# Design of Artificial Neural Networks Using Differential Evolution Algorithm

Beatriz A. Garro[1], Humberto Sossa[1], and Roberto A. Vázquez[2]

[1] Centro de Investigación en Computación – IPN
Av. Juan de Dios Batiz, esquina con Miguel de Othon de Mendizábal
Ciudad de México, 07738, México
[2] Escuela de Ingeniería – Universidad La Salle
Benjamín Franklin 47 Col. Condesa CP 06140 México, D.F
bgarrol@ipn.mx, hsossa@cic.ipn.mx, ravem@ipn.mx

**Abstract.** The design of an Artificial Neural Network (ANN) is a difficult task for it depends on the human experience. Moreover it needs a process of testing and error to select which kind of a transfer function and which algorithm should be used to adjusting the synaptic weights in order to solve a specific problem. In the last years, bio-inspired algorithms have shown their power in different non-linear optimization problems. Due to their efficiency and adaptability, in this paper we explore a new methodology to automatically design an ANN based on the Differential Evolution (DE) algorithm. The proposed method is capable to find the topology, the synaptic weights and the transfer functions to solve a given pattern classification problems.

## 1 Introduction

Differential Evolution (DE) is an algorithm based on the classical steps of the Evolutionary Computation. However, it does not use binary encoding as an ordinary genetic algorithm; DE uses real number vectors, [1]. It does not use a probability density function to self-adapt its parameters as an Evolution Strategy [2]. Instead, DE performs mutation based on the distribution of the solutions in the current population. In this way, search directions and possible step-sizes depend on the location of the individuals selected to calculate the mutation values [3].

A feed-forward artificial neural network (ANN) is a powerful tool widely used in the field of pattern recognition and time series analysis. However, despite their power in some practical problems, ANNs cannot reach an optimum performance in several non-linear problems. This fact is caused because the parameters, used during learning phase such as learning rate, momentums, among others, do not allow compute the best set of synaptic weights.

Several works that use evolutionary strategies for training ANNs have been reported in the literature. Refer for example to [4], [5] and [6]. In general, the authors present modified PSO algorithms as an alternative for training an ANN [7], [8] and [9]; however most of the research is focused only in the evolution of the synaptic weights and sometimes in the optimum selection of the number of neurons in hidden layers [10] and [11].

K.W. Wong et al. (Eds.): ICONIP 2010, Part II, LNCS 6444, pp. 201–208, 2010.

DE algorithm has been less used in this kind of work. For example, in [12] and [13] the authors proposed a modified DE algorithm for the training of a multilayer neural network. In [14] three neural networks' architectures with different training techniques are evaluated and trained with a DE algorithm applied to a forecasting weather problem.

In this paper we explore a new methodology to automatically design an ANN based on a DE algorithm. This research includes not only the problem of finding the optimal set of synaptic weights of an ANN but also its topology and the transfer functions for each neuron. In other words, given a set of inputs and desired patterns, the proposed method will be capable to find the best topology, the number of neurons, the transfer function for each neuron and the synaptic weights in order to design an ANN that can be used to solve a given problem. Finally, the proposed method is tested with several non-linear problems and compared against PSO and back propagation algorithms.

## 2 Basics on Feed-Forwards Neural Networks

A neural network is a massively parallel-distributed processor made up from simple processing units. Each value of an input pattern $\mathbf{A} \in \mathbb{R}^N$ is associated with its weight value $\mathbf{W} \in \mathbb{R}^N$, which is normally between 0 and 1. The output of the neurons will be then performed as,

$$y = f\left(\sum_{i=1}^{N} a_i w_i + \theta\right) \tag{1}$$

where $f(x)$ is the transfer function which generates the output from the neuron.

Basically, learning is a process by synaptic weights $\mathbf{W}$ and bias levels $\theta$ of a neural network are adapted through a continuing process based on a labeled set of training data made up of $p$ input-output samples:

$$\mathbf{T}^{\xi} = \left\{\left(\mathbf{x}^{\xi} \in \mathbb{R}^N, \mathbf{d}^{\xi} \in \mathbb{R}^M\right)\right\} \forall \xi = 1, \ldots, p \tag{2}$$

where $\mathbf{x}$ is the input pattern and $\mathbf{d}$ the desired response.

Given the training sample $\mathbf{T}^{\xi}$, the requirement is to compute the free parameters of the neural network so that the actual output $\mathbf{y}^{\xi}$ of the neural network due to $\mathbf{x}^{\xi}$ is close enough to $\mathbf{d}^{\xi}$ for all $\xi$ in a statistical sense. In this sense, we might use the mean-square error given in eq. 3 as the objective function to be minimized:

$$e = \frac{1}{p \cdot M} \sum_{\xi=1}^{p} \sum_{i=1}^{M} \left(d_i^{\xi} - y_i^{\xi}\right)^2 \tag{3}$$

One of the most commonly used supervised ANN model is feed-forward network that uses backpropagation (BP) learning algorithm [15-16] to minimize the objective function given by eq. 3.

## 3 Basics on Differential Evolution

In 1995 an adaptive and efficient scheme emerged: Differential Evolution (DE) algorithm, proposed by Kenneth Price and Rainer Storn [17]. Due to its exploration capacity over a search space of a given problem, the DE algorithm avoids staying in a local optimum. It has few parameters and it converges to the optimum faster than others evolutionary techniques (the solution's representation is given by vectors of real numbers). All these characteristics convert the DE in an excellent algorithm for optimization of a complex, non-differential and non-continuous problems. [18].

The algorithm consists in randomly choosing a target vector and a base vector, in addition, two different members of the population must be randomly chosen. In order to realize the mutation is necessary to do a subtraction between these last two vectors. And then, the result is multiplied by a constant parameter denoted by F. Immediately after, the result of this operation and the base vector, chosen at the beginning of the algorithm, are summed. This new vector is called the mutated vector. At once, it is realized the crossover operation which involves a comparison (variable by variable) between the mutated vector and the target vector, creating another vector called trial vector. The comparison consists of a simple rule: If a random number is either equal or higher than a crossover rate CR it is preserved the variable of the trial vector, otherwise is preserved the variable of the target vector. Finally the last step is the selection of the vector that has to generate the new population: the trial vector or the target vector. Only the vector with the best fitness is selected. The pseudo code of "DE/rand/1/bin" is shown in the next algorithm, adapted from [3].

```
1. Randomly select two vectors from the current generation.
2. Use these two vectors to compute a difference vector.
3. Multiply the difference vector by weighting factor F.
4. Form the new trial vector by adding the weighted difference vec-
   tor to a third vector randomly selected from the current popula-
   tion.
```

## 4 Design of an ANN Using DE

In this section it is described how given a set of patterns $\mathbf{T}$, the synaptic weights, the architecture and the transfer function of each neuron of an ANN can be automatically adjusted by means of a DE algorithm.

In order to achieve this goal, we codify this information as follows: each vector will be represented by a matrix $\mathbf{x} \in \mathbb{R}^{(MNN+2)\times MNN}$ composed by three parts: topology, synaptic weights and transfer function.

The topology of the ANN is codified based on the binary square matrix representation of a graph $\mathbf{x}$ where each component $x_{ij}$ represents the connections between neuron $i$ and neuron $j$ when $x_{ij} = 1$. However, instead of evolving this binary information we decided to codify this information into its decimal base value and then evolve it. For example suppose that next binary code "01101" represents the connections of a *i*-th neuron to 5 neurons where only neurons 2, 3, and 5 are connected to *i*. This binary code is transformed into its decimal base value resulting in "13", which

will be the number that we will evolve instead of the binary value. This scheme is much faster. Instead of evolving a string of $MNN$ bits, we evolve only a decimal base number. The synapses weights of the ANN are codified based on the square matrix representation of a graph $\mathbf{x}$ where each component $x_{ij}$ represents the synaptic weight between neuron $i$ and neuron $j$. Finally, the transfer function for each neuron will be represented by an integer from 0 to 5 representing one of the 6 transfer functions used in this research: logsig, tansig, sin, gaussina, linear and hard limit. These functions were selected because they are the most popular and useful transfer functions in several kinds of problems. Figure 1 shows the individual representation of the ANN.

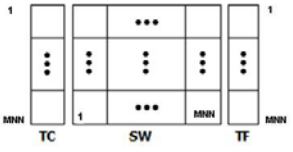



**Fig. 1.** Individual representation of an ANN. Note that TC represents the topology of the network, SW represents the synaptic weights and TF represents TF of each neuron.

Another alternative to codify the topology could be based on the synaptic weights: values less than a given threshold means no connection. However, determining the best threshold could bring other kind of problems; that is why in this paper we will not focus in this scheme.

The fitness function which measures the performance of an individual is given by eq. 3 where the output $y_i$ of the ANN is computed as follows (note that the restriction $j < i$ is used to avoid the generation of cycles in the ANN):

1) For the first $nfin$ neurons, the output $o_i = a_i$.
2) For each neuron $n_i$, $i = nfin, \ldots, MNN$.
   a) Get connections by using individual $\mathbf{x}_{1,i}$.
   b) For each neuron $j < i$ connected to $n_i$, $o_i = f(o)$ where $f$ is a transfer function given by individual $\mathbf{x}_{nfout,i}$ and $o$ is compute using eq. 1.
3) Finally, $y_i = o_i$, $i = MNN - nfouy, \ldots, MNN$.

## 5 Experimental Results

In order to evaluate the accuracy of the proposed method, several experiments were performed using 4 data sets [19]: XOR, iris plant, wine, and breast cancer.

All data sets were partitioned into two sets: a training set and a testing set. For the iris plant data set, the first 120 examples were used for the training set, and the remaining 30 examples for the testing set. For the wine set, the first 90 examples were

used for training set, and the remaining 89 examples for the testing set. For the breast cancer, the first 600 examples were used for training set, and the remaining 83 for testing set. For the XOR problem, the first 4 examples of the 2-D version were used for training, and noisy versions of these examples were used for the testing set.

The input features of all data set were rescaled in a range between $[0,1]$. The outputs were encoded encoded by the 1-to- $c$ representations for $c$ classes.

The parameters for the DE algorithm were set as follows: population size $NP$ was set to 50, the number of generations was set to 2000, the population was initialized in the range $[-5,5]$, $CR = 0.9$, and $F = rand[0.3, 0.7]$.

Ten expertiments over each data set using the same parameters were performed. It is important to notice that the topology, the transfer functions for each neuron and the set of synaptic weights for each ANN were automatically determined by the DE algorithm. Transfer functions were labeled as *logsig* (LS), *tansig* (HT), *sin* (SN), *radbas* (GS), *pureline* (LN) and *hard limit* (HL).

Fig. 2 shows 2 ANNs generated with the proposed method for the XOR problem. Something important to metion about this set of experimets is that most of the generated ANNs had the same topology and transfer function. Fig. 3 shows 2 of the 10 ANNs designed by the basic DE algorithm. As you can appreciate, the ANNs generated with DE algorithm use different transfer functions and the connections between neurons are completely different. Fig. 4 shows 2 ANNs generated with the proposed method for the wine data set. Finally, 2 of the 10 ANNs obtained by the basic DE algorithm for the breast cancer data set are shown in Fig. 5.

It is worth mentioning that topologies of the ANNs designed with the proposed method are different to the traditional feed-forward; the topologies obtained present lateral connections and connections between input and output layers. Moreover, in Fig. 5(a) we can observe that one of the features from the input pattern (neuron 7) does not contribute to the output of the ANN. This effect can be seen as a reduction of the dimensionality.

Table 1 shows the average MSE found by four algorithms during the design of the ANN (topololy, synaptic weights and transfer functions): Basic PSO, second generation of PSO (SGPSO), a modified PSO (MPSO) algorithm [11] and the basic DE algorithm. We can observe that the proposed methodology provides better results using the DE algorithm under the same conditions during training phases as equal as during testing phase. Compared against BP algorithm for an ANN composed by 3 layers, learning rate was of 0.1 for the same data sets and 2000 epochs. The proposed method provided comparable results. However, the advantage of our metodology is that we do not need to apriori select the architecture, transfer function and learning algorithm of the ANN; althoght the proposed method seems to require more epochs (2000x50) than BP (2000), it automatically designs the ANN by itself.

Table 2 shows the classification error achieved by the proposed methodology using DE and MPSO which was the PSO algorithm that provides the best results.

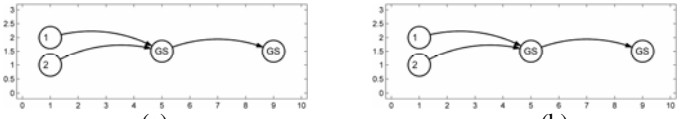
Based on the previous results we can consider DE algorithms as an alternative to automatically design an ANN using only a labeled set of training data. Furthermore, if we modify the individual coding scheme, the proposed methodology could generated other kinds of ANN as feed-forward networks (without lateral conections), radial basis NN and even recurrent NN.

**Table 1.** Comparision of different methods in terms of the minimum square error (MSE)

| DB | Basic PSO | | 2GPSO | | MPSO | | BP | | DE/rand/1/bin | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Tr. Er. | Te. Er. | Tr. Er. | Te. Er. | Tr. Er. | Te. Er. | Tr. Er. | Te. Er. | Tr. Er. | Te. Er. |
| *XOR* | 0 | 0 | 0 | 0 | 0 | 0 | 0.0097 | 0.011 | 6.1E-9 | 0.0120 |
| *Iris* | 0.202 | 0.237 | 0.182 | 0.092 | 0.057 | 0.173 | 0.0075 | 0.128 | 0.028 | 0.0146 |
| *Wine* | 0.233 | 0.305 | 0.276 | 0.058 | 0.076 | 0.295 | 0.0001 | 0.016 | 0.072 | 0.0748 |
| *Breast Cancer* | 0.032 | 0.018 | 0.325 | 0.013 | 0.035 | 0.316 | 0.0085 | 0.013 | 0.021 | 0.0068 |

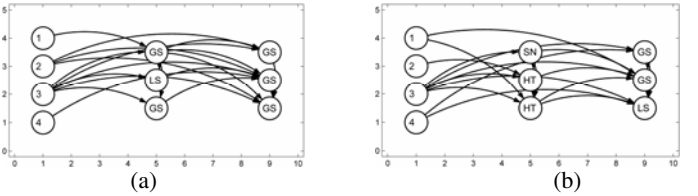
**Table 2.** Average accuracy of the proposed method in terms of the classification error (CER)

| DB | MPSO | | DE/rand/1/bin | |
|---|---|---|---|---|
| | Tr. Er. | Te. Er. | Tr. Er. | Te. Er. |
| *XOR* | 0.025 | 0.025 | 0 | 0.025 |
| *Iris* | 0.043 | 0.0165 | 0.0425 | 0.0033 |
| *Wine* | 0.201 | 0.268 | 0.0811 | 0.0764 |
| *Breast Cancer* | 0.032 | 0.014 | 0.0225 | 0.0036 |



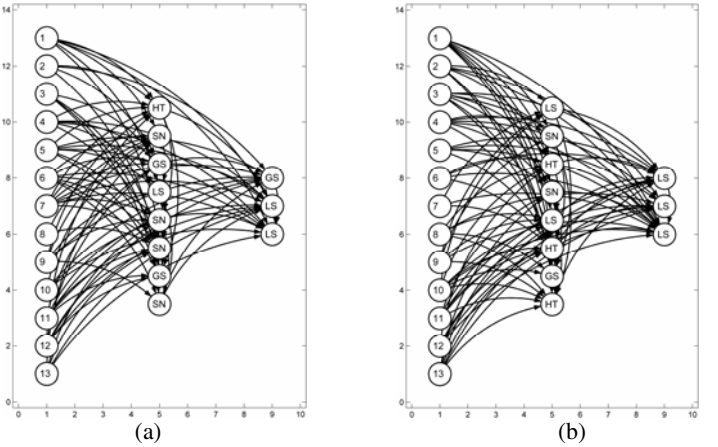

(a) (b)

**Fig. 2.** Two topologies obtained with their transfer function for each neuron using DE/rand/1/bin algorithm for the XOR problem



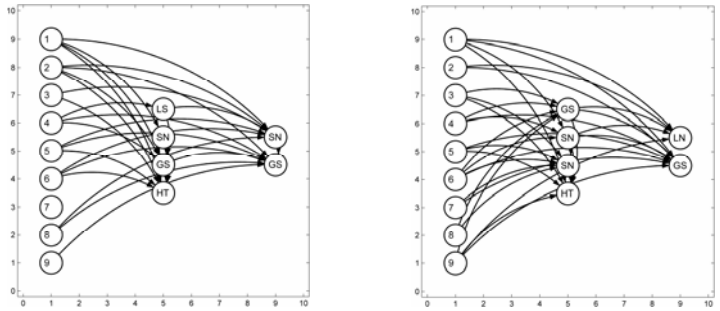

(a) (b)

**Fig. 3.** Two topologies obtained with their transfer function for each neuron using DE/rand/1/bin algorithm for the Iris database




(a) (b)

**Fig. 4.** Two topologies obtained with their transfer function for each neuron using DE/rand/1/bin algorithm for the Wine database




(a) (b)

**Fig. 5.** Two topologies obtained with their transfer function for each neuron using DE/rand/1/bin algorithm for the Cancer database

## 6 Conclusions

In this paper a methodology to automatically design an ANN was proposed. This new alternative is based on bio-inspired algorithms. Particularly in this paper, the Differential Evolution (DE) algorithm was addopted. The design of a neural network can be seen as an optimization problem, which consists on finding the best values of the synapses, the best topology and the best transfer functions for each neuron which minimize an error function. For that reason, DE algorithm is suitable to automaticly design the ANN bases on the optimization of an error function (minimization of an objective function).

The accuracy of the proposal was tested using several non-linear problems and the results show a clear advantage over traditional schemes which involve the selection of a learning algorithm, a topology and the transfer functions. Compared against BP our proposed method provides comparable results. Moreover, the results achieved using the proposed methodology combined with DE were better compared to those obtained with PSO based algorithms. As was observed in the above figures, the proposed method generates different topologies with different transfer functions and in some cases is posible to reduce the dimentionality of the input patterns.

Nowadays we are studying other kind of error measures such as fuzzy classification error [20] in order to compare different algorithms and prove the robustness of the methodology. On the other hand, we are designing a new objective function based on these error measures. Moreover, we are implementing other bioinspired techniques such as bee colony optimization to automatically desing an ANN.

**Acknowledgements.** Authors thank SIP-IPN under grant 20100468 and COFAA for the economical support. They also thank the European Union, the European Commission and CONACYT for the economical support. This paper has been prepared by economical support of the European Commission under grant FONCICYT 93829. The content of this paper is an exclusive responsibility of the CIC-IPN and it cannot be considered that it reflects the position of the European Union. We thank also the reviewers for their comments for the improvement of this paper.

## References

[1] Goldberg, D.E.: Genetic Algorithms in Search, Optimizationand Machine Learning. Addison-Wesley Publishing Co., Reading (1989)
[2] Schwefel, H.-P.: Evolution and Optimization Seeking. John Wiley & Sons, Chichester (1995)
[3] Mezura-Montes, E., Velazquez-Reyes, J., Coello Coello, C.A.: A Comparative Study of Differential Evolution Variants for Global Optimization. In: GECCO (2006)
[4] Tejen, S., et al.: A Hybrid Artificial Neural Networks and Particle Swarm Optimization for Function Approximation. In: ICIC, vol. 4, pp. 2363–2374 (2008)
[5] Chau, K.W.: Application of a PSO-based neural network in analysis of outcomes of construction claims. Automation in Construction 16, 642–646 (2007)
[6] Chatterjee, A., et al.: A Particle Swarm Optimized Fuzzy-Neural Network for Voice Controlled Robot Systems. IEEE Trans. on Ind. Electronics 52, 1478–1489 (2005)
[7] Wang, Z., et al.: Particle Swarm Optimization and Neural Network Application for QSAR. In: Proceedings 18th Parallel and Distributed Processing Symposium (2004)
[8] Zhao, L., Yang, Y.: PSO-Based Single Multiplicative Neuron Model for Time Series Prediction. Expert Systems with Applications 36, 2805–2812 (2009)
[9] Da, Y., Ge, X.R.: An improved PSO–based ANN with simulated annealing technique. Neurocomput. Lett. 63, 527–533 (2005)
[10] Yu, J., Xiand, L., Wang, S.: An Improved Particle Swarm Optimization for Evolving Feedforward Artificial Neural Networks. Neural Processing Letters 26, 217–231 (2007)
[11] Garro, B.A., Sossa, H., Vazquez, R.A.: Design of Artificial Neural Networks using a Modified Particle Swarm Optimization Algorithm. In: Proc. IEEE Int. Joint Conf. Neural Networks, pp. 938–945 (2009)
[12] Ilonen, J., Kamarainen, J.-K., Lampinen, J.: Differential Evolution Training Algorithm for Feed-Forward Neural Networks. Neural Processing Letters 17(1), 93–105 (2003)
[13] Guiying, N., Yongquan, Z.: A Modified Differential Evolution Algorithm for Optimization Neural Network. In: Proceedings of ISKE (2007)
[14] Abdul –Kader, H.M.: Neural Networks Training Based on Differential Evolution Algorithm Compared with Other Architectures for Weather Forecasting34. IJCSNS 9(3) (March 2009)
[15] Anderson, J.A.: Introduction to Neural Networks. MIT Press, Cambridge (1995)
[16] Werbos, P.J.: Backpropagation through time: What it does and how to do it. Proc. IEEE 78, 1550–1560 (1990)
[17] Storn, R., Price, K.: Differential Evolution - a Simple and Efficient Adaptive Scheme for Global Optimization over Continuous Spaces, TR-95-012, ICSI (1995)
[18] Price, K.V., Storn, R.M., Lampinen, J.A.: Differential evolution: a practical approach to global optimization. Springer, Heidelberg (2005)
[19] Murphy, P.M., Aha, D.W.: UCI repository of machine learning databases. Dept. Inf. Comput. Sci., Univ. California, Irvine, CA (1994)
[20] Mendis, B.S.U., Gedeon, T.D.: A comparison: Fuzzy Signatures and Choquet Integral. IEEE Congress on Computational Intelligence, 1464–1471 (2008)