DIRECT ENCODING EVOLUTIONARY LEARNING ALGORITHM FOR MULTILAYER MORPHOLOGICAL PERCEPTRON

Jorge L. Ortiz and Roberto Piñeiro

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

ABSTRACT

This paper presents a method based on evolutionary computation to train multilayer morphological perceptron (MLMP). The algorithm calculates network parameters such as its connection weights, pre-synaptic and post-synaptic values for a given network topology. Morphological perceptron are a new type of feed-forward artificial neural network based on lattice algebra which can be used for pattern classification. The representation scheme is based on a tree data structure which the algorithm to perform operations such as crossover by replacing or switching whole nodes between parent. Adaptive mutation is used as the genetic algorithm approaches convergence to fine tune network parameters final values. The algorithm uses a special fitness function based on the mean square error of the pattern classification and introduces a penalty function to reduce the number of redundant neurons in the resulting neural network.

1. INTRODUCTION

Multilayer morphological perceptron are excellent pattern classifiers characterized by simpler network architectures and faster convergence in the learning algorithms. Ritter and Sussner [1] presented an algorithm to train single-layer morphological perceptrons on $\Re^n$ space. Sussner [2] proposed a different algorithm which could train two layers morphological perceptrons. Contrasting from the classical perceptron learning rule, which may never converge, these algorithms converge in a finite number of steps. At this time, no additional algorithms have been proposed to train multilayer morphological perceptrons. Some hybrid learning algorithms using genetic algorithms have been presented, but no comprehensive approach to use genetic algorithm have been presented.

This paper proposes an alternative training method for one and/or two layers morphological perceptron architectures based on evolutionary computation. The algorithm presented in this article introduces the use of genetic evolution and adaptive mutation techniques to the evolution of morphological neural networks connection weights as well as other network parameters.

2. MORPHOLOGICAL PERCEPTRON

Morphological Neural Networks (MNN) are a new type of neural networks introduced by Ritter, et.al. [1], [2]. [3], [4]. These neural networks replace the classical operations of multiplication by the addition, and replace addition in the traditional neuron by maximum or minimum operations. The maximum and minimum operations perform nonlinear operations before the application of the transfer function resulting in properties completely different from those properties from traditional neural networks. MNN utilize algebraic lattice operations structure known as semi-ring $(\Re_{\pm\infty}, \vee, \wedge, +, +')$ contrasting from traditional neural networks which are based on the algebraic structure known as ring $(R,+,\times)$. The operations $\wedge$ and $\vee$ denote binary operations of minimum and maximum, respectively. The topology of the single-layer morphological perceptron (SLMP) is similar to the traditional architecture of the single-layer perceptron. The output of the SLMP is computed using Equation 1

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

where x_i denotes the value for the i-th neuron, w_{ij} denotes the synaptic strength between the i-th neuron and the j-th neuron, $r_{ij} \in \{-1,+1\}$ denotes the excitatory or inhibitory pre-synaptic influence of the i-th neuron on the j-th neuron, and $p_j \in \{-1,+1\}$ denotes the post-synaptic response of the j-th neuron to the total input received. Figure 1 shows the computational model used by a morphological neuron and a decision boundary in a $\Re^2$ dimensional space.

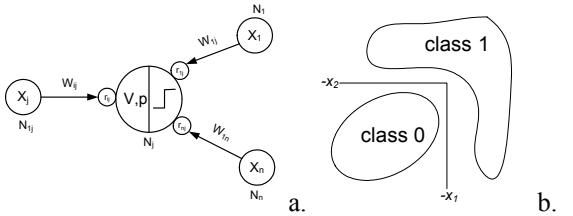

Figure 1. (a) Computational Model for Morphological Neural Network **(b)** Morphological Perceptron Decision Boundary.

The activation function *f* applied by the morphological perceptron is a Heaviside or hard limit function shown in Equation 2.

$$f:\Re \to 0,1$$
$$x \to \begin{cases} 1 \text{ if } x > 0 \\ 0 \text{ if } x \le 0 \end{cases} \quad (2)$$

3. EVOLUTIONARY COMPUTATION

Evolutionary Computation [5] is based on the idea that basic concepts of biological reproduction and evolution can be used as a model to solve computer based problems. Darwinian evolution is a robust search and optimization mechanism. The most important areas of investigation in simulated evolution include evolution strategies, evolutionary programming and genetic algorithms. Evolutionary algorithm's three main operators are selection, recombination and mutation. A population of possible solutions is maintained and encoded into data structures that represent chromosomes of an organism. Elements of that population can then mate, mutate, in other words, reproduce and evolve, directed by the fitness measure that evaluates the quality of the population with respect a goal. Genetic algorithms are a robust search and optimization mechanism which can be applied to problems where heuristic solutions are not available.

3.1. Evolutionary Artificial Neural Network

Evolutionary Artificial Neural Networks can be seen as dynamic systems that adapts the architecture and connection weights by itself. According to Yao [5], evolution in artificial neural networks can be found at three different levels: connection weights, architectures, and learning rules.

The training of a morphological neural network could be described as the search for the set of weights, presynaptic and post-synaptic response, and neuron operations for a given network architecture. The training of connection weights may be seen as the minimization of an error function by adjusting its weights interactively. Mostly mean square error of target and actual outputs, averaged over all the test patterns is used as the fitness function.

3.2. Evolutionary Morphological Perceptron

Evolutionary Morphological Perceptron is used to define the set of optimal connection weights, synaptic values, and neuron operations for a pre-defined architecture. The algorithms are able to train one- and two-layers morphological perceptrons. The algorithm requires the number of neurons to be used in the first layer. All the outputs from the first layer are passed as inputs for the neurons in the second layer, which consists of a single morphological neuron.

3.2.1. Encoding of the Organism

The neural network is encoded directly into the chromosome, which means that all the information about the architecture, weights, and synaptic response are represented in the chromosome. The multilayer morphological perceptron is encoded into the chromosome using a tree structure, where each node of the tree represents a neuron, and each branch represents the relationship between two neurons. The terminal nodes of the tree data structure represent the input layer of the network. The tree structure representation was selected because it perfectly matches the topology of the morphological perceptron. In this case, each neuron has only one output. Figure 2 shows an example of how a 2-layers morphological perceptron may be encoded into a tree data structure.

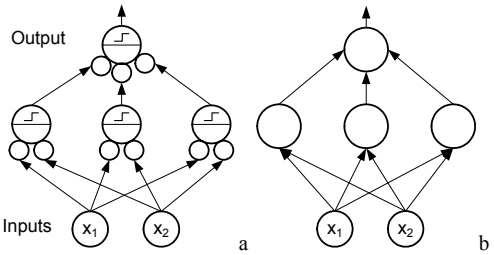

Figure 2. (a) Morphological Neural Network. **(b)** Corresponding Representation as a Tree Data Structure.

Each node contains special registers where a reference of the inputs nodes is maintained. The connection weights and synaptic response for each connection are saved into these registers in addition to the type of operation computed by the neuron (maximum or minimum). Since the same hard-limiting function is used for all the neurons in the neural network, it is not necessary to encode it into the chromosome.

As a general rule for the representation of the networks into tree structure, all the layers must have at least two or more nodes, except for the output layer, which must consist of one node. Due to the nature of the computational model the number of inputs for the nodes on each level must be at least two. Since the architecture is fixed and the tree structure must be balanced, every node in the same layer must have the same number of inputs, but the number of inputs may vary from layer to layer.

3.2.2. Selection

The selection is applied to the current population in order to create an intermediate population. In the intermediate population recombination and mutation are applied to create the next population. Rank selection is used to select the group of individual that will become parents for the next generation. Rank selection ranks each individual and a fitness value is assigned according to the rank it receives. The worst individual receives the value of 1, the second worst receives the value of 2, etc. and the best individual receives the value of *N* (the number of individuals in the population). The probability of an individual to be selected is equal to the fitness of the individual divided by the total fitness of all the individuals.

3.2.3. Recombination

The recombination is performed combining the genetic information from two parents in order to produce two offspring. After the two parents (parent 1 and parent 2) are selected, a crossover point is randomly chosen, as shown in Figure 3. The crossover produces different offspring depending on the crossover point. The crossover point could be the root node (the output node), or a terminal node, but it must be at the same level in both parents.

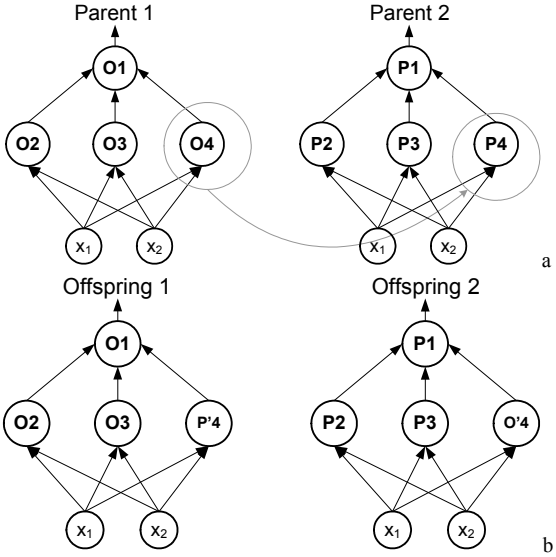

Figure 3. Crossover. **(a)** Initial Parents. **(b)** New Individuals Formed using Syntactically Constrained crossover

If the selected crossover point is the root node, of the connecting nodes may be combined between the parents, and not just the information contained in the root node. *Arithmetic crossover* may be used for to combine the floating point values of the connection weights, in addition the information represented as integers.

This network representation using a tree structure format allows the algorithm to perform operations such as crossover replacing or switching whole neurons between parent networks.

3.2.4. Mutation

Due to the nonlinear nature of the fitness function, adaptive mutation is used in this implementation. The mutation is applied in two different ways. In the first case, a node is selected randomly, and then some of the information in the registers for that node is modified according to specific probabilities. The weights are adjusted by adding or subtracting random values in a predefined range.

On the other hand, as the fitness of the best organism reaches a threshold, the mutation probabilities of the most of the MNN parameters are reduced to minimal values (close to 0%), with the exception of the connection weights mutation probability, which remains unchanged. In addition, the range in which connection weights are changed is reduced. This adaptive mutation approach reduces the chance of an organism to mutate as the problem starts to converge. This is very important in this type problem due the nonlinearity and discontinuity of the fitness function and the neural computational model.

3.2.5. Evaluation Function

The fitness function for this application has different components, such as the Mean Square Error of the classified patterns and other parameters related to the network architecture. The ideal scenario would be to get the same number of decision boundaries and number of neurons to be the same. This means optimum performances is obtained avoiding possible decision boundaries overlapping.

The fitness function for an organism that decodes into a single layer morphological perceptron is evaluated according to the performance in classification of the data set used during the training, based on the Mean Square Error (MSE) shown in Equation 3.

$$MSE = \frac{1}{N} \cdot \left[\sum_{i=1}^{N} (y_i - d_i)^2 \right] \qquad (3)$$

where N is the total number of patterns used during the training, y_i is the class where pattern x_i belongs and d_i is the class assigned by the neural network.

For two layers morphological perceptron, the organism is evaluated according to the performance in classification in addition to a penalty assigned to the number of redundant perceptrons in the network. A perceptron p_1, shown in Figure 4a, is considered to be redundant in relation to perceptron p_2, shown in Figure 4b, if the region defined by the perceptron p_1 in the $\Re^n$ space is equal to the region produced by a perceptron in the second layer which receives the output from perceptrons p_1 and p_2 as inputs, as shown in Figure 4c. The resulting fitness function used to evaluate the individuals is shown in Equation 4.

$$fitness = k_1 \cdot (1 - MSE) + k_2 \cdot p / C_2^t \qquad (4)$$

where weighting factors $k_1 = 1/3$, $k_2 = 2/3$ are used. N is the total number of test patterns, t is the total number of neurons in the first layer, p is the number of correctly placed boundaries, and $C_m^n = n!/[m!(n-m)!]$, which represent total possible neuron-boundary combinations.

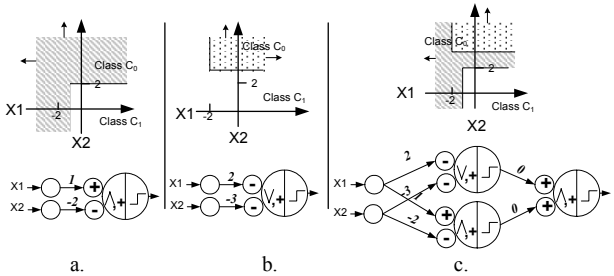

Figure 4. Redundant perceptrons. Region produced by two perceptrons (a) and (b), are combined (c). The resulting region does not differ from region defined by the second perceptron

3.3. Classification of Patterns into Multiple Classes

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:

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

A neural network may be trained for each entry in the classification vector. To defining 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 5a shows the set of test patterns, and their corresponding binary vector. Figure 5b 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 5c shows the test patterns must be regrouped in order to build the neural network for the second entry in the binary vector.

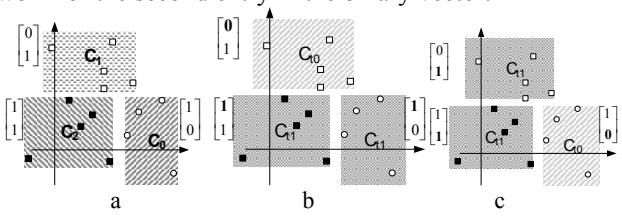

Figure 5 Distribution of patterns into temporary groups used during the training process.

4. EXPERIMENTAL RESULTS

Several tests were conducted using 2-, 3- and 4- dimensional spaces. Figure 6 shows the results of the 2-dimensional case and its corresponding network architecture. In the figure, patterns from class C_0 are represented by the plus sign ('+') and class C_1 patterns are represented by circles ('o'). Patterns misclassified from the class C_0 are represented by crosses ('x') and patterns misclassified from class C_1 are represented by stars ('*').

The probabilities used for the genetic operators were: 80% for crossover and the mutation rate was dynamically adjusted(3%-30%) Experimental results show that in most of the performed tests at least 95% of the patterns were classified correctly. The algorithm was tested using the Iris Fished Data. The Iris Fisher Data set consists of 150 patterns of divided equally among three classes. Each class has 4 different attributes. Half of the test patterns were used to train the neural network, and the other half of the patterns were used to test the predictions of the resulting neural network. The resulting neural network was able to classify correctly 99% of the patterns used during the training process in most of the tests. The resulting neural network was able to predict correctly the second half of the patterns with an accuracy of 96%. The network configuration consists of a two-layer morphological perceptron, with two neurons in the first layer and one neuron in the second layer.

5. CONCLUSION

This paper presented a comprehensive evolutionary training algorithm to obtain multiple layer morphological neural networks parameters. The algorithm calculates network parameters such as weights, pre-synaptic and post-synaptic, as well as the maximum and minimum operators. 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. More tests will be conducted and genetic algorithms techniques will be applied to evolve other network characteristic to improve its performance and generate different architectures to solve classification problems.

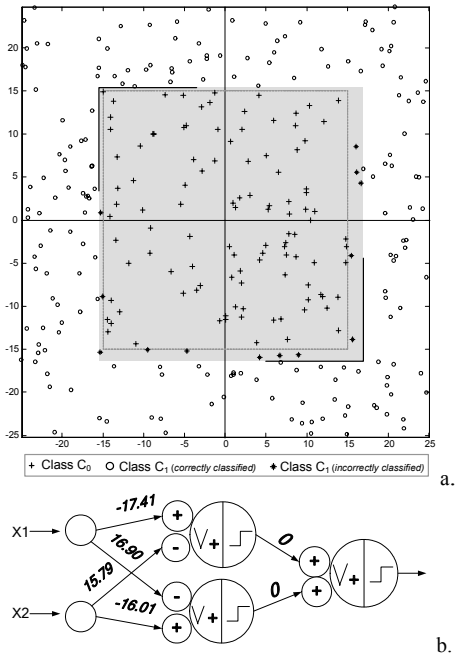

Figure 6. (a) 2-Dimension Problem and the Corresponding Architecture Data **(b)** Two Perceptrons are used in the First Laver to Define its Boundaries.

6. REFERENCES

[1] G.X. Ritter, P. Sussner, An Introduction to Morphological Neural Networks, *Procs. 13th International Conference on Pattern Recognition*, Austria, April 1996, 709-717.

[2] P. Sussner, Morphological Perceptron Learning, *Procs. of the 1998 IEEISIC/CIRA /ISAS Joint Conference*, Gaithersburg, MD, Sept. 1998

[3] G.X. Ritter, T.W. Beavers, Morphological perceptrons, *International Joint Conference on Neural Networks*, 1999, 605-610.

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

[5] X. Yao, Evolving Artificial Neural Networks, *Proc. of the IEEE*, Sept. 1999, 1423-1447.