# Indirect Encoding Evolutionary Learning Algorithm for the Multilayer Morphological Perceptron

**Jorge L. Ortiz, Ph.D., PE and Roberto C. Piñeiro, BS**

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

**Abstract**

This article describes an indirectly encoded evolutionary learning algorithm to train morphological neural networks. The indirect encoding method is an algorithm in which the training of the neural network is done by finding the solution without considering the exact connectivity of the network. Looking for the set of weights and architecture in a reduced search space, this simple, but powerful training algorithm is able to evolve to a feasible solution using up to three layers required to perform the pattern classification. This type of representation provides the necessary compactness required by large networks. The algorithm was tested using Iris Fisher data and a prototype was written using Matlab.

## Introduction

Morphological Neural Networks (MNN) are a new type of neural networks described by Ritter, Sussner, and Beavers (Ritter and Sussner 1996), (Ritter and Sussner 1997), (Sussner 1998), and (Ritter and Beavers 1999). 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 performing a nonlinear operation before the application of the activation or transfer function. MNN utilize algebraic lattice operations structure 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 $\wedge$ and $\vee$ denote minimum and maximum binary operations, respectively.

Genetic Algorithms (Yao 1999) have proven to be effective to search for an optimal solution in very large, complex, and irregular search spaces such as the neural networks architectures. This article describes a method using genetic algorithms that can be used to train the morphological neural networks introduced by Ritter, Sussner and Beavers. The algorithm can be used to train up to three layers morphological perceptron architectures based on evolutionary computation, which are able to classify most traditional pattern classification problems.

## Morphological Neural Networks

Morphological neural networks are a new type of neural network, based on lattice operations. The morphological neuron follows the mathematical model described by Equation 1,

$$f\left(p_j \cdot \bigvee_{i=1}^{n} r_{ij}\left(x_i + w_{ij}\right)\right) \qquad (1)$$

where $\vee$ is the maximum operator (or minimum operator $\wedge$ can be used), $x_i$ is the $i$-th input value for the j-th neuron, $w_{ij}$ denotes the synaptic weight associated between the $i$-th input and the $j$-th neuron, $r_{ij}$ represents the inhibitory or excitatory pre-synaptic value between the $i$-th input and the $j$-th neuron, and $p_j$ represents the post-synaptic response of the $j$-th neuron. Both $r_{ij}$ and $p_j$ can assume values of {+1, -1}.

In addition, the morphological perceptron uses a special hard-limit transfer function, as shown in Equation 2:

$$f:\mathbb{R}\to 0,1 \qquad x \to \begin{cases} 1 \text{ if } x>0 \\ 0 \text{ else} \end{cases} \qquad (2)$$

Figure 1 shows a graphical representation of a two-layer morphological neural network.

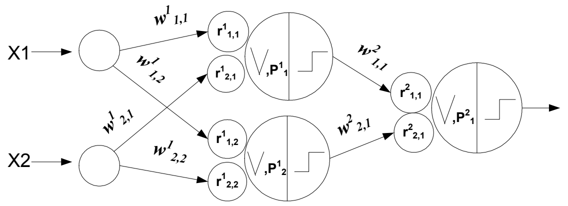

**Figure 1**. Two layer morphological perceptron architecture.

The articles presented by Sussner (Sussner 1998), describe the different effects produced by changing parameters of the morphological neuron. Figure 2a shows the effect of using positive values as the pre-synaptic values for a two-input morphological neuron, and the corresponding classification region obtained is shown in Figure 2b. Figure 2c shows the effect of using negative values as pre-synaptic values in a morphological neuron, and the corresponding classification region is shown in Figure 2d.

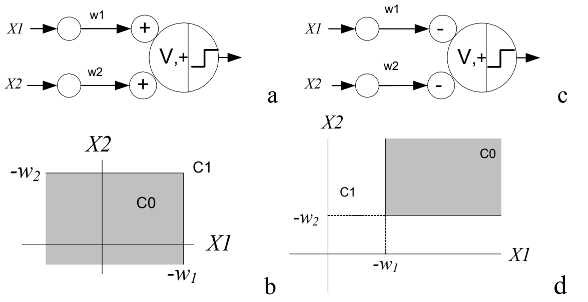

**Figure 2**. (a) A morphological neuron with positive pre-synaptic values and (b) the corresponding region defined for the class $C_0$. (c) A morphological neuron with negative pre-synaptic values and (d) the corresponding region defined for the class $C_0$.

## Training by Indirect Encoding

The proposed algorithm identifies the number of necessary neurons needed to perform the classification, the set of weights, and the architecture for the morphological neural network that can be used to classify patterns. In general, a morphological perceptron can separate only two classes. In order to classify multiple classes, a vector that contains a binary pattern is assigned to each class, for example:

$$C_0 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \; C_1 = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \text{ and } C_2 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

A neural network may be trained for each entry in the classification vector. Building a neural network to classify all the test patterns for the first entry in the vector correctly, requires to assign test patterns from classes that have the value of 0 to a temporary class $C_{t0}$, otherwise to the class $C_{t1}$. Those temporary classes will be used during the training process of the neural network. Figure 3a shows the set of test patterns, and their corresponding binary vector. Figure 3b shows how all test patterns have been regrouped into temporary classes. A multilayer morphological perceptron is built in such a way that it will be able to separate the patterns from the new classes $C_{t0}$ and $C_{t1}$. The output of that network is assigned to the first entry in the binary vector. Figure 3c shows the test patterns must be regrouped in order to build the neural network for the second entry in the binary vector.

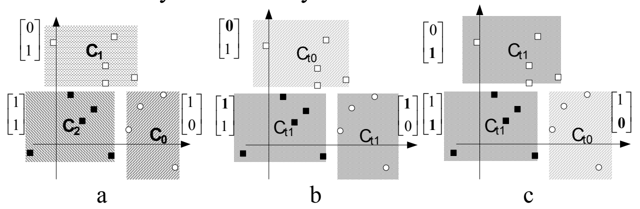

**Figure 3**. (a) Set of test patterns and their corresponding binary vector. (b) and (c) how the patterns are regrouped in temporary classes.

The encoding of the network parameters is done using an indirect encoding method, instead of looking for the number of required neurons for the classification, the set of weights for each neuron, the number of layers and interconnections between neurons. The problem is restated in such a way that the solution for this new problem results in a simpler representation. Once a solution is found a morphological neural network is built using the indirectly encoded information.

The example in Figure 4 shows a 2-dimensional space example where the classification patterns are grouped into clusters. The boundaries for these clusters can be approximated by succession of rectangular regions where the corners of each of these regions can be seen as the decision boundaries of a morphological neural network, as is shown in Figure 4. The same concept can be extended to a higher domain space.

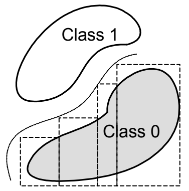

**Figure 4**. The region of the class $C_0$ is approximated by a succession of rectangles.

Once these regions are defined, the corners can be used as the decision boundaries in order to build a morphological neural network able to classify patterns between two classes. The indirect encoding of the problem provides a solution with enough information that can be used to rebuild a morphological neural network. An algorithm is implemented to define the way this information may be decoded to build the morphological neural network. This process includes the way this information is fit into the chromosome, and the crossover and mutation techniques implemented to solve the problem.

The neural network architectures used to create the hypercubes have the following restrictions: the neural network architecture must have three layers, except in the simple case in which all the patterns can be grouped using a single hypercube, where it will have only two layers; the last layer will have only one neuron, which uses the minimum operator, this neuron is connected to all the outputs from all the neurons of the second layer; all the neurons in the first layer use the maximum operator and one of the two neurons for each hypercube uses +1 for all the pre-synaptic values and the other one uses -1. Additionally, the weights for the connection in the layers two and three will always be 0 and the post-synaptic value for all the neurons in the neural network will be +1. Figure 5 shows a diagram of the architecture of the neural network as described in this training algorithm.

### Encoding of the Organism

The way the problem is encoded into the chromosome affects the performance of the algorithm. Different from other approaches, in this research nothing regarding to the connection weights or the relationship between the neurons, or the neural network architecture will be encoded into the chromosome. Since the problem is encoded indirectly, the chromosome keeps only enough information to identify each of the test patterns that belongs to the class $C_{t0}$. Inside of the chromosome or genotype, there are groups or set of patterns, each of them represents clusters of patterns. Each set must contain at least one pattern, and no empty groups can be used.

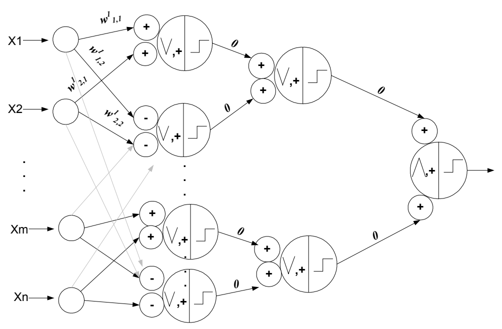

**Figure 5**. Shows how the morphological neural network architecture may look.

An example is shown in Figure 6, where the class $C_{t0}$ contains ten 2-dimensional patterns, identified as P# enclosed by a square, where # is an integer number that represents each pattern. The patterns that do not belong to the class $C_{t0}$ are represented with circles. It is important to highlight that only those patterns that belong to class $C_{t0}$ are encoded into the chromosome, using an integer value that corresponds to each pattern. When the initial population is generated, the patterns are randomly distributed into the chromosome in no particular order and the groups are randomly generated. In Figure 6b, shows how the patterns may be coded in the chromosome, in addition to the graphic representation of the hypercube that encloses each pattern group defined in the chromosome.

After all groups are defined, the elements of each group are used to define the limits of the hypercube that are used as the decision boundaries for the morphological neural network. A hypercube is defined for each group in the chromosome in such a way that it includes all the elements for that particular group.

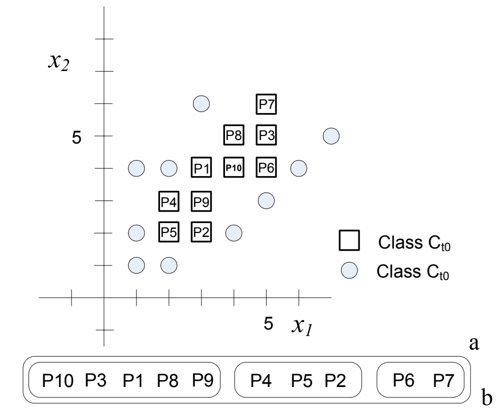

**Figure 6.** An example of how the patterns may be encoded into the chromosome of a randomly generated organism.

## Recombination

The crossover used in the implementation of the algorithm selects a set of *n* elements randomly from different groups defined in the chromosome of the first parent. The selected elements are identified and their positions are exchanged in the first chromosome, according to order they appear in the second parent.

The process is repeated again, but this time the exchange of elements is done in the second parent based on the order they appear in the first parent.

Figure 7a shows the chromosome of parent 1, with 10 patterns coded on it. Also, Figure 7b shows the hypercube for each group defined in the chromosome. Figure 8 shows the chromosome of the second parent as it will be used for the crossover process.

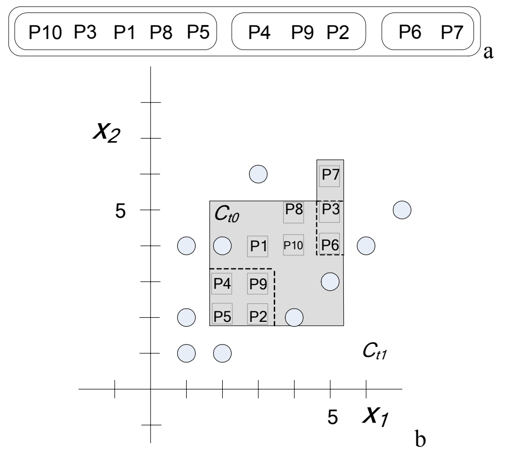

**Figure 7**. (a) Chromosome of first parent and (b) the corresponding set of hypercubes.

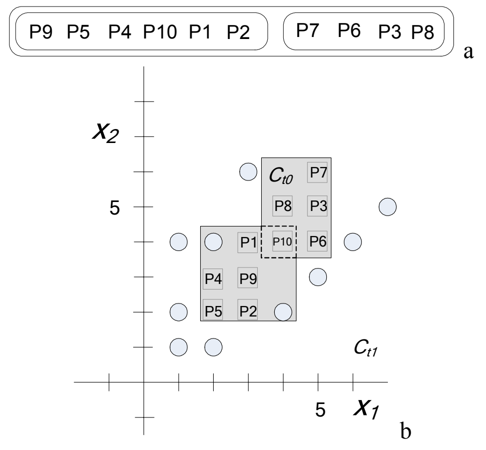

**Figure 8.** (a) Second parent used for the crossover and (b) the corresponding set of hypercubes.

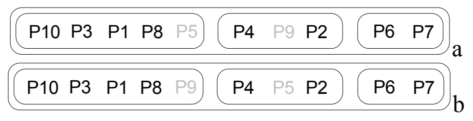

**Figure 9**. (a) First parent before the crossover and (b) the resulting offspring.

Assume P9 and P5 are the selected elements from the first parent, as shown in Figure 9a. Now these elements must be identified in the second parent and the order is exchanged according the way they appear in the second parent. The final result after the elements were exchanged is shown in Figure 9b. In order to obtain the second offspring the process is repeated, but this time the selection and exchange of the elements is done in the second parent according to the order they appear in the first parent.

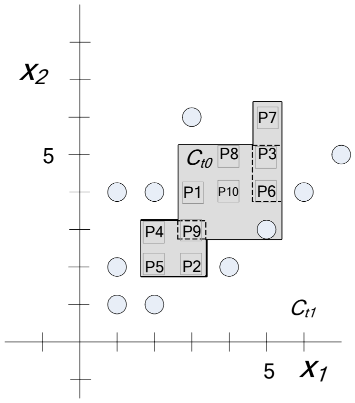

**Figure 10.** Hypercubes for the resulting offspring.

## Mutation

The mutation operation on a chromosome, in the proposed algorithm, consists of two possible operations: fusion of two groups or division of a group into two new groups. In the fusion of two groups, two groups are randomly selected, and then all the elements of these two groups are combined to create a new group. Using this approach mutation is used to introduce changes in the way groups were created before. The other groups in the chromosome remain untouched. Figure 11a, shows an example of how the groups are defined before the mutation and Figure 11b shows the resulting chromosome after mutation. This example of mutation shows elements from different regions grouped into one set, combining those elements that may be grouped together into a single hypercube. Figure 12 shows the graphical effect of the mutation in the hypercubes.

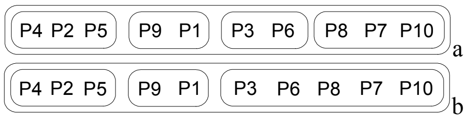

**Figure 11**. (a) Chromosome before mutation and (b) after mutation using group division.

Another example of a possible mutation operation is the redistribution of the elements of a group into two different groups. In this case, one group must be selected and all the elements of the group are distributed randomly between the two new groups. Figure 13a shows an example of a chromosome before the mutation and after the mutation, Figure 13b, where the elements of a group has been distributed into two different groups. This mutation operation helps to separate those elements that should not be in the same group. Figure 14 shows mutation effect graphically.

## Reconstruction of the Neural Network

The information encoded in the chromosome will be used to build a morphological neural network that is evaluated later to calculate network fitness. The boundaries of the hypercubes must be decoded to build the neural network. Each group defined in the chromosome creates a hypercube large enough to enclose all the test patterns defined in that group. For each dimension in the hypercube, the maximum and minimum values will be used as the weights for the neurons that will define the hypercube. Figure 15 shows an example of a chromosome and a 2-dimensional space and the corresponding maximum and minimum values for the first hypercube defined in the chromosome.

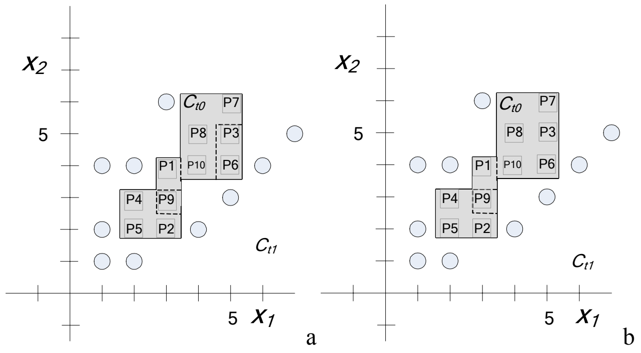

**Figure 12.** (a) The effect in the regions defined by the groups in the chromosome before mutation and (b) after mutation.

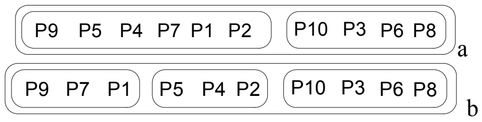

**Figure 13**. (a) Chromosome before mutation and (b) after mutation by combining two groups.

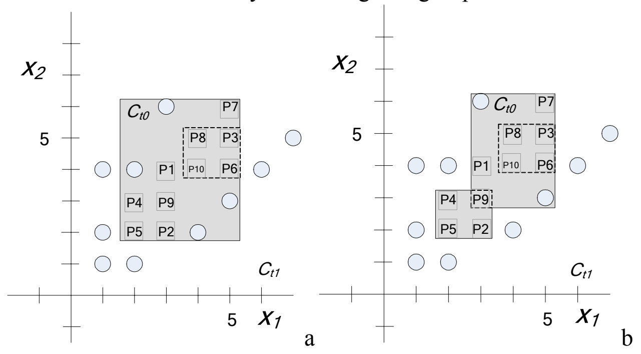

**Figure 14.** (a) Graphical effect of mutation in the regions defined by the groups in the chromosome before mutation and (b) after mutation.

The maximum values for each dimension of the hypercube will be used as the set of weights for one of the neurons that will be added in the first layer, as shown in Figure 16a. All the pre-synaptic values for this neuron will be +1, the post-synaptic value will be +1, and the maximum operator will be the used by the neuron. On the other hand, the minimum values for each dimension of the hypercube will be used as the set of weights for the second neuron that will be added in the first layer, as shown in Figure 16b. All the pre-synaptic values for this neuron will be -1, the post-synaptic value will be +1, and the operator used by the neuron will be the maximum operator. These two neurons will be connected to a new neuron in the second layer. All the weights for the neuron in the second layer will be 0, the pre-synaptic values will be +1, the post-synaptic value will be +1, and the operator used will be the maximum operator. Figure 17 shows the resulting neural network for the first hypercube.

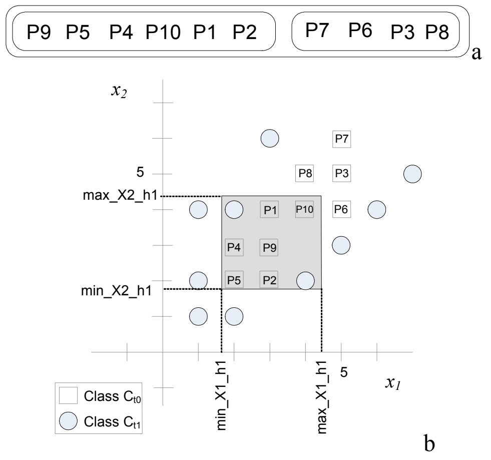

**Figure 15.** (a) An organism encoded into a chromosome and (b) the corresponding hypercube for the first group defined in the chromosome.

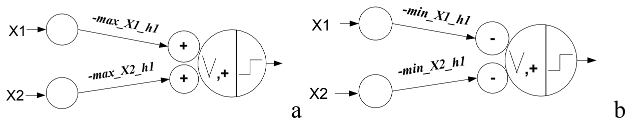

**Figure 16.** (a) Upper-right corner of the hypercube and (b) lower-left corner of the hypercube.

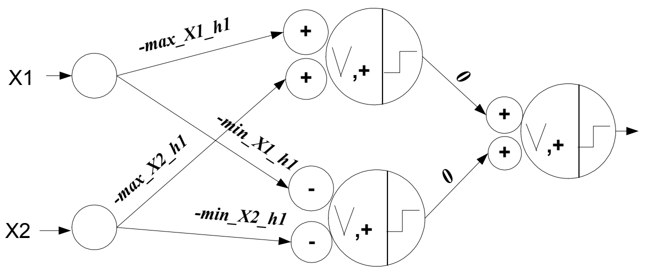

**Figure 17**. Neural network for a single hypercube.

The boundaries of the second hypercube will be used to build another branch of the neural network that will be added to the final neural network. Figure 18 shows the region defined by the second hypercube enclosing all the elements of the second group of the chromosome, and Figure 19 shows the neural network that corresponds to the network that defines that particular hypercube. As can be seen in Figure 20, the first two neural networks are combined with an additional neuron in the third layer. All the weights for the last neuron will be 0, and the pre-synaptic and post-synaptic values for this neuron will be +1. The third layer morphological neuron will use the minimum operator.

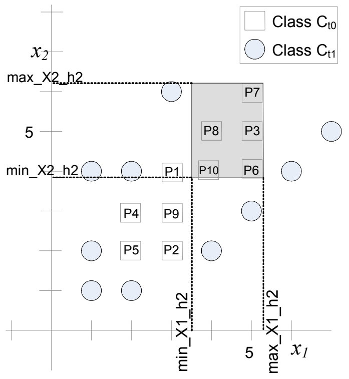

**Figure 18.** Region defined by the second group in the chromosome.

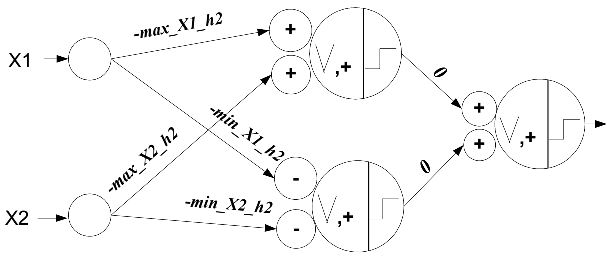

**Figure 19**. Resulting neural network for the second hypercube.

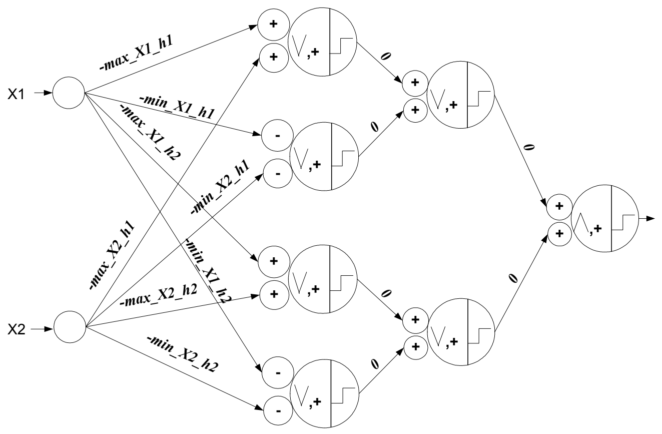

**Figure 20**. Final neural network for the chromosome defined in Figure 15a.

## Evaluation Function

Each organism must be evaluated according to the features it has and only those organisms that have the desired features will survive and mate other organisms in order to transmit their own characteristics to the future generations.

One of the most important factors to take in consideration must be the number of misclassified patterns. Another objective is to reduce the network complexity by using the minimum number of neurons needed to classify all the patterns correctly. This can be achieved by determining the minimum number of hypercubes necessary to enclose all test patterns. When a hypercube is added or removed from the chromosome, the architecture of the neural network changes. Changes are limited to the architectural constrains previously established. New neurons are added or removed from the first and second layer of the network as a hypercube is added or removed, respectively.

The fitness function is defined in Equation 3:

$$f(o) = \frac{1}{(k+1)^2 \cdot l} \qquad (3)$$

where $k$ is the number of patterns incorrectly classified. The value of $l$ represents the number of neuron groups defined in the chromosome. Each neuron group consists of three neurons as shown in Figure 19. The fitness function allows to minimize the number of misclassified patterns as well as the number of hypercubes or neurons used to solve the problem.

## Selection

A selection process is used to allow organisms who have higher fitness to transmit their features with higher probability than those who have a lower fitness. In order to consider that an organism is able to transmit their characteristics to future generations, the best 50% of the population that meets the requirements is selected. This accelerates the convergence reducing those members of the population that are not desirable. Wheel roulette is used to select the group of organism that will become parents for the next generation. The probability of an organism to be selected is equal to the fitness of the organism divided by the total fitness of all the organisms.

## Results

Several tests were conducted using 2-, 3-, and 4-dimensional spaces. Experimental results show that in all of the performed tests 100% of the patterns used for training were classified correctly. Twenty organisms were used as the initial population, and convergence was reached typically in 100 iterations. However, network topology improves with more iterations reducing the number of redundant neurons while optimizing the fitness function.

The algorithm was tested using the Iris Fished Data. The Iris Fisher Data is a set that consists of 150 patterns divided equally among three classes. Half of the test patterns were used to train the system obtaining a 100% correct classification. The other half of the patterns were used to test the network obtaining up to 96% correctly classified patterns.

Figure 21 shows an example of how the algorithm selected the decision boundaries for a set of patterns presented by Sussner (Sussner 1998).

## Conclusion

This paper presented an indirect encoding evolutionary training algorithm to obtain multiple layer morphological perceptron parameters. The algorithm looks for a set of network weights in a reduced space simplifying and accelerating the convergence of the problem. The training algorithm determines the necessary number of neurons, using up to three layers, required to perform the pattern classification. The evolutionary training allows to obtain different architectural solutions for the same problem and the fitness function searches for the smaller number of neurons needed to solve the problem. The algorithm allows the training of networks for multidimensional data sets such as the Iris Fisher Data. Different solutions for the same problem can be accomplished using this method.

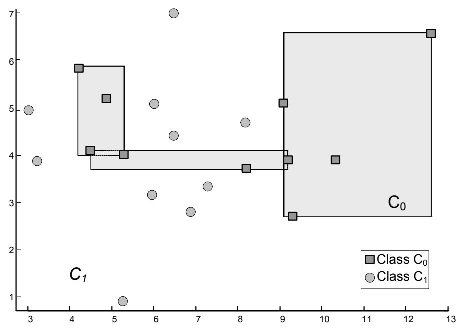

**Figure 21.** Decision boundaries found by the learning algorithm.

## References

Ritter, G.X., Beavers, T., 1999. Introduction to Morphological Perceptrons, In *Proceedings of the 1999 Artificial Neural Networks in Engineering Conference*, 173-178. St. Louis, MO.

Ritter, G.X., Sussner, P., 1997. Morphological Perceptrons, *Proceedings of International Conference on Intelligent Systems and Semiotics - A Learning Perspective*, Gaithersburg, Maryland.

Ritter, G.X., Sussner, P., 1996 An Introduction to Morphological Neural Networks, *Proceedings of the 13th International Conference on Pattern Recognition*, 709-717. Austria

Sussner, P., 1998. Morphological Perceptron Learning, In *Proceedings of the 1998* Joint Conference on the Science and Technology of Intelligent Systems, Gaithersburg, MD.

Yao, X., 1999. Evolving Artificial Neural Networks, In *Proceedings of the IEEE*, 1423-1447.