Multilayer Perceptron: Architecture Optimization and Training

Hassan Ramchoun, Mohammed Amine Janati Idrissi, Youssef Ghanou, Mohamed Ettaouil

Modeling and Scientific Computing Laboratory, Faculty of Science and Technology, University Sidi Mohammed Ben Abdellah, Fez, Morocco

***Abstract* — The multilayer perceptron has a large wide of classification and regression applications in many fields: pattern recognition, voice and classification problems. But the architecture choice has a great impact on the convergence of these networks. In the present paper we introduce a new approach to optimize the network architecture, for solving the obtained model we use the genetic algorithm and we train the network with a back-propagation algorithm. The numerical results assess the effectiveness of the theoretical results shown in this paper, and the advantages of the new modeling compared to the previous model in the literature.**

***Keywords* — Multilayer Perceptron (MLP), Architecture Optimization, Non-Linear Optimization, Genetic Algorithm, Feed-Forward Neural Network Training.**

I. Introduction

IN recent years, neural networks have attracted considerable attention as they proved to be essential in applications such as content-addressable memory, pattern recognition and optimization.

Learning or training of ANN is equivalent to finding the values of all weights such that the desired output is generated to corresponding input, it can be viewed as the minimization of error function computed by the difference between the output of the network and the desired output of a training observations set [1].

Multilayer Perceptron is the most utilized model in neural network applications using the back-propagation training algorithm. The definition of architecture in MLP networks is a very relevant point, as a lack of connections can make the network incapable of solving the problem of insufficient adjustable parameters, while an excess of connections may cause an over-fitting of the training data [3]. Especially, when we use a high number of layer and neurons this is our case in this paper.

Optimizing the number of connection and hidden layer for establishing a multilayer Perceptron to solve the problem remains one of the unsolved tasks in this research area Multilayer Perceptron consists of input layer, output layer and hidden layers between these two layers. The number of these layers is dependent on the problem [8]. In this work, we optimize the number of hidden layers and the number of neurons in each hidden layer and process of to deal with a few connection to increase the speed and efficiency of the neural network. We model this problem of neural architecture in terms of a mixed-integer non-linear problem with non-linear constraints.

The next section present and discuss related works on neural network architecture optimization. Section 3 describes the artificial neural networks. In Section 4, we present the problem of optimization neural architecture and a new modeling is proposed. And before concluding, experimental results are given in the section 4.

II. Related Works

A number of approaches in the literature have taken account the architecture optimization. This section describes only those works that are more or less similar to our work.

Global search may stop the convergence to a non-optimal solution and determine the optimum number of ANN hidden layers. Recently, some studies in the optimization architecture problems have been introduced in order to determine neural networks parameters, but not optimally [3].

Traditional algorithms fix the neural network architecture before learning [4]. Others studies propose constructive learning [5]-[6]. It begins with a minimal structure of hidden layers; these researchers initialized the hidden layers, with a minimal number of hidden layer neurons. The most of researchers treat the construction of neural architecture (structure) without finding the optimal neural architecture [7].

T.B Ludermir et al [14] propose an approach for dealing with a few connections in one hidden layer and training with deferent hybrid optimization algorithms.

In our previous work we take account optimization of hidden layers with introducing one decision variable for layer [1], and in another work we have taken account the hidden node optimization in layers, for training this two models we have used a back-propagation algorithms [2].

III. Units Feed-Forward Neural Networks for Pattern Classification

A data set for pattern classification consists of a number of patterns together with their correct classification. Each pattern consists of a number of measurements (i.e., numerical values).

The goal consists in generating a classifier that takes the measurements of a pattern as input, and provides its correct classification as output. A popular type of classifier is feed-forward NNs [9].

A feed-forward NN consists of an input layer of neurons, an arbitrary number of hidden layers, and an output layer. Feed-forward NNs for pattern classification purposes consist of as many input neurons as the patterns of the data set have measurements, i.e., for each measurement there exists exactly one input neuron. The output layer consists of many neurons as the data set has classes. Given the weights of all the neurons connections, in order to classify a pattern, one provides its measurements as input to the input neurons, propagates the output signals from layer to layer until the output signals of the output neurons are obtained. Each output neuron is identified with one of the possible classes. The output neuron that produces the highest output signal classifies the respective pattern.

A. Multilayer Perceptron

A multilayer Perceptron is a variant of the original Perceptron model proposed by Rosenblatt in the 1950 [10]. It has one or more hidden

DOI: 10.9781/ijimai.2016.415

layers between its input and output layers, the neurons are organized in layers, the connections are always directed from lower layers to upper layers, the neurons in the same layer are not interconnected see Fig. 1.

The neurons number in the input layer equal to the number of measurement for the pattern problem and the neurons number in the output layer equal to the number of class, for the choice of layers number and neurons in each layers and connections called architecture problem, our main objectives is to optimize it for suitable network with sufficient parameters and good generalisation for classification or regression task.

B. Back-propagation and Learning for de MLP

Learning for the MLP is the process to adapt the connections weights in order to obtain a minimal difference between the network output and the desired output, for this raison in the literature some algorithm are used such as Ant colony [11] but the most used called Back- propagation witch based on descent gradient techniques [12].

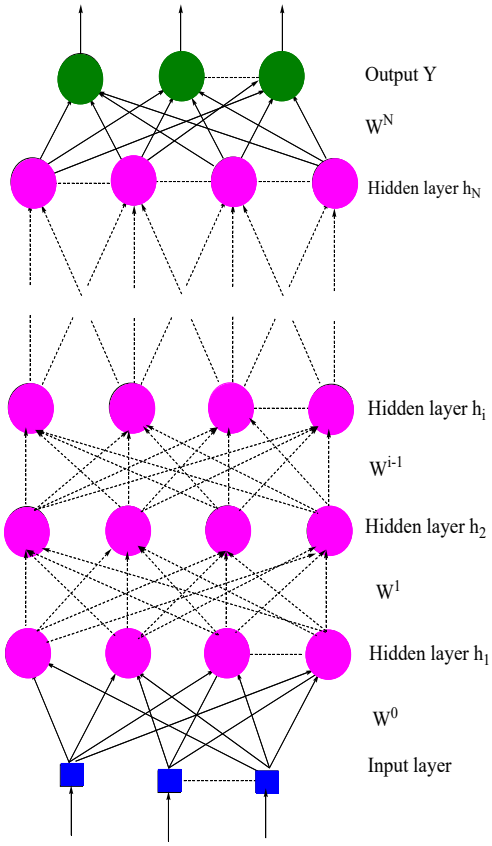

Fig 1. Feed forward neural network structure.

Assuming that we used an input layer with n_0 neurons $X = (x_0, x_1, \dots, x_{n_0})$ and a sigmoid activation function

$$f(x) = \frac{1}{1+e^{-x}} \quad (1)$$

To obtain the network output we need to compute the output of each unit in each layer:

Now consider a set of hidden layers $(h_1, h_2, \dots, h_N)$. Assuming that n_i are the neurons number by each hidden layer h_i.

For the output first hidden layer

$$h_i^j = f\left(\sum_{k=1}^{n_{i-1}} w_{k,j}^0 \, x_k\right) \quad \text{j=1,…,} n_i \quad (2)$$

The outputs h_i^j of neurons in the hidden layers are computing as flows:

$$h_i^j = f\left(\sum_{k=1}^{n_{i-1}} w_{k,j}^{i-1} \, h_{i-1}^k\right) \; i = 2, \dots, N \; and \; j = 1, \dots, n_i \quad (3)$$

Where $w_{k,j}^i$, is the weight between the neuron k in the hidden layer i and the neuron j in the hidden layer $+1$, n_i is the number of the neurons in the ith hidden layer, The output of the ith can be formulated as following:

$$h_i = {}^{t}(h_i^1, h_i^2, \dots, h_i^{n_i}) \quad (4)$$

The network output are computing by

$$y_i = f\left(\sum_{k=1}^{n_N} w_{k,j}^N \, h_N^k\right)$$
$$Y = (y_1, \dots, y_j, \dots, y_{N+1}) = F(W, X) \quad (5)$$

Where $w_{k,j}^N$ is the weight between the neuron k in the Nth hidden layer and the neuron j in the output layer, $n_N$is the number of the neurons in the Nth hidden layer, Y is the vector of output layer, F is the transfer function and W is the matrix of weights, it's defined as follows:

$$W = [W^0, \dots, W^j, \dots, W^N]$$
$$W^i = \left(w_{j,k}^i\right)_{\substack{0 \le i \le N \\ 1 \le j \le n_{i+1} \\ 1 \le k \le n_i}} \text{ where } w_{j,k}^i \in \mathbb{R}$$

To simplify we can take $n = n_i \;\; \forall i = 1, \dots, N$ for all hidden layers. Where X is the input of neural network and f is the activation function and $\bar{W^i}$ is the matrix of weights between the i^{th} hidden layer and the $(i+1)^{th}$ hidden layer for $i = 1, \dots, N-1$, W^0 is the matrix of weights between the input layer and the first hidden layer, and $\bar{W^N}$ is the matrix of weights between the *N*th hidden layer and the output layer.

IV. Proposed Model to Optimize the MLP Weights and Architectures

The MLP architecture definition depends on the choice of the number of layers, the number of hidden nodes in each of these layers and the objective function but another approach which is introduced in this paper allows to control all connections between layers and to delete some of them, when there are no connections between the node n_i and layer $i+1$ and when all neurons are deleted in layers i we delete it.

In this work, we assign to each connection a binary variable which takes the value 1 if the connection exists in the network and 0 otherwise. Also we associate another binary variable for the hidden layers.

Notation:

N: Number of hidden layers.

n_0: Number of neurons in input layer.

n_i : Number of neurons in hidden layer i.

n_{op} : Optimal number of hidden layer.

n_{N+1}: Number of neurons in output layer.

X: Input data of neural network.

Y: Calculated output of neural network.

h_i^j: Output of neuron j in hidden layer i.

f: Activation function.

d: Desired output.

u_i: Binary variable for $i = 1, \dots, N-1$. $U = (u_1, u_2, \dots, u_{N-1})$

$v_{k,j}^i$: Binary variable for $i = 2, \dots, N$, $j = 1, \dots, n_i$ and $k = 1, \dots, n_{i-1}$

We computed the output of neural network by the following formulation:

$$F(U, V, W, X) = Y = \left(y_1, y_2, \dots, y_{n_{N+1}}\right) \tag{6}$$

A. Output of first hidden layer

The neurons of first hidden layer are directly connected to the input layer (data layer) of the neural network.

The output for each neuron in the first hidden layer is calculated by:

$$h_1 = \begin{pmatrix} h_1^1 \\ \vdots \\ h_1^k \\ \vdots \\ h_1^{n_1} \end{pmatrix} = \begin{pmatrix} f\left(\sum_{k=1}^{n_0} w_{k,1}^0 x_k\right) \\ \vdots \\ f\left(\sum_{k=1}^{n_0} w_{k,j}^0 x_k\right) \\ \vdots \\ f\left(\sum_{k=1}^{n_0} w_{k,n_1}^0 x_k\right) \end{pmatrix} \tag{7}$$

B. Output for the hidden layer i = 2,..., N

To calculate the output of each neuron for the hidden layer i, where $i = 2, \dots, N$, we propose the rule:

$$h_i = \begin{pmatrix} h_i^1 \\ \vdots \\ h_i^k \\ \vdots \\ h_i^{n_i} \end{pmatrix} = (1 - u_{i-1}) h_{i-1} +$$

$$u_{i-1} \begin{pmatrix} 1 - \prod_{k=1}^{n_{i-1}} \left(1 - v_{k,1}^{i-1}\right) \\ \vdots \\ 1 - \prod_{k=1}^{n_{i-1}} \left(1 - v_{k,j}^{i-1}\right) \\ \vdots \\ 1 - \prod_{k=1}^{n_{i-1}} \left(1 - v_{k,n_i}^{i-1}\right) \end{pmatrix} \begin{pmatrix} f\left(\sum_{k=1}^{n_{i-1}} v_{k,1}^{i-1} w_{k,1}^{i-1} h_{i-1}^k\right) \\ \vdots \\ f\left(\sum_{k=1}^{n_{i-1}} v_{k,j}^{i-1} w_{k,j}^{i-1} h_{i-1}^k\right) \\ \vdots \\ f\left(\sum_{k=1}^{n_{i-1}} v_{k,n_i}^{i-1} w_{k,n_i}^{i-1} h_{i-1}^k\right) \end{pmatrix} \tag{8}$$

where $k = 1, \dots, n_{i-1}$ and $j = 1, \dots, n_i$

C. Output for the neural network (layer N+1)

The output of the neural network is defined by the following expression:

$$Y = \begin{pmatrix} y_1 \\ \vdots \\ y_k \\ \vdots \\ y_{n_{N+1}} \end{pmatrix} = \begin{pmatrix} f\left(\sum_{k=1}^{n_N} w_{k,1}^N h_N^k\right) \\ \vdots \\ f\left(\sum_{k=1}^{n_N} w_{k,j}^N h_N^k\right) \\ \vdots \\ f\left(\sum_{k=1}^{n_N} w_{k,N+1}^N h_N^k\right) \end{pmatrix} \tag{9}$$

D. Objective function

The objective function of the proposed model such as in the previous work [2] is the error calculated between the obtained output and desired output:

$$\|F(U, V, X, W) - d\|^2 \tag{10}$$

But we propose to modify this objective function and adding one term for error connections regularized by α in order to control variations of weights in training and optimization phase.

E. Constraints

- The first constraint guarantees the existence of the hidden layers.

$$\sum_{i=1}^{N} u_i \geq 1 \tag{11}$$

- These constraints insured communication between neurons, connections and layers

$$u_{i-1} \times \prod_{j=1}^{n_i} \prod_{k=1}^{n_{i-1}} (1 - v_{k,j}^{i-1}) = 0 \quad \forall i = 2, \dots, N \tag{12}$$

$$(1 - u_{i-1}) \times \sum_{j=1}^{n_i} \sum_{k=1}^{n_{i-1}} v_{k,j}^{i-1} = 0 \quad \forall i = 2, \dots, N \tag{13}$$

- The weights values are the real number.

V. Implementation And Numerical Results

To illustrate the advantages of the proposed approach, we apply our algorithm to a widely used dataset, Iris dataset for classification [13]. It consists of three target classes: Iris Setosa, Iris Virginica and Iris Versicolor. Each species contains 50 data samples. Each sample has four real-valued features: sepal length, sepal width, petal length and petal width. By doing this, all features have the same contribution to Iris classification.

For this task an MLP with sigmoid activation function and a back-propagation training after obtaining an optimal architecture by Genetic algorithm is used, in this section we present a parameters setting, implementations procedures and final results.

A. Parameters setting

We use the genetic algorithm to solve the architecture optimization problems. To this end, we have coded individual by one chromosomes; moreover, the fitness of each individual depends on the value of the objective function obtained by two terms one for network error and the second for the network connections.

The individual of the initial population are randomly generated, and u_i, $v_{k,j}^i$ take the value 0 or 1, and the weights takes random values in space $[-0.5, 0.5]$. After creating the initial population, each individual is evaluated and assigned a fitness value according to the fitness function.

The fitness suggested in our work is the following function:

$$f(i) = M - f(s) \tag{14}$$

We applied crossover and mutation operator, in this step, new individuals called children are created by individuals selected from the population called parents for more exploring the researches spaces of solution.

In Table I we present all parameters used in our experiments

TABLE I
IMPLIMENTATION PARAMETRES

Nmax	16	32	48
Pm	0.200	0.200	0.300
Pc	0.700	0.800	0.700
s	0.520	0.500	0.550
α	0.010	0.300	0.450
β	0.395	0.397	0.340

Nmax: total number of weights, Pm: Probability of mutation, Pc: Probability of crosser, α: Regularization parameters for connection error, β: Training rate, s: Threshold

B. Results for the optimization methodology

After determining the optimal number of hidden layer, in this case three, and the total number of connection. We use an architecture contains four neurons in each layer and we can initialize the neural networks by value of weights obtained with Genetic algorithm. Our algorithm tested on instances for Iris data.

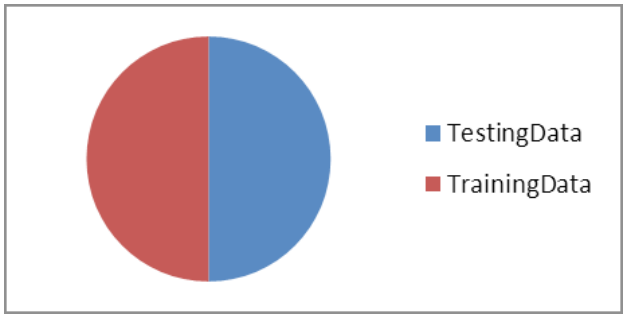

Fig.2 Partition of data base

The Table II presents the obtained clustering results of training and testing data. We remark that the proposed method permits to classify all the training data only one from Versicolor and two from the same type for testing data.

TABLE II
CLASSIFICATION FOR DATA SET (PROPOSED METHOD)

	Connect (%)	MC	Accuracy (%)
Tr.D	50	1	98.7
Tes.D	50	2	97.3

M.C: Misclassified data, Connect (%): percentage of connections weights used in the network between hidden layers, Tr.D: Training Data, Tes.D: Testing Data

We Remark that the obtained clustering results of testing data shows that our method gives the good results, because all the testing data were correctly classified except two. In fact; these elements (misclassified) are from the Versicolor class.

TABLE III
CLASSIFICATION FOR DATA SET (PREVIUOS METHOD)

	Nr. T. D	Connect (%)	MC	Accuracy (%)
Tr.D	75	100	3	96
Tes.D	75	100	2	97.3

From Tables above we can see that the proposed method gets a higher average classification accuracy rate and we have used a few connections than the existing methods. And we can conclude that the proposed approach in this paper gives better results compared to the neural methods optimizing neurons hidden layers.

TABLE IV
COMPARISON FOR IRIS DATA CLASSIFICATION

Method	Connec (%)	It.	M.T.	M.TS	A.T (%)	A.TS (%)
EBP	100	500	3	2	96	97.3
EBP	100	800	2	1	97.3	98.6
RBF	100	85	4	4	94.6	94.6
RBF	100	111	4	2	96	97.3
SVM	_	5000	3	5	94.6	93.3
Previous Method	100	100	3	2	96	97.3
Proposed Method	50	647	1	2	98.7	

It: Number of iterations, M.T: Misclassified for training set, M.TS: Misclassified for testing set, A.T: Accuracy for training set, A.TS: Accuracy for testing set.

The results are shown in the Tables IV, we can see that The comparison of the average classification accuracy rate, convergence iterations and number of connections used of the proposed method with other existing neural networks training algorithms: Error Back-Propagation (EBP), Radial Basis Function (RBF) neural networks and Support Vector Machine (SVM) chow that our model present two quality: few connections and higher average classification accuracy rate.

VI. Conclusion

A Model is developed to optimize the architecture of Artificial Neural Networks. The Genetic Algorithm is especially appropriate to obtain the optimal solution of the nonlinear problem. This method is tested to determine the optimal number of hidden layers and connection weights in the Multilayer Perceptron and the most favorable weights matrix after training. We have proposed a new modeling for the multilayer Perceptron architecture optimization problem as a mixed-integer problem with constraints. Depending on the Iris data, the results obtained demonstrates the good generalization of neural networks architectures. In conclusion, the optimal architecture of artificial neural network can play an important role in the classification problem. We can call the proposed approach to solve our model with other metaheuristics and we are going to tray with many other data base for real problems: Diabetes, Thyroid, Cancer….

References

[1] M. Ettaouil and Y. Ghanou, "Neural architectures optimization and Genetic algorithms", Wseas Transactions On Computer, Issue 3, Volume 8, 2009, pp. 526-537.

[2] M.Ettaouil M.Lazaar and Y.Ghanou "Architecture optimization model for the multilayer perceptron and clustering" Journal of Theoretical and Applied Information Technology 10 January 2013. Vol. 47 No.1.

[3] T.B Ludermir "Hybrid Optimization Algorithm for the Definition of MLP Neural Network Architectures and Weights" Proceedings of the Fifth International Conference on Hybrid Intelligent Systems (HIS'05) 0-7695-2457-5/05 20.00 2005 IEEE.

[4] JOSEPH RAJ V. 'Better Learning of Supervised Neural Networks Based on Functional Graph – An Experimental Approach', WSEAS TRANSACTIONS on COMPUTERS, Issue 8, Volume 7, August 2008.

[5] D. Wang, 'Fast Constructive-Coverting Algorithm for neural networks and its implement in classification', *Applied Soft Computing* 8 (2008) 166-173.

[6] D. Wang, N.S. Chaudhari, 'A constructive unsupervised learning algorithm for Boolean neural networks based on multi-level geometrical expansion', *Neurocomputing* 57C (2004) 455-461.

[7] T. Kohonen, 'Self Organizing Maps', *Springer*, 3eme edition, 2001. *Neural Netw*., vol. 17, no. 6, pp. 1452–1459, Nov. 2006.
[8] E. Egriogglu, C, Hakam Aladag, S. Gunay, 'A new model selection straegy in artificiel neural networks', *Applied Mathematics and Computation* (195) 591-597, 2008.
[9] Bishop CM (2005) Neural networks for pattern recognition. MIT Press, Cambridge.
[10] Rosenblatt, "The Perceptron: A Theory of Statistical Separability in Cognitive Systems", Cornell Aeronautical Laboratory, Report No. VG-1196-G-1, January, 1958.
[11] Krzysztof Socha Christian Blum "An ant colony optimization algorithm for continuous optimization: application to feed-forward neural network training "
[12] D.Salamon, "Data compression", Springer, 2004.
[13] S. J. Bolaños, R. G. Crespo, V. H. Medina-García, "Patterns of software development process, International Journal of Interactive Multimedia and Artificial Intelligence, vol. 1, no. 4, pp. 33-40, 2011.
[14] T. B. Ludermir, A. Yamazaki, and C. Zanchettin, "An optimization methodology for neural network weights and architectures," *IEEE Trans.*

Hassan Ramchoun is a PhD student in the Laboratory of Modeling and Scientific computing at the Faculty of Sciences and Technology of Fez, Morocco, he is a member of Operational Research and Computer group. He works on Neural Network, probabilistic modeling, classification problems, and statistical learning methods.

Mohammed Amine Janati Idrissi is a PhD student in Modeling and Scientific Computing laboratory at the Faculty of Sciences and Technology of Fez, Morocco. His research interests include metaheuristics, artificial neural networks, and applications.

Youssef Ghanou is a PhD in Department of Computer Engineering, High School of Technology, Moulay Ismaïl University, B. P. 3103, 50000, Toulal, Meknes, Morocco. His research interests include: Operational Research, metaheuristics, artificial neural network, optimization and applications.

Mohamed Ettaouil is a Doctorate Status in Operational Research and Optimization, FST University Sidi Mohamed Ben Abdellah USMBA, Fez. PhD in Computer Science, University of Paris 13, Galilee Institute, Paris France. He is a professor on the Faculty of Science and technology of Fez FST, and he was responsible of research team in modelization and pattern recognition, operational research and global optimization methods. He was the Director of Unit Formation and Research UFR: Scientific computing and computer science, Engineering Sciences. He is also a responsible of research team in Artificial Neural Networks and Learning, modelization and engineering sciences, FST Fez. He is an expert in the fields of the modelization and optimization, engineering sciences.