_2 = [2,3,0,1,5]$
$CR_0 = [3,1,2,0,0] \Rightarrow O_3 = [3,2,1,0,0] \Rightarrow O_3 = [3,2,1,0,4]$

The final population is:

$CR_0 = [0,3,2,1,6]$, $CR_1 = [0,3,1,2,6]$
$CR_2 = [2,3,0,1,5]$, $CR_3 = [3,2,1,0,4]$

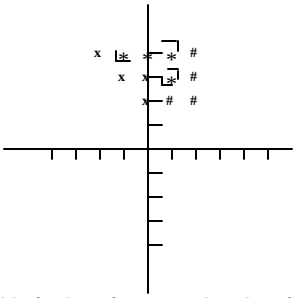

Figure 8. A possible final configuration that classify test point for the class C_0

The process is repeated until the best solution is found. In this case the best solution could be [0,3,1,2,1], but there are other possible solutions. This process usually converges in 5-6 generations for this number of test patterns.

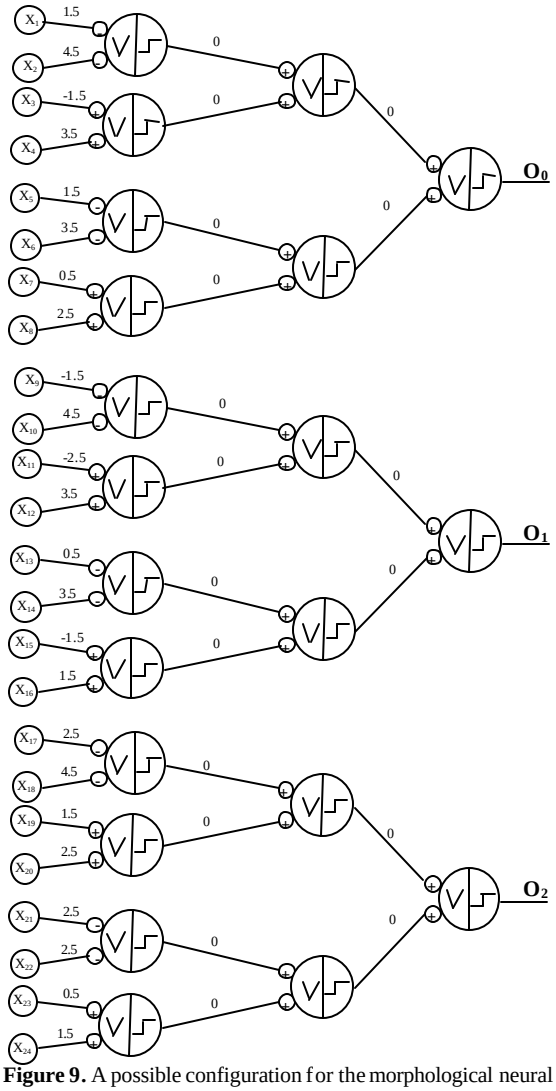

Figure 9. A possible configuration for the morphological neural network.

In the second iteration i=1 and the class C_1, is the class to be trained, then C_t = {(-1,4), (1,4), (1,3), (0,4), (1,2), (2,2), (2,3), (2,4)} and in the last iteration i=2 and the class C_2 is the class to be trained, then C_t = {(-1,4), (1,4), (1,3), (0,4), (-1,3), (0,2), (0,3), (-2,4)}. The final result of the training could be similar to Figure 10. Figure 9 shows a possible configuration for the morphological neural network that classifies the system. Table 1 shows the meaning of the outputs O_0, O_1 and O_2.

O_0	O_1	O_2	
0	0	0	none
1	0	0	**C_0**
0	1	0	**C_1**
0	0	1	**C_2**

Table 1. Interpretation of the results from the outputs O_0, O_1 and O_2 of the network.

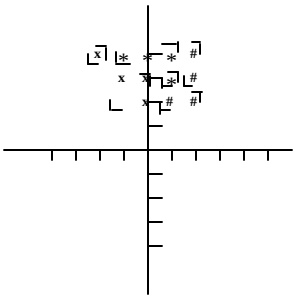

Figure 10. A possible final result of the system.

6. Conclusions

This paper describes one implementation of the training method for a morphological neural network. There are many other implementations of the crossover and mutation operations that can be applied to this method. Also the solution can be encoded in another way. This method work well

7. References

[Fogel94] Fogel, D.B., "An introduction to simulated evolutionary optimization" Neural Networks, IEEE Transactions on, Volume: 5 Issue: 1, Jan. 1994 Page(s): 3-14

[Fogel97] Fogel, D.B. "Evolutionary Computation: A New Transactions" Evolutionary Computation, IEEE Transactions on, Volume: 1 Issue: 1, April 1997 Page(s): 1-2

[Lima01] Lima, C.A.M., Coelho, A.L.V., Silva, M.E.S., Gudwin, R.R. and Von Zuben, F.J.: Hybrid Training of Morphological Neural Networks: A Comparative Study, Procs. of National Meeting of Artificial Intelligence (ENIA), Congress of Brazilian Computing Society

(SBC), pp. 1499-1507, Fortaleza, Brazil, July-August 2001

[Ritter96] Ritter, G.X.; Sussner, P. “An introduction to morphological neural networks”, Pattern Recognition, 1996. Proceedings of the 13th International Conference on, Volume: 3, 1996 Page(s): 709 - 717 vol.4

[Ritter98] Ritter, G.X.; Beaver, T.W. “Morphological perceptrons” Neural Networks, 1999. IJCNN '99. International Joint Conference on, Volume: 1, 1999, Page(s): 605-610 vol.1

[Ritter99] Ritter, G., Beavers, T.W. “An introduction to morphological perceptrons”, ANNIE ’99 Proceedings of the, Nov. 1999, Page(s): 1-6

[Sussner98] Sussner, P. “Morphological perceptron learning” Intelligent Control (ISIC), 1998. Held jointly with IEEE International Symposium on Computational Intelligence in Robotics and Automation (CIRA), Intelligent Systems and Semiotics (ISAS), Proceedings of the 1998 IEEE International Symposium on, 1998 Page(s): 477-482

[GA] – The Genetic Algorithm Optimization Toolbox (GAOT) for Matlab 5, North Carolina State University.