Hybrid Training of Morphological Neural Networks: A Comparative Study

Clodoaldo Ap. M. Lima, André L. V. Coelho, Mário E. S. Silva,

Ricardo R. Gudwin, Fernando J. Von Zuben

DCA-FEEC-Unicamp

P.O. Box 6101 Campinas/SP 13083-970, Brazil

{moraes,coelho,ernesto,gudwin,vonzuben}@dca.fee.unicamp.br

Abstract. *In this paper, we present a hybrid algorithm for training a morphological neural network, which combines an evolutionary programming technique with a non-linear optimization method based on gradient information. The aim behind such fusion of techniques is to properly exploit the high non-linearity features exhibited by the morphological neuron. The presented simulation results seek to demonstrate the viability of applying this new training algorithm to function approximation and classification problems, comparing its performance with the multi-layer perceptron in complex functions.*

1. Introduction

The *multi-layer morphological neural network* (hereafter, MLM) [1][2][3] can be classified as a feedforward neural network whose hidden layer is composed by *morphological neurons* and the output layer is designed as a linear combination of signals. Similarly to a multi-layer perceptron network (MLP), the training of a morphological network can be placed as an unrestricted optimization problem. The goal is to minimize a performance index, just as the *mean squared error* (MSE) $J(w^{inp}, w^{out})$, with regard to the network parameters (w):

$$J(w^{inp}, w^{out}) = \frac{1}{N}\sum_{t=1}^{N}\sum_{i=1}^{n_{out}}(y_{it} - d_{it})^2 \quad (1)$$

where N refers to the amount of training data patterns and y_{it} and d_{it} are, respectively, the actual and desired network responses at the i-th output for the t-th training pattern.

Several algorithms for training MLP networks are found in the literature. Amongst them, the *backpropagation* (BP) [4] is the most deployed in practice. This is a generalization of the least mean squared algorithm (thus, a gradient descent method) in which the weights are adjusted with regard to the cost function $J(w)$, adopting the following learning rule:

$$w_{k+1} = w_k - \eta_k g_k$$

where

$$g_k = \left[\frac{\partial J(w_{1k})}{\partial w_{1k}} \frac{\partial J(w_{2k})}{\partial w_{2k}} \cdots \frac{\partial J(w_{nk})}{\partial w_{nk}} \right]$$

and η_k is the learning rate at the *k*-th iteration. Despite its implementation simplicity, BP sometimes shows both inefficiency and inefficacy problems in the search for optimal solutions. This fact gives rise to some new extensions (e.g. adaptive backpropagation, Quickprop, and Rprop [5][12]), aiming at improving the BP performance.

The employment of *evolutionary computation* (EC) based techniques as an alternative to gradient-based training of neural nets has received much attention recently [6]. These techniques can be characterized as stochastic search methods which may be applied during the learning stage, permitting the iterative process to escape from local minima and to converge to the cost function global minimum. Yet, for some complex problems, this convergence turns to be a critical performance factor, calling for the necessity of combining EC variants with some other *gradient*-based training algorithms. Here, following the recent trend towards hybrid learning techniques [7], we focus our attention on employing an *evolutionary programming* (EP) [8] based methodology (proposed by Saranavan and Fogel [9]), combined with error backpropagation rules, for the training of both MLPs and MLMs to the tasks of 3D benchmarking functions modeling [10] and pattern classification [11].

2. Morphological Neuron Model

The model underlying the morphological neuron has its origin on the mathematical morphology theory, comprehending an image algebra whose main operations, *dilation* and *erosion*, are based on transformations derived from sophisticated segmentation and filtering methods, largely employed in the image processing field. In [1], it is shown that an image sub-algebra includes the mathematical formalization found in most of the current neural networks models. This narrow relationship between the mathematical morphology and neural nets theories has been attracting the researchers' attention for a particular and very prominent feature present in both areas: the high level degree of *non-linearity*. This property allows a single morphological neuron to have the capability of resolving some difficult non-linearly separable problems, such as the XOR [6].

In the canonical artificial neural network model, the first step in computing the next state of a neuron involves the linear operation of multiplying neural values by their synaptic strengths and adding the results. An activation function (with thresholding) usually succeeds the linear operation accomplished in the prior step, seeking to provide the network with non-linearity effects, as shown as follows:

$$x_j(t+1) = f\left(\sum_{i=1}^{n_{inp}} x_i(t).w_{ij} - \theta_j \right)$$

In morphological neural networks, addition and maximum (or minimum) replace the operations of multiplication and addition, respectively. By taking the maximum (or minimum) of sums instead of the sum of products, the morphological network computation is non-linear before thresholding. As a consequence, the properties of this new kind of network are totally different from those of traditional ones, presenting by this way novel means of being deployed and assessed. The

mathematical model of the morphological neuron [2] is described both by Eq. (2) and Fig. 1:

$$u_j = f\left(p_j \bigvee_{i=1}^{n_{inp}} r_{ij}\left(x_i + w_{ij}\right) \right) \quad (2)$$

where $\vee$ indicates the supremum operation, n_{inp} is the number of incoming signals and p_j and r_{ij} assume values in {-1,1}.

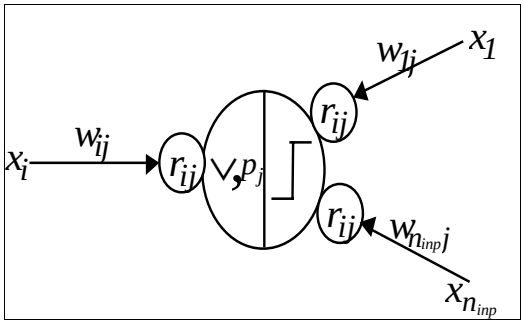

Figure 1. Morphological neuron.

As for the MLPs, it is also possible to devise a range of topologies for an MLM. Fig. 2 brings one of such structures.

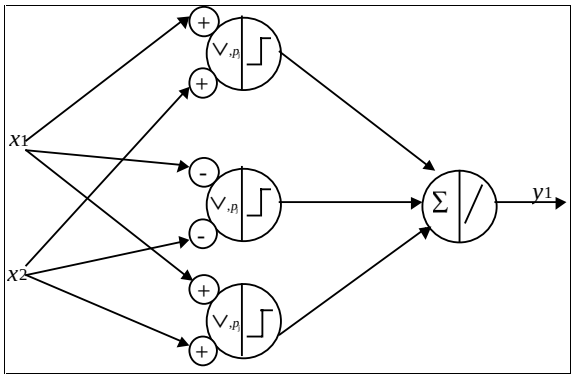

Figure 2. An MLM topology.

Traditionally, the morphological neural network has been applied to pattern recognition and memory association tasks [2][3]. In order to exploit the morphological neuron's high non-linearity, one investigates in this paper the MLM performance when applied to other classes of problems, those relating to *function approximations* and *pattern classification*. Function approximation is the task of learning or representing a function, by generating approximately the same input-output behavior based on available training data, while pattern classification deals with the categorization of a series of input data into one of *M* distinct, already available, classes [6].

In such context, the appliance of the gradient descent method brings with itself some particularities, such as to define the derivatives of the maximum and minimum functions. In [12], a detailed study is presented in this direction, having demonstrated that such functions are differentiable in almost all points of the explored *n*-dimensional space. Let $h_j = \max_i(r_{ij}(x_i + w_{ij}^{inp}))$ or $h_j = \min_i(r_{ij}(x_i + w_{ij}^{inp}))$, so it can be stated that:

$$\frac{\partial h_j}{\partial w_{ij}} = \begin{cases} r_{ij},\ if\ \ h_j = r_{ij}(x_i + w_{ij}^{inp}),\ i = 1,2,\ldots,n^{inp} \\ 0,\ otherwise \end{cases} \quad (3)$$

The error backpropagation equation for the morphological neural network is akin to the one for the MLP, given by Eq. (1). The output of each hidden layer neuron can be written as

$$u_j = f(p_j \cdot h_j)$$

where $f(.)$ refers, in our implementation, to the hyperbolic tangent function.

Finally, the gradient computation can be summarized as follows:

$$\frac{\partial J}{\partial w_{ij}^{inp}} = \sum_{t=1}^{N}\sum_{k=1}^{n_{out}} (y_{kt} - d_{kt}).w_{jk}^{out} . \frac{df}{dh_j} \cdot \frac{dh_j}{dw_{ij}^{inp}}$$

$$\frac{df}{dh_j} = (1 - u_j^2) \cdot p_j \qquad \frac{\partial J}{\partial w_{jk}^{out}} = \sum_{t=1}^{N} (y_{kt} - d_{kt}).u_j$$

3. EP for MLP/MLM Training

Evolutionary programming [8] comprehends a stochastic optimization strategy similar to genetic algorithms (GAs), which, instead of seeking to emulate specific genetic operators as observed in nature, places emphasis on the behavioral linkage between parents and their offspring. EP is a useful method of optimization when other techniques such as gradient descent or direct, analytical discovery are not possible. Combinatorial and real-valued function optimization in which the optimization surface or fitness landscape is "rugged", possessing many locally optimal solutions, are well suited for evolutionary programming.

The basic EP algorithm involves a three-step process to be repeated until a threshold for iteration is exceeded or an adequate solution is obtained: (*i*) one chooses an initial population of trial solutions at random; (*ii*) each solution is then replicated into a new intermediary population and each of the new offspring solutions are mutated according to a distribution of mutation types, ranging from a minor to an extreme value with a continuum of mutation types in between (the severity of mutation is judged on the basis of the functional change imposed on the parents); (*iii*) each offspring solution in an intermediary population is assessed by computing its fitness, which is typically done by a stochastic tournament, in such a way to determine the *N* best solutions to be retained for the next generation.

Saravanan and Fogel have devised an EP-based neurocontrol methodology [9] which was adopted here as our EC strategy to both MLP and MLM training. In such proposal, a population of neural nets is randomly initialized, each one assigned to a pair of real-valued vectors, here depicted as (w_i, σ_i), whose dimensions correspond to the number of weights (including the bias) in the fixed net structure that ought to be adjusted (n). w_i is, thus, an n-dimensional real-valued vector corresponding to object variables (weights), whereas σ_i is an n-dimensional real-valued vector corresponding to what Saravanan and Fogel called “strategy parameters”, referring to the adaptable standard deviations that determine the iterated step size updates for each connection.

Each member of the population is, in each generation, evaluated in accordance with a pre-established objective function, which, in our case, is given by Eq. (4).

$$fitness_j = \frac{1}{N}\sum_{t=1}^{N}\sum_{i=1}^{n_{out}}(y_{it} - d_{it})^2 \quad (4)$$

The generation of an offspring ($chrom_{gmut}$, σ_{gmut}) from each parent is given by the following perturbation operations (mutation):

$$chrom_{gmut}(l) = chrom_g(l) + \sigma_g(l).N_{g^l}(0,1) \quad (5)$$

$$\sigma_{gmut}(l) = \sigma_g(l).\exp(\tau^1.N_{g^l}(0,1) + \tau^2.N_{g^l}(0,1)) \quad (6)$$

$\forall\ l \in \{1,...,N\}$, where $chrom_{gmut}(l)$, $chrom_g(l)$, $\sigma_{gmut}(l)$, and $\sigma_g(l)$ denote the g-th component (parameter) of the vectors $chrom_{mut}$, $chrom$, σ_{mut}, and σ, respectively, and $N(0,1)$ refers to a unidimensional Gaussian random variable with mean zero and variance one. The authors employed the tournament process for deciding among genitors and their offspring those that best satisfies the error optimization criterion. The number of individuals (among genitors and offspring) taking part into the tournament for the evaluation of each new individual j in the population should be given (tm).

In this paper, for MLM training purposes, the p and r parameters are also evolved through the EP process. Besides this, we propose a variation to the aforementioned methodology, by following the same philosophy of the work of McLoone *et al.* [7]: To combine different, complementary techniques for improving the optimization process. Thus, the mutation operators given by Eqs. (5),(6) are enhanced with a fine-tuning process, which is done by applying a backpropagation-based correction of the weights of a new offspring for a limited, typically few, number of iterations (q). The idea is to introduce a refinement method (performed by the BP) into a global search technique, EP.

4. Simulation Results

In this section, some comparative results regarding the performance of MLPs and MLMs are shown, assessing their capabilities in dealing with function approximation and pattern classification problems. These results encompass BP, EP, and BP-EP (hybrid) training. For the purpose of function approximation, we use two complex tridimensional curves extracted from [10] (depicted in Fig. 3 and Table 1), whereas for pattern classification assessment, we employ the "two intertwined spirals" problem addressed by Lang and Witbrock [11] (Fig. 4). In the simulations, the number of neurons of the hidden layer was set to 5 to function approximation and 15 to pattern classification, the parameter q discussed in Section 3 was set to 5, the number of iterations of the backpropagation, for both MLP and MLM simulations, was 2000, the number of generations for the EP and BP-EP (for both MLP and MLM) was 1000, and the number of simulations realized for each configuration was 15. In EP and BP-EP, the tournament parameter tm was set to 10. The squashing function used in the MLP was the hyperbolic tangent and the weights, for both types of networks, are initialized randomly via a uniform distribution.

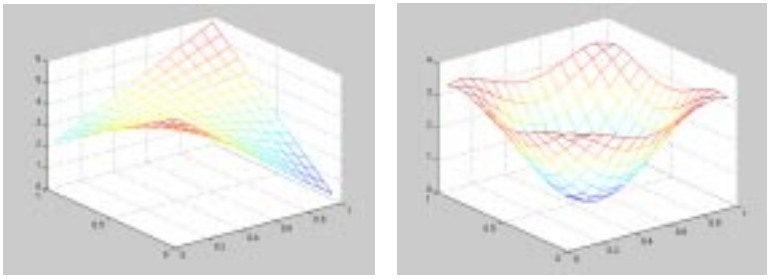

Figure 3. Benchmarking functions for the function approximation problem.

4.1. Function Approximation

The functions g^k are assumed unknown in such a way that it is only possible to obtain samples of their values for input vectors $\boldsymbol{x}$ defined to adequately cover the approximation space. Following the procedure adopted by Hwang and others [10] to produce a training data set with dimension N_{train} = 225, we generated input vectors $\boldsymbol{x}_l$ (l=1,...,225) by means of a uniform distribution of values in the range [0,1]. We repeat the same procedure for generating the test data set with dimension N_{test} = 2500.

Function	Analytic Expression
g^1	$g^{(1)}(x_1, x_2) = 10{,}391 \cdot [(x_1 - 0{,}4) \cdot (x_2 - 0{,}6) + 0{,}36]$
g^2	$g^{(2)}(x_1, x_2) = 24{,}234 \cdot [r^2(0{,}75 - r^2)], r^2 = (x_1 - 0{,}5)^2 + (x_2 - 0{,}5)^2$

Table 1. Analytical expressions of the benchmarking functions.

In the following, we present the results achieved for each type of network, MLP and MLM, trained with supervised, evolutionary and hybrid approaches, considering the above benchmarking functions. Table 2 and Table 3 bring the results for the best MSE achieved during the training an test phases.

	MLP-BP	MLP-EP	MLP-H	MLM-BP	MLM-EP	MLM-H
g^1	0.0172	0.0413	**0.0010**	0.0277	0.0643	**0.0212**
g^2	0.0129	0.0372	**0.0118**	0.0512	0.0819	**0.0389**

Table 2. Best MSE results achieved in the training phase for simulations using different learning approaches over MLP/MLM configurations.

	MLP-BP	MLP-EP	MLP-H	MLM-BP	MLM-EP	MLM-H
g^1	0.0243	0.0438	**0.0082**	0.0286	0.0685	**0.0308**
g^2	0.0202	0.0433	**0.0221**	0.0532	0.0942	**0.0395**

Table 3. Best MSE results achieved in the test phase for simulations using different learning approaches over MLP/MLM configurations.

4.2. Pattern Classification

For the task of classification, we used the "two-spirals" problem depicted in Fig. 4, which consists of two intertwined rings whose equations are given below:

Spiral #1	Spiral #2
x = 1 + (r +0.1) * cos(t)	x = 1 - (r +0.2) * cos(t)
y = 1+ (r+0.1) * sin(t)	y = 1- (r+0.2) * sin(t)

where *r* is defined in the range [0-1] and *t* is defined in the range [0-3π]. The idea is to categorize input patterns into one of two classes (50% of patterns should belong to each class). For simulation purposes, we followed the same procedure as applied to the function approximation problem for the generation of training (N_{train} = 194) and test (N_{test} = 1024) data sets.

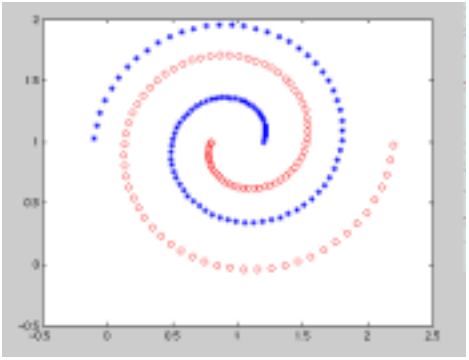

Figure 4. The "two intertwined spirals" problem.

In the sequence, we present the results achieved for each type of network, MLP and MLM, trained with supervised, evolutionary and hybrid approaches, considering the above benchmarking functions. Table 4 and Table 5 bring the best percentage of correct classification results (PCC) achieved during the training an test phases.

MLP-BP	MLP-EP	MLP-H	MLM-BP	MLM-EP	MLM-H
86,15	84,30	**97,15**	84,12	82,15	**96,65**

Table 4. Best PCC results achieved in the training phase for simulations using different learning approaches over MLP/MLM configurations.

MLP-BP	MLP-EP	MLP-H	MLM-BP	MLM-EP	MLM-H
85,43	83,90	**91,80**	83,18	80,78	**90,49**

Table 5. Best PCC results achieved in the test phase for simulations using different learning approaches over MLP/MLM configurations.

4.3. Discussion

Assessing the results, we can see that employing only supervised training for MLMs in function approximation problems shows relatively poor efficiency if the equations presented in [1][2] are maintained. Perhaps, this inefficiency owes to the fact that the values of *r* and *p*, as defined for the appliance of BP, are not updated through the synaptic weights adjustment, in view of the restriction that they can only assume integer values. Besides this, as there exists a maximum/minimum operator before the application of a non-linear function (turning the error surface more complex), in order to backpropagate the error up to the input weights, only a component of the weight vector is updated for each presented pattern sample. Another possible limitation factor was the simple modeling of the derivative operator applied over maximum/minimum operators (see Eq. (3)) in the BP case; future work should concentrate on comparing such modeling with others available in literature [12]. In the attempt to solve such deficiencies of supervised training, the application of an EC algorithm was investigated as a complementary technique for MLMs.

Moreover, the strategy of employing only the EP algorithm as the learning procedure was hampered by the inefficiency behind the convergence process, which, for several times, surpassed the allowed number of search epochs. In order to hasten this convergence rate, supervised training was employed (in a second phase) over the new individuals generated by the application of the mutation operators, tuning, by this manner, their features according to the problem in hand.

Through the presented simulations, it can be observed that the hybrid technique has produced far better results in the g^1 and g^2 cases, both for MLP and MLM configurations. MLP has shown better performance than MLM maybe because those curves have few non-smooth features, the case in which MLM is more apropos to be applied to. Nevertheless, the results already point to the viability of applying MLMs for the mapping of complex functions, mainly those combining *min*, *max*, or modular operations. For the classification problem, MLM has shown results equivalent to those provided by the MLP, and the employment of the hybrid approach has improved both network configurations as well.

5. Final Remarks

In this work, we present a hybrid algorithm for training morphological and multi-layer perceptron neural networks, which combines an evolutionary programming technique with a non-linear optimization method based on gradient information. The aim behind such fusion of techniques is to properly exploit the high non-linearity features exhibited by the morphological neuron as well as to improve the convergence of MLPs.

As future work, it would be interesting to evaluate the possibility of employing real values for the *r* and *p* parameters, which could lead to a better understanding of the dynamics of these variables in the MLM learning process. As well, in order to avoid the cases where low x_i values are precluded by high w_{ij} values, Eq. (2) might be expanded to incorporate new pondering factors (such as *a* below) whose values could also be fixed by the evolutionary process:

$$x_j(t+1) = f\left(p_j \bigvee_{i=1}^{n_{inp}} r_{ij}\left(a_{ij}x_i(t) + w_{ij}\right)\right)$$

Finally, as a direct result of the analysis presented here, it is necessary to highlight that, despite the high modeling power behind the non-linear characteristics of the morphological neuron (in relation to the traditional Perceptron), the process of training an MLM is extremely complex, requiring much computational effort. This justifies the development of alternate training approaches, such as the hybrid one proposed here.

6. References

[1] G.X. Ritter and P. Sussner: "An Introduction to Morphological Neural Networks", *Procs. of the 13th International Conference on Pattern Recognition*, Vol. IV, Track D, pp. 709-717, Austria, April 1996.

[2] P. Sussner, "Morphological Perceptron Learning", *Procs. of the 1998 IEE ISIC/CIRA/ISAS Joint Conference*, Gaithersburg, MD, September, 1998.

[3] G.X. Ritter and P. Sussner, "Morphological Associative Memories", *IEEE Transactions on Neural Networks*, Vol. 9, No. 2, pp. 281-293, March 1998.

[4] Y. Chauvin and D.E. Rumelhart (eds.), "Backpropagation: Theory, Architecture, and Applications", Lawrence Erbaum Associates Publishers, 1995.

[5] S. Fahlman, "Fast Learning Variations on the Backpropagation: An Empirical Study", *Procs. 1988 Connectionist Models Summer School*, D. S. Touretzky, G. Hinton, T. Sejnowski, pp. 38-51, 1988.

[6] K. Mehrotra, C. Mohan, and S. Ranka, "Elements of Artificial Neural Networks", MIT Press, 1997.

[7] S. McLoone, M.D. Brown, and G. Irvin, "A Hybrid Linear/Nonlinear Training Algorithm for Feedforward Neural Networks", *IEEE Transactions on Neural Networks*, Vol. 9, No. 4, pp. 669-684, July 1998.

[8] D.B. Fogel and L. J. Fogel, "An Introduction to Evolutionary Programming", *LNCS* Vol. 1063, pp. 21-33, 1996.

[9] N. Saravanan and D.B. Fogel, "Evolving Neural Control Systems", *IEEE Expert*, Vol. 10, No. 3, pp. 23-27, June 1995.

[10] J.-N. Hwang, S.R. Lay, M. Maechler, R.D. Martin, and J. Schimert, "Regression Modeling in Back-propagation and Projection Pursuit Learning", IEEE Transactions on Neural Networks, Vol. 5, No. 3, pp. 342-353, 1994.

[11] K.J. Lang and M. J. Witbrock, "Learning to Tell Two Spirals Apart", Procs. of the 1988 Connectionist Models Summer School, pp. 52-59, Morgan Kaufmann, June 17-26, 1988.

[12] X. Zhang, C.-C. Hang, "The Min-Max Function Differentiation and Training of Fuzzy Neural Networks". *IEEE Transactions on Neural Networks*, Vol. 7, No. 5, pp. 1139-1150, September 1996.