New Modeling of Multilayer Perceptron Architecture Optimization with Regularization: An Application to Pattern Classification

H. Ramchoun, M A. Janati Idrissi, Y. Ghanou, and M. Ettaouil

***Abstract*—The Multilayer Perceptron (MLP) is the most useful artificial neural network to estimate the functional structure in the non-linear systems, but the determination of its architecture and weights is a fundamental problem due to their direct impact on the network convergence and performance. This paper presents an extension of our latest work where we have treated the problem of optimizing the number of connections weights and hidden layers in MLP. We introduce an objective function regularized by the sum of connections used by the network and weight decay, to solve the resulting model, we use a genetic algorithm and we train the network with back-propagation method. To test our methodology we use a data classification: wine, iris, seeds and medical data (Cancer, thyroid). We compare the obtained results with those of similar works existing in the literature. The numerical results illustrate the advantages of our approach as a new method of architecture optimization and training.**

***Index Terms*—Multilayer Perceptron, Architecture optimization, Objective function, Genetic algorithm, Weight decay.**

I. Introduction

ARTIFICAL Neural Network (ANN) models are an important class of models that have attracted considerable attention and have been deployed in many applications: pattern recognition, classification, forecasting and optimization. The most utilized model in ANN applications is the Multilayer Perceptron (MLP) which consists of an input layer, output layer and hidden layers [8].

Training the MLP 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]. In the literature there are some algorithms for this task such as Ant Colony Optimization [28], Particle Swarm Optimization [25], Levenberg-Marquardt, but the most utilized technique is the Back-propagation algorithm (BP) and there is another way to train the MLP such as the Bayesian training framework proposed by Neal and Mackay [26- 27].

A manual selection of the ANN parameters and architectures involves difficulties such as the following: the need for a priori knowledge on the problem domain and ANN functioning in order to define these parameters, the exponential number of parameters that need to be adjusted.

The selection 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 hidden layers and neurons.

This paper presents a new model which automatically optimizes the ANN architecture and performance. This method is based on modeling the problem of architecture by means of non-linear constraints program such as in [24] but we want to penalize the cost function by the term defining the sum of network connections used, weighted by the parameter that we want to choose by tests. In this paper, the performance of our approach is investigated in the simultaneous optimization of multilayer perceptron (MLP) architecture and weights.

The proposed model presents some interesting characteristics: 1) Optimal search for useful connections. 2) Optimization of network size and 3) Automatic approach for finding the best network architecture.

The next section presents and discusses related works on neural network architecture optimization. Section 3 describes the training of Multilayer Perceptron. 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 5.

II. Related Works

There are a number of approaches in the literature that take into account the architecture optimization for MLP. This section describes works that are more or less similar to our article.

Manuscript received July 12, 2016; revised December 20, 2016.

H. Ramchoun is PhD student in Laboratory of modeling and scientific computing at the Faculty of Sciences and Technology of Fez, Morocco (corresponding author phone: +212642800501; e-mail: ramchounhassan@gmail.com)

M A. Janati Idrissi is PhD student in Laboratory of modeling and scientific computing at the Faculty of Sciences and Technology of Fez, Morocco (e-mail: medamine.janatidrissi@gmail.com).

Y. Ghanou is with the Department of Computer Engineering, High School of Technology, Moulay Ismaïl University, B. P. 3103, 50000, Toulal, Meknes, Morocco (e-mail: youssefghanou@yahoo.fr).

M. Ettaouil is a Doctorate Status in Operational Research and Optimization, Faculty of Sciences and Technology, Sidi Mohamed Ben Abdellah University, B.P. 2202 Imouzzer route, Fez, Morocco. (e-mail: mohamedettaouil@yahoo.fr).

The performance of MLP depends, on the one hand, on various aspects of design such as the choice of an optimal network topology including the number of hidden layers and nodes for each hidden layer and on the other hand The appropriate learning algorithm and the initialization of the weights [23]. 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 [3], in order to determine neural networks parameters, but not optimally.

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 neurons number of hidden layer. The most of researchers treat the construction of neural architecture without finding the optimal neural architecture [7].

T B. Ludermir and 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 treat optimization of hidden layers with introducing a decision variable for each layer [1], in another work we have taken account the hidden node optimization in layers[2], and recently a Multi objective approach [29], for training this three models we have used a back-propagation algorithms.

III. Multilayer Perceptron and Training

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 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 figure 1.

The choice of layers number and neurons in each layers and connections called architecture problem, 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.

Our main objective is to optimize this architecture for suitable network with sufficient parameters for classification or regression task.

B. *Back-propagation and learning*

The 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 algorithms are used such as Ant colony [11], but the most used called Back-propagation witch based on descent gradient techniques [12].

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 $f(x)$ where:

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

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 in each hidden layer h_i.

For the output of first hidden layer

$$h_1^j = f\left(\sum_{k=1}^{n_0} w_{k,j}^0 \, x_k\right) \qquad j = 1, \dots, n_1 \quad (2)$$

The outputs h_i^j of neurons in the hidden layers are calculated as follows:

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

Where w^i is the weight between the neuron k in the

hidden l s

the num t

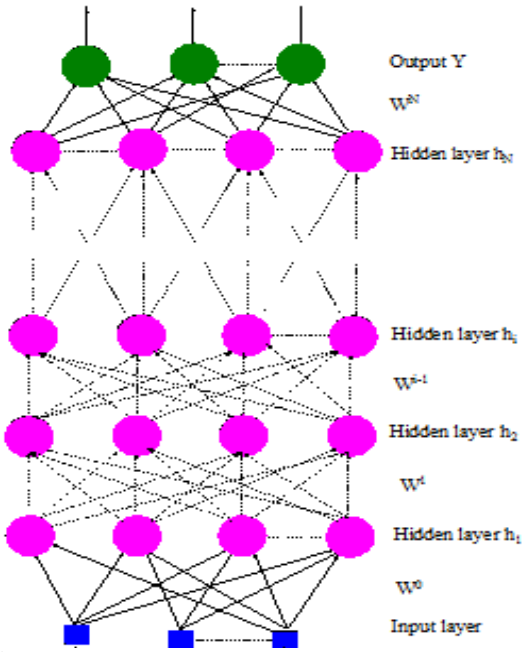

Fig. 1 Feed forward neural network structure

of the i^{th} layers can be formulated by:

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

The network outputs are computed by

$$y_i = f\left(\sum_{k=1}^{n_N} w_{k,j}^N \, h_N^k\right) \qquad (5)$$

$$Y = \left(y_1, \dots, y_j, \dots, y_{N+1}\right) = F(W, X) \qquad (6)$$

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

$$W = [W^0, \dots, W^j, \dots, W^N]$$

$$W^i = \left(\mathrm{w}_{\mathrm{j,k}}^{\mathrm{i}}\right)_{\substack{0 \le i \le N \\ 1 \le j \le n_{i+1} \\ 1 \le k \le n_i}} \text{ where } \mathrm{w}_{\mathrm{j,k}}^{\mathrm{i}} \in \mathbb{R} \qquad (7)$$

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 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 W^N is the matrix of weights between the N^{th} hidden layer and the output layer.

IV. Proposed Model to Optimize the MLP Weight and Architecture

The MLP architecture definition depends on the choice of the number of layers, the number of hidden nodes in each layers and the objective function, but another approach which is introduced in this paper describe a method to remove some unnecessary connection between the hidden layers. We remove the neuron of layer i when there are no connections between this neuron and neurons 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.

A. Notations

N : Number of hidden layers.
n_0 : Number of neurons in input layer.
n_i : Number of neurons in hidden layer *i.*
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 $i = 1, \dots, N-1$. $U = (u_1, u_2, \dots, u_{N-1})$
$v_{k,j}^i$: Binary variable $i = 2, \dots, N,\ j = 1, \dots, n_i$ and $k = 1, \dots, n_{i-1}$

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

$$F(U,V,W,X) = Y = \left(y_1, y_2, \dots, y_{n_{N+1}}\right) \tag{8}$$

B. 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:

$$h_1 = \begin{pmatrix} h_1^1 \\ \vdots \\ h_1^k \\ \vdots \\ h_1^{n_1} \end{pmatrix} = \begin{pmatrix} f\left(\sum_{k=1}^{n_0} w_{k,1}^0\, x_k\right) \\ \vdots \\ f\left(\sum_{k=1}^{n_0} w_{k,j}^0\, x_k\right) \\ \vdots \\ f\left(\sum_{k=1}^{n_0} w_{k,n_1}^0\, x_k\right) \end{pmatrix} \tag{9}$$

C. Output of the hidden layers i=2,…,N

To compute the output of each neuron for the hidden layer *i*, where *i = 2,…, N*, we propose this formula:

$$h_i = \begin{pmatrix} h_i^1 \\ \vdots \\ h_i^k \\ \vdots \\ h_i^{n_i} \end{pmatrix} = (1-u_{i-1})h_{i-1} + u_{i-1} \begin{pmatrix} 1-\prod_{k=1}^{n_{i-1}}\left(1-v_{k,1}^{i-1}\right) \\ \vdots \\ 1-\prod_{k=1}^{n_{i-1}}\left(1-v_{k,j}^{i-1}\right) \\ \vdots \\ 1-\prod_{k=1}^{n_{i-1}}\left(1-v_{k,n_i}^{i-1}\right) \end{pmatrix} \begin{pmatrix} f\left(\sum_{k=1}^{n_{i-1}} v_{k,1}^{i-1}\, w_{k,1}^{i-1} h_{i-1}^k\right) \\ \vdots \\ f\left(\sum_{k=1}^{n_{i-1}} v_{k,j}^{i-1}\, w_{k,j}^{i-1} h_{i-1}^k\right) \\ \vdots \\ f\left(\sum_{k=1}^{n_{i-1}} v_{k,n_i}^{i-1}\, w_{k,n_i}^{i-1} h_{i-1}^k\right) \end{pmatrix} \tag{10}$$

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

D. Output of the neural network (layer N+1)

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

$$Y = \begin{pmatrix} y_1 \\ \vdots \\ y_k \\ \vdots \\ y_{n_{N+1}} \end{pmatrix} = \begin{pmatrix} f\left(\sum_{k=1}^{n_N} w_{k,1}^N h_N^k\right) \\ \vdots \\ f\left(\sum_{k=1}^{n_N} w_{k,j}^N h_N^k\right) \\ \vdots \\ f\left(\sum_{k=1}^{n_N} w_{k,N+1}^N h_N^k\right) \end{pmatrix} \tag{11}$$

E. Objective function

Considering N_C classes in the data set, the true class of the pattern x from the training set A is defined as

$$\gamma(X) \in \{1,2, \dots, N_C\} \tag{12}$$

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

$$\varepsilon(U,V,X,W) = \|F(U,V,X,W) - \gamma(X)\|^2 \tag{13}$$

The classification error for the training set can be define

$$E(U,V,W,X) = \frac{1}{2\ card\ (A)} \sum_{X \in\ A} \varepsilon(U,V,X,W) \tag{14}$$

We propose to add regularization term for error previously defined, in order to control variations of connections weights used in training and optimization phase. The maximal number of connections used by the network is

$$N_{max} = \sum_{i=0}^{N} n_i\, n_{i+1} \tag{15}$$

In our case we can estimate the percentage of connections used in the network by:

$$\psi(v) = \frac{1}{N_{max}} \sum_{i=0}^{N} \sum_{j=0}^{n_i} \sum_{k=0}^{n_{i+1}} v_{j,k}^i \tag{16}$$

In machine learning Regularization is a mechanism of introducing additional information to prevent over-fitting, This information is usually a penalty for complexity to favor models with small coefficients, such as bounds on the vector space norm. To this, different formula can be found in literature, in our case we use the following term:

$$\mathrm{K(w)} = \frac{\beta}{2} \sum_{\mathrm{i=0}}^{\mathrm{N}} \sum_{\mathrm{j=0}}^{\mathrm{n_i}} \sum_{\mathrm{k=0}}^{\mathrm{n_{i+1}}} \frac{\mathrm{w_{j,k}^{i\ 2}}}{1+\mathrm{w_{j,k}^{i\ 2}}} \tag{17}$$

Finally we propose our cost function as follows

$$c(u,v,w,x) = E(u,v,w,x) + \frac{\alpha}{2}\psi(v) + K(w) \tag{18}$$

Where α and β are parameters can be determined by repetition of experiments.

The estimation of regularization parameter β can also find its interpretation automatically as part of the Bayesian approach to neural networks described in [26].

F. Constraints

- The first constraint guarantees the existence of at least one hidden layer.

$$\sum_{i=1}^{N} u_i \geq 1 \quad (19)$$

- These constraints insured communication between neurons, connections and layers

$$u_{i-1} \times \prod_{j=1}^{n_i} \prod_{k=1}^{n_{i-1}} (1 - v_{k,j}^{i-1}) = 0 \qquad \forall i = 2, \dots, N \quad (20)$$

$$(1 - u_{i-1}) \times \sum_{j=1}^{n_i} \sum_{k=1}^{n_{i-1}} v_{k,j}^{i-1} = 0 \qquad \forall i = 2, \dots, N \quad (21)$$

- The weights values are the real number.

The neural architecture optimization problem can be formulated as the following model:

$$(P) \begin{cases} Min\ c(u, v, w, x) \\ Subject\ to: \\ \sum_{i=1}^{N} u_i \geq 1 \\ u_{i-1} \times \prod_{j=1}^{n_i} \prod_{k=1}^{n_{i-1}} (1 - v_{k,j}^{i-1}) = 0 \\ (1 - u_{i-1}) \times \sum_{j=1}^{n_i} \sum_{k=1}^{n_{i-1}} v_{k,j}^{i-1} = 0 \\ W = \left(w_{k,j}^{i}\right)_{\substack{0 \leq i \leq N \\ 1 \leq j \leq n_{i+1} \\ 1 \leq k \leq n_i}} where\ w_{k,j}^{i} \in \mathbb{R} \\ u_i \in \{0,1\} \quad where\ i = 1, \dots, N \\ V = \left(v_{k,j}^{i}\right)_{\substack{1 \leq i \leq N-1 \\ 1 \leq k \leq n_{i-1} \\ 1 \leq j \leq n_i}} where\ v_{k,j}^{i} \in \{0,1\} \end{cases}$$

Many exact methods for solving mixed-integer non-linear programming (MINLPs) include innovative approaches and related techniques taken and extended from Mixed-integer programming (MIP). Outer Approximation (OA) methods [16-17], Branch-and-Bound (B&B) [18-19], Extended Cutting Plane methods [22], and Generalized Bender's Decomposition (GBD) [17] for solving MINLPs have been discussed in the literature since the early 1980's. These approaches generally rely on the successive solutions of closely related NLP problems. For example, B&B starts out forming a pure continuous NLP problem by dropping the integrality requirements of the discrete variables (often called the relaxed MINLP or RMINLP). Moreover, each node of the emerging B&B tree represents a solution of the RMINLP with adjusted bounds on the discrete variables [15].

The disadvantage of the exact solution methods mentioned above is that they become computationally intensive as the number of variables is increased throughout the procedure. Therefore, efficient heuristic methods are required to solve large-size instances accurately.

The heuristics methods for solving combinatorial optimization have now a long history, and there are virtually no well-known, hard optimization problems for which a meta-heuristic has not been applied. Often, meta-heuristics obtain the best known solutions for hard, large-size real problems, for which exact methods are too time consuming to be applied in practice.

In the following section, we propose to use the genetic algorithm because it is one of the most metaheuristics adapted to our optimization problem.

G. Genetic Algorithm's framework

The Genetic Algorithm (GA) was introduced by J. HOLLAND to solve a large number of complex optimization problems [21]. Each solution represents an individual who is coded in one or several chromosomes. These chromosomes represent the problem's variables. First, an initial population composed by a fix number of individuals is generated, then, operators of reproduction are applied to a number of individuals selected switch their fitness. This procedure is repeated until the maximums number of iterations is attained. GA has been applied in a large number of optimization problems in several domains, telecommunication, routing, scheduling, and it proves it's efficiently to obtain a good solution [20]. We have formulated the problem as a non-linear program with mixed variables. We summarize the steps of the genetic algorithm, see figure 2, as follows

(i) Choose the initial population of individuals
(ii) Evaluate the fitness of each individual in the population
(iii) Repeat on this generation
  a. Select the best-fit individuals for reproduction.
  b. Crossover and Mutation operations.
  c. Evaluate the individual fitness of new individuals.
  d. Replace least-fit population with new individuals.

Until termination (time limit, fitness achieved, etc.)

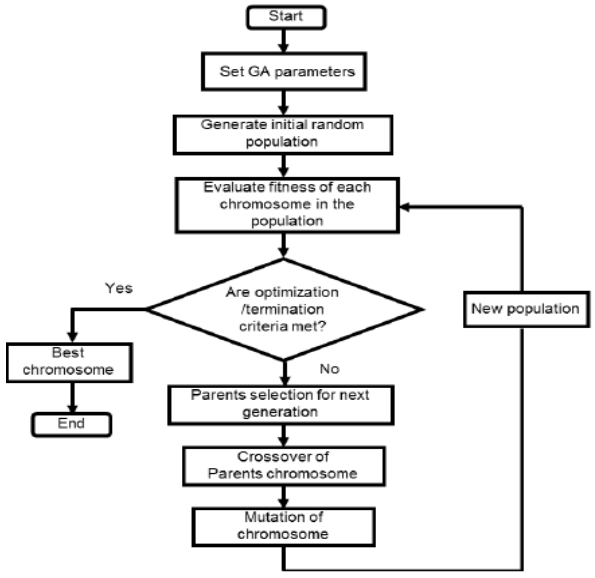

Fig .2. Genetic algorithm flowchart

Coding and initial population: In our application, we have encoded an individual by three chromosomes, the first one represent the matrix of weights W, the second represents the vector U and the third code the vector V which is an array of decision variables who takes 0 or 1.

The first step in the functioning of a GA is, then, the generation of an initial population. Each member of this population encodes a possible solution to the problem.

The individuals of the initial population are randomly generated, u_i , $v_{k,j}^i$ take the value 0 or 1, and the weights matrix takes random values in space $[x_{min}, x_{max}]^p$ where $x_{min} = min\{x_k^i\}$ and $x_{max} = max\{x_k^i\}$ for $k = 1, \dots, p$ and $i = 1, \dots, n$.

Evaluating individuals and selection: After creating the initial population, each individual is evaluated and assigned a fitness value according to the fitness function. In this step, each individual is assigned a numerical value called fitness which corresponds to its performance; it depends essentially on the value of objective function. An individual who has a great fitness is the one who is the most adapted to the problem.

The fitness suggested in our work is the following function:

$$f(individual) = M - c(u, v, w, x) \quad (22)$$

Minimize the value of the objective function is equivalent to maximize the value of the fitness function, were M is a very great number.

The selection method used in this paper is the Roulette Wheel Selection (RWS), which is a proportional method of selection, in this method; individuals are selected according to their fitness see [1] for details.

Crossover: The crossover is very important phase in the GA, at this step, new individuals called children are created by individuals selected from the population called parents. Children are constructed as follows:

We fix a point of crossover, the parents are cut switch this point, the first part of parent 1 and the second of parent 2 go to child 1 and the rest go to child 2, we choose 3 different crossover points, the first for the matrix of weights and the second is for vector U, but the third is for vector V.

Mutation: The rule of mutation is to keep the diversity of solutions in order to avoid local optimums. It corresponds on changing the values of one (or several) value (s) of the individuals who are (or were) (s) chosen randomly.

Stop criteria: The optimization process stops for the following reasons:

(i) The maximum number of iteration is reached.
(ii) The maximum threshold error is achieved.

V. Data Set Experiments

In this section a number of experiments were conducted with standard benchmark data sets of the University of California Irvine (UCI) machine learning repository [13] to test the performance of our methodology. Five classification problems are used: Fisher's iris data set, Seed data set, Breast cancer Wisconsin, Wine data set and Thyroid data.

The table I chow the summary of the used data sets along with the number of examples, number of attributes and class.

TABLE I
CHARACTERISTICS OF USED DATA SET

Database	Examples	Attributes	Class
Iris	150	4	3
Wine	178	13	3
Cancer	699	9	2
Seed	210	7	3
Thyroid	7200	21	3

Breast Cancer Wisconsin: The Wisconsin Breast Cancer Data set, consists of 699 cases, of which 458 are diagnosed as benign and the remaining 241 are known to be malignant. There are no missing attributes in the data set, and in this case, we are interested in classifying the breast tumor as benign and malignant. These two-class problems determine if a patient suffers breast cancer based on several characteristics of the nuclei's cells. We are used the first 349 of the patterns as training set (i.e, for optimizing the NN weights), and the remaining 350 as test set.

Iris data set 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.

The wine data set consists of three target classes: the first class corresponds to 59, the second corresponds to 71 and the last one corresponds to 48 examples, each sample has 13 real-valued features. These data are the results of a chemical analysis of wines growing in the same region in Italy but derived from three different cultivars. The analysis determined the quantities of 13 constituents found in each of the three types of wines. The database includes 178 instances.

Thyroid: This data set contains information related to thyroid dysfunction. The problem is to determine whether a patient has a normally functioning thyroid, an under-functioning thyroid (hypothyroid) or an over-active thyroid (hyperthyroid). There are 7200 cases in the data set, with 21 attributes used to determine to which of the three classes the patient belongs. This dataset was obtained from [13].

VI. Implementations and Numerical Results

To illustrate the advantages of the proposed approach, we tested it on databases described above. For this task, after we have obtained a good architecture by GA, we train the obtained MLP by back-propagation algorithm. In this section we present parameters setting, implementations procedures and final results.

A. Parameters setting

We use the GA to solve the architecture optimization problem. To this end, we have coded individual by one chromosome. The individuals of the initial population are generated randomly, vectors u_i, $v_{k,j}^i$ take the value 0 or 1, and the weights take 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:

$$f(individual) = M - c(individual) \quad (23)$$

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 research space of solution.

TABLE II
INITIALS AND OPTIMIZED ARCHITECTURES

Database	Initials Number of hidden layers	Optimized number of hidden layers	Neurons number in each hidden layer
Iris	4	3	4
Wine	4	2	13
Cancer	4	2	9
Seed	4	2	7
Thyroid	4	2	4

TABLE III
PARAMETERS USED IN OUR EXPERIMENTS

P_m	0.200	0.200	0.300
P_c	0.700	0.800	0.700
s	0.520	0.500	0.550
α	0.010	0.300	0.450
μ	0.395	0.397	0.340
β	0.253	0.342	0.651

P_m: Probability of mutation
P_c: Probability of crossover
α: Parameter for connection error
β: Regularization parameter
μ: Learning rate
s: Threshold

We begin with the architecture of four hidden layers with an appropriate hidden unit number according to data set wanted to classify. To simplify, for each data, we chose the same neurons number for all hidden layers. After the first experiments we choose the architectures and parameters summarized in the tables above.

B. *Results for the MLP network*

In experiments, for each topology, ten runs were performed with 10 distinct random weights initializations, to compute the classification rate, we measure the classification error percentage for training and testing set by following these formulas:

$$Class(\%) = \frac{Mal\ classified\ \times 100}{Card(training\ set)}$$

$$Class(\%) = \frac{Mal\ classified\ \times 100}{Card(Testing\ set)}$$

We use the same topology obtained by the GA above. Table IV presents the classification error of the test set obtained in the training of a fully connected MLP by using a gradient descent with only learning rate. The last table column show the resulting number of NN weights computed by (15).

TABLE IV
RESULTS FOR MLP NEURAL NETWORKS

Data	Class(%)	Nb of Hidden layers	Neurons Nb in each hidden layer	Nb of weights (N_{max})
Iris	4	3	4	60
Wine	3,37	2	13	377
Seed	5,71	2	7	119
Cancer	2,28	2	9	180
Thyroid	3,53	2	4	112

Therefore, it becomes important to optimize the network topologies, particularly the useful connections and the hidden layers number in order to verify if the MLP networks can obtains better performances when the architecture is defined by our optimization methodology.

C. *Optimization methodology results*

After determining the total of optimal number of actives connections and the optimal number of hidden layers, Table II, according to data set used in the experiments we can initialize the neural networks by value of weights obtained with GA and we divided all data sets into two equal training and testing set, We use the back-propagation algorithm for training the network with a learning rate and a maximum number of 10000 iterations and we run our algorithm ten times for each data set, we obtained the results presented in the Table V.

TABLE V
DATA SETS CLASSIFICATION (PROPOSED METHOD)

Data set	Class/Connec(%)	Tr.D	Tes.D
Iris	Class(%)	1,33	2,66
	Connec(%)	50	50
Wine	Class(%)	1,12	2,25
	Connec(%)	77.45	77,45
Seed	Class(%)	0	5,71
	Connec(%)	77,31	77,31
Cancer	Class(%)	3,71	2
	Connec(%)	78,34	78,34
Thyroid	Class(%)	1.92	3
	Connec(%)	92,85	92,85

Connect (%): percentage of connections weights used in the network between layers
Tr.D: Training Data
Tes.D: Testing Data

We note that, the obtained clustering results for the test bases show that our method provides us the existence of the unnecessary connections.

From Tables above we can see that the proposed method gets a higher average classification accuracy rate where we have used a few connections rather than the MLP alone. Just for seed database we have obtained the same percentage of misclassified data where we use the percentage of 77.31 %. In the next section we will compare our obtained results with other existing methods in the literature.

The performance analysis of the testing set by MLP, and proposed method are also represented graphically in Fig. 3. As noted from Tables IV and V, the proposed method gives the best accuracy in testing phase with a percentage of used connections lower than 100 %.

The Figure 4 illustrates the evolution of the mean square error (MSE) as a function of iterations during the training phase, we note that for each database the error decreases, then stabilizes from a certain number of iterations which shows the performance of our method.

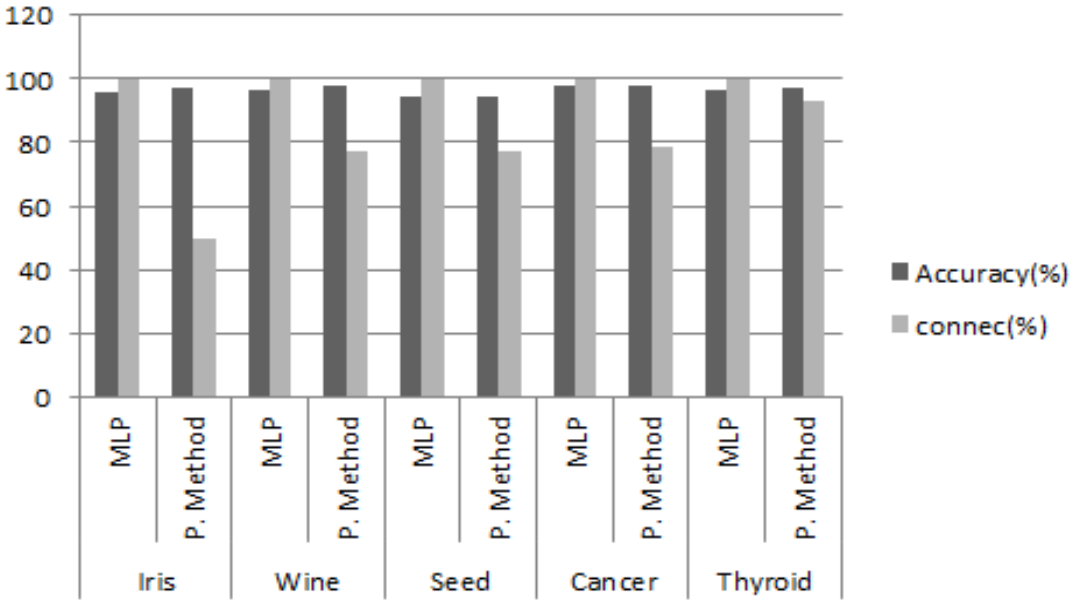

Fig. 3 Performance comparison of MLP Network and Proposed method during testing phase.

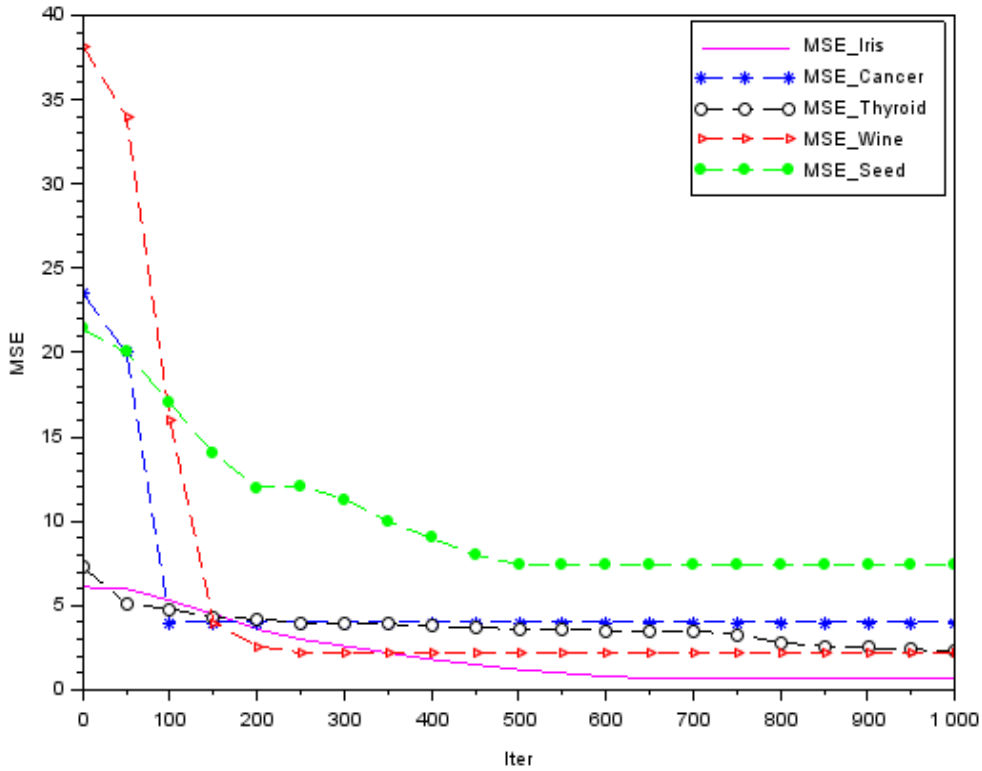

Fig. 4 MSE evolution during training phase for 1000 iterations

D. Results comparison

The problem of architecture optimization and training exists in the literature, but with different proposed models and different training methodology. We describe some of them and we compare it with our proposed method

- ACO+BP: This work demonstrates the hybrid training method: Optimization of MLP architecture and Training using Ant Colony Algorithm [11].
- The GATSA method was used by T.B. Ludermir and al for one hidden layer MLP neural network [3-14].
- Our previous work where we deal with optimizing neurons number and hidden layers in the network [1-2]
- BP: This method allows training a fully connected MLP with Back-propagation algorithm

The Table VI displays the average performance of each investigated optimization technique. These results were obtained for each technique by the optimization of the number of connections and weights connections values of an MLP. The parameters evaluated were the following: the classification error (class) of the test sets and the percentage of network connections. For all data sets, the optimized ANN obtains a lower classification error than those obtained by MLP without architecture optimization (Table V), and the mean number of connections is lower than the maximum number allowed.

TABLE VI
RESULTS COMPARISON

		GATSA +BP	ACO +BP	BP	Prev.M	Prop.M
Iris	Class	5,25	2,66	4	2,66	2,66
	Connec	68,15	100	100	100	**50**
Wine	Class	-	-	3,37	-	**2,25**
	Connec	-	-	100	-	**77,45**
Cancer	Class	7,19	6	2,28	-	**2**
	Connec	72,67	100	100	-	**78,34**
Seed	Class	-	7,27	5,71	-	5,71
	Connec	-	100	100	-	**77,31**
Thyroid	Class	7,15	-	3,53	-	**3**
	Connec	87.36	-	100	-	92,85

In most of the simulations, the best performance to optimize the ANN architecture was obtained by the proposed method.

We notice that the Iris data set as the classification error was around 2.66 is better than the previous method which carries into consideration the optimization of neurons and layers, effectively with a number of connections lower, we were able to exceed the method proposed by T.B. Ludermir in term of classification rate.

For the Cancer data set, the best classification error was of 2 (2.28 in a full connected MLP with back-propagation training) and 7.19 with GATSA+BP method with a lower number of connections, on the other hand the ACO+BP method was of 2.14 but in a full connected MLP, however we can conclude that our proposed method for this data set is the good one. In the Seed and Wine data set we can compare our proposed method to the full connected MLP with back-propagation and we conclude the effectiveness of our approach.

TABLE VII
RESULTS 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	97.3

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 of the Table VII chow that, For Iris database, 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) show that our approach present three qualities: few connections, higher average classification accuracy rate and a mean number of iterations.

VII. CONCLUSION

We presented an automatically model to optimize the architecture of Multilayer Perceptron. This paper has chown that the deletion of some unnecessary connections and layers can be successfully used for the optimization of the MLP network topology and weights. The Genetic Algorithm is especially appropriate to obtain the optimal solution of the non-linear model. This method has been tested to determine the optimal number of hidden layers, actives connections weights and the most favorable weights matrix after training. Depending on the data sets: Iris, Cancer, Thyroid, Wine and Seed, The obtained results demonstrate the good generalization of Multilayer Perceptron architectures. To conclude, the optimal architecture of artificial neural network, especially when we optimize the connections number with regularization, can play an important role for the classification

rate and the problem complexity. For our future work we can use other meta-heuristics such as ACO, PSO to solve our model and methodology of experiences plans to determine efficiently some parameters.

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 and al, 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] Y. Ghanou, G. Bencheikh, "Architecture Optimization and Training for the Multilayer Perceptron using Ant System," IAENG International Journal of Computer Science, vol. 43, no.1, pp 20-26, 2016

[12] D. Salamon, Data compression. Springer, 2004.

[13] UC Irvine Machine Learning Repository, 333 data sets for machine learning are available at www.ics.uci.edu/mlearn/MLRepository.html

[14] T. B. Ludermir, A. Yamazaki, and C. Zanchettin, An optimization methodology for neural network weights and architectures. *IEEE Tranas.*

[15] J. F. Benders. (1962) Partitioning procedures for solving mixed-variables programming problems. Numer. Math., 4:238.

[16] E. Egriogglu, C, Hakam Aladag, S. Gunay,(2008) A new model selection straegy in artificiel neural networks. Applied Mathematics and Computation (195), pp. 591-597.

[17] R.Fletcher and S. Leyffer. (1994) Solving Mixed Integer Programs by Outer Approximation. Math. Program. 66, 327–349.

[18] O.K. Gupta and A. Ravindran. (1985) Branch and Bound Experiments in Convex Nonlinear Integer Programming. Manage Sci., 31 (12), pp. 1533–1546.

[19] Quesada and I.E. Grossmann. (1992) An LP/NLP Based Branch and Bound Algorithm for Convex MINLP Optimization Problems. Computers Chem. Eng., 16 (10/11), pp. 937–947.

[20] K. Deep, K. Pratap Singh, M.L. Kansal, C. Mohan, (2009) A real coded genetic algorithm for solving integer and mixed integer optimization problems. Applied Mathematics and Computation, pp. 505–518.

[21] J. Holland, (1992) Genetic Algorithms, pour la science, n°179, Edition of Scientific American, pp. 44-50.

[22] T.Westerlund and F. Petersson. (1995) A Cutting Plane Method for Solving Convex MINLP Problems. Computers Chem. Eng., 19, pp. 131–136.

[23] F. Ahmad, N. A. Mat Isa, Z. Hussain, M. K. Osman, S. N. Sulaiman (2014) A GA-based feature selection and parameter optimization of an ANN in diagnosing breast cancer. Pattern Anal Applic DOI 10.1007/s10044-014-0375-9.

[24] H. Ramchoun, M A. Janati Idrissi, Y. Ghanou, M. Ettaouil ''Multilayer perceptron: Architecture optimization and training'' International Journal of Artificial Intelligence and Interactive Multimidia Vol.4 no 1 pp. 26-30, 2016.

[25] M. Carvalho and T.B. Ludermir Hybrid Training of Feed-Forward Neural Networks with Particle Swarm Optimization in International Conference on Neural Information Processing (ICONIP2006), Part II, LNCS 4233, pp. 1061-1070, 2006.

[26] Mackay, D. J. C. (1992). A practical Bayesian framework for back-propagation networks. Neural Computation, 4 (3), 448-472

[27] Neal, R. M. (1996). Bayesian learning for neural networks, volume 118 of lecture notes in statistics. Springer-Verlag.

[28] Krzysztof Socha, Christian Blum. An ant colony optimization algorithm for continuous optimization: application to feed-forward neural network training neural computation and application 16 235-247 2007

[29] M A. Janati Idrissi, H. Ramchoun, Y. Ghanou, M. Ettaouil. ''Genetic algorithm for neural network architecture optimization'' IEEE Proceeding of The 3rd International Conference of Logistics Operations Management 2016, GOL 2016, 23-25 May 2016, Morocco.