Architecture Optimization and Training for the Multilayer Perceptron using Ant System

Y. Ghanou, and G. Bencheikh

***Abstract*— we present in this paper an Ant Colony Algorithm to optimize the performance of the multilayer Perceptron. Indeed, the performance of the multilayer Perception depends on its parameters such as the number of neurons in the hidden layer and the connection weights. In this respect, we firstly model the problem of neural architecture and training in terms of a mixed-integer problem with a linear constraint, and secondly, we propose an Ant Colony Algorithm to solve it. The experimental results illustrate the advantage of our approach as a new method of training and architecture optimization.**

***Index Terms*—Multilayer Perceptron (MLP), Neural Architectures Optimization, Non-linear optimization, Ant Colony Algorithms, Supervised Training, Classification.**

I. Introduction

ONE of the most important problems that neural network designers face today is choosing an appropriate neural architecture for a given application. In this way, we cite constructive methods [18], [7], [3] and pruning methods [28], [10].

The architecture optimization and training of neural networks is a complex task of great importance in problems of supervised learning. Optimization of neural architectures and training method has an influence on the quality of networks measured by their performance [29].

Finding a solution of the hybrid training and the architecture optimization requires a solution a mixed-integer nonlinear optimization problem with a linear constraint [20]. Such problems are NP-complete, so different metaheuristics are applied in the literature to solve it. Genetic algorithms are frequently combined with neural methods to select best architectures and avoid the drawbacks of local minimization methods [8], [29]. Other methods of neural architecture optimization are proposed, such as particle swarm optimization [27].

This paper suggests finding both optimal neural network architecture and the connection weights. In this respect, we optimize the number of neurons in the hidden layer by formulating the problem of neural architecture into a mixed-integer nonlinear problem with linear constraint. We apply an Ant colony algorithm to solve the obtained problem which provides optimal architecture and generalization performance.

Manuscript received March 05, 2015; revised September 29, 2015.

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

G. Bencheikh is with Department of Economics, Faculty of Law, Economics and Socials, Moulay Ismaïl University, B. P. 3103, 50000, Toulal, Meknes, Morocco (e-mail: ghizlane_bencheikh@yahoo.fr).

II. Multilayer Perceptron Architecture

Rosenblatt's work created much excitement, controversy, and interest in neural net models for pattern classification in that period and led to important models abstracted from his work in later years. Currently the names (single-layer) Perceptron and Multilayer Perceptron are used to refer to specific artificial neural network structures based on Rosenblatt's Perceptrons.

ANN is composed of a series of neurons, which have multiple connections with other neurons. Each connection is associated a weight which can be varied in strength, in analogy with neurobiology synapses. A typical ANN architecture is known as the multilayer perceptron (MLP) where the neural network operates is relatively simple. The input layer receives the input vector, each neuron in the input layer receives a value (a component of the input vector), and produces a new value (output) which it sends to each neuron of the next layer [21].

A multilayer Perceptron (MLP) is a variant of the Artificial Neural Network. 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 [22].

The architecture of an Artificial Neural Network is a layout of neurons grouped into layers. The main parameters of ANN are the number of layers, the number of neurons per layer, connectivity level and type of neuron interconnections [9].

The first layer of the neural network is the input layer, we assume that it contains *n* neurons, the last layer of the network is the output layer, and we assume that it contains *m* neurons.

In the Perceptron model, a single neuron with a linear weighted net function and a threshold activation function is employed. The input to this neuron $x = (x_1, x_2, \ldots, x_n)$ is a feature vector in a n-dimensional feature space. The net function *f* is the weighted sum of the inputs:

$$f(x) = w_0 + \sum_{i=1}^{n} w_i . x_i \qquad (1)$$

Input Layer: A vector of variable values $(x_1, x_2, \ldots, x_n)$ is presented to the input layer. The input layer distributes the values to each of the neurons in the first hidden layer.

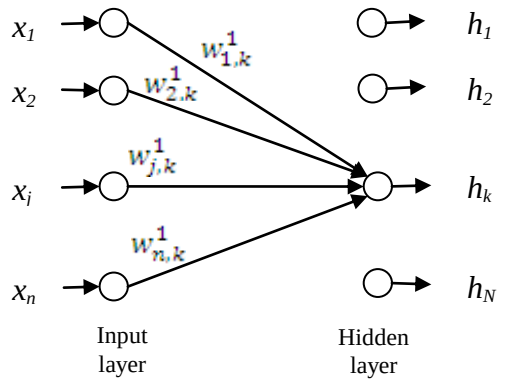

Fig. 1. Outputs for the first hidden layer

Hidden Layer: A group of neurons between the input layer and the output layer. The outputs from the hidden layer are distributed to the output layer. The neurons of first hidden layer are directly connected to the input layer (data layer) of the neural network; Figure (Fig. 1) shows the connections between the first hidden layer and the inputs of the network.

To calculate the outputs for the first hidden layer, we use the following:

$$H = \begin{pmatrix} h_1 \\ \vdots \\ h_k \\ \vdots \\ h_{N_{\max}} \end{pmatrix} + \begin{pmatrix} f\left(\sum_{j=1}^{n} w^1_{j,1} x_j\right) \\ \vdots \\ f\left(\sum_{j=1}^{n} w^1_{j,k} x_j\right) \\ \vdots \\ f\left(\sum_{j=1}^{n} w^1_{j,N_{\max}} x_j\right) \end{pmatrix} \quad (2)$$

The figure (Fig. 2) shows the connections between the input layer and the hidden layer of the neural network.

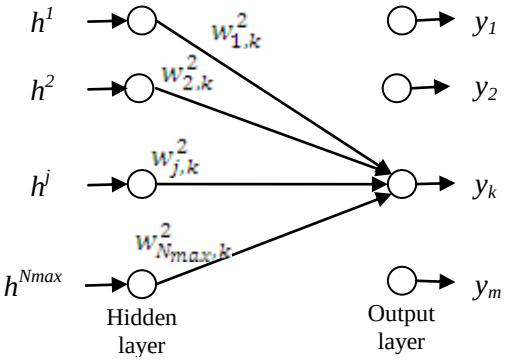

Fig. 2. Outputs for the neural network

Output Layer: Arriving at a neuron in the output layer. The connections between the last hidden layer and Output layer of the neural network are shown in Figure 2.

The outputs *Y* of the network are calculated by this following:

$$Y = \begin{pmatrix} y_1 \\ \vdots \\ y_k \\ \vdots \\ y_m \end{pmatrix} + \begin{pmatrix} f\left(\sum_{j=1}^{N_{\max}} u_j w^2_{j,1} h_j\right) \\ \vdots \\ f\left(\sum_{j=1}^{N_{\max}} u_j w^2_{j,k} h_j\right) \\ \vdots \\ f\left(\sum_{j=1}^{N_{\max}} u_j w^2_{j,m} h_j\right) \end{pmatrix} \quad (3)$$

Neural networks with three layers of neurons with linear outputs and a sufficient number of hidden neurons can approximate with arbitrary precision any Borel-measurable function of a finite dimensional space to another [15], [16]. This property justifies that the number of hidden neurons should be chosen appropriately to obtain the desired precision.

III. Model of Training and Neural Architecture Optimization

The good performance of an ANN is obtained by minimizing the error (distance) between the desired output and the calculated output. This is influenced by the parameters of the ANN.

The training was done in a supervised learning that is for every input vector, the desired output is given and weights are adapted to minimize an error function that measures the discrepancy between the desired output and the output computed by the network. The mean squared error (MSE) is employed as the error function. In this article, a new optimization model is introduced, in order:

-- To optimize the artificial neural architecture,

-- To adjust the parameters of ANN.

In this new proposed strategy, we can formulate the problem of neural architecture optimization as nonlinear constraint programming with mixed variables. We apply the Ant System algorithm to solve it.

For modeling the problem of neural architecture optimization, we define the following parameters:

A. Notations

$N_{\max}$: The maximum number of hidden neurons.
n : The number of neurons by in input.
m : The number of neurons by in output.
$nopt$: The optimal number of hidden neurons.
X : The input of ANN
h : The output of the hidden layer
Y : The calculated (actual) output of ANN
d : The desired output of ANN
f : The activation function of the all neurons.
F : The transfer function of artificial neural network

$$u_i = \begin{cases} 1 & if\ neuron\ i\ is\ used \\ 0 & otherwise \end{cases}$$

We note $U = (u_1, u_2, \ldots, u_N)^T$ and $nopt = \sum_{i=1}^{N_{\max}} u_i$

$W^1 = \left(w_{i,j}^1\right)_{\substack{1\le i\le n \\ 1\le j\le N_{\max}}}$ represents the matrix of weight between neurons of the input layer and those of the hidden layer.

$W^2 = \left(w_{i,j}^2\right)_{\substack{1\le i\le N_{\max} \\ 1\le j\le m}}$ represents the matrix of weight between neurons of the hidden layer and those of the output layer.

We note $W = \left[W^1, W^2\right]$.

B. Cost function

The objective function of the mathematical programming model is the distances between the calculated output vector and the desired output:

$$\left\|F(U, X, W) - d\right\|^2 \qquad (4)$$

Where $F(U,X,W) = Y$

C. Constraint

To guarantee the existence of one hidden neuron at least, we propose the following constraint

$$\sum_{i=1}^{N_{\max}} u_i \ge 1 \qquad (5)$$

D. Optimization Model

We can eliminate some neurons from the perceptron without loss the quality of this network, we can use the variables (u_i) and we introduce the following optimization model (P):

$$(P)\ \begin{cases} \min\left\|F(U, X, W) - d\right\| \\ S.C.: \\ \sum_{i=1}^{N_{\max}} u_i \ge 1 \\ w_{i,j}^1 \in IR \\ w_{i,j}^2 \in IR \end{cases}$$

Our approach begins with a maximal number of hidden neurons. The training process, all observations are sequentially input to learning system in an arbitrary order. The number of hidden layers and the values of weights are determined by the optimization model. In addition, the number of hidden neurons must be decided simultaneously with a training phase (weights adjust).

The literature indicates that basically two approaches are employed in solving the mixed-integer problem, namely, exact methods and heuristic methods. The present work falls in the second group, because the exact procedures need extremely high iterative computing for solving [1], [6], [13].

IV. Ant Colony Optimization Applied To MLP Architecture Optimization And Training

The ant colony optimization (ACO) is a metaheuristic approach introduced by Marco Dorigo in 1992 to solve combinatorial optimization problems [4]. The ACO consists of a colony of artificial ants, which construct iteratively solutions to a given instance of a combinatorial optimization problem and use pheromone trails to communicate [5]. Each ant of the colony represents a solution of the problem as a path and puts a quantity of pheromone on its path according to the solution quality. However, to avoid the reinforcement of paths with bad quality, a step of evaporation is applied at the end of each iteration of the algorithm.

A. Representation of an ant

Each ant of the colony represents a solution of the problem; it is characterized by :

-- A binary vector *U* indicating if neurons are activated or not

--Two matrixes: W^1 and W^2 representing respectively the weighting values between the input layer and the hidden layer and between the hidden layer and the output layer.

--The cost of the solution constructed by the ant. Noted C.

The vector *U* and the matrixes W^1 and W^2 are respectively initialized randomly in {0, 1}, [-1, 1] and [-1, 1], while *C* is calculated according to (4).

$$C = \left\|F(U, X, W) - d\right\|^2 \qquad (4)$$

We note that, *F(U, X, W)* is the output of the MLP noted *Y*, so to determine the value of *C*, we have to calculate the outputs of the hidden layer (*H*) and the output layer (*Y*). These vectors are respectively obtained by expressions (2) and (3).

B. Construction of a solution

The ant begins its solution by the generation of the weighting values of W^1 and W^2 and the binary vector *U*, and then calculates the value of *C*. From iteration to another the ant tries to improve the obtained results by changing some values of W^1, W^2 and *U* in order to minimize the cost function.

To illustrate the process of the modification of W^1, W^2 and U, we consider the graph presented in fig 3.

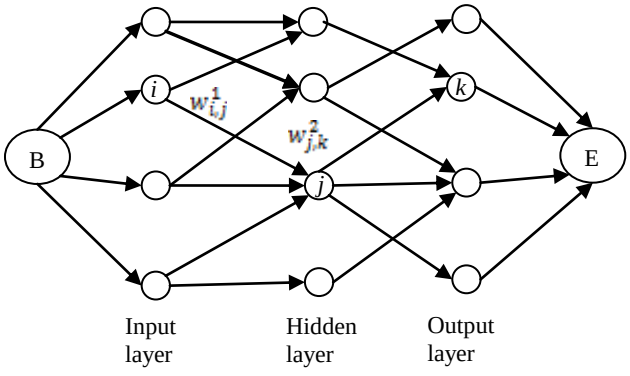

Fig. 3 Graphical representation of the Ant Colony Algorithm

This graph is defined by three levels of nodes, corresponding to the input, hidden and output layers. Nodes of each level are the neurons of the corresponding layer. An arc (i, j) between input and hidden layer is weighted by $w_{i,j}^1$ and an arc (j, k) between hidden and output layer is weighted by $w_{j,k}^2$. Nodes B and E represent the beginning and the

end of the graph. The arcs linked to these two nodes are weighted by 0.

In the first step, the ant has to select the arc between input and hidden layer which is going to be modified. In this step, we are going to change the value of W^1.

This selection is made according to the following probability rule:

$$p_{rj}^{k}(t)=\begin{cases}\dfrac{(\tau_{rj})^{\alpha}.(\eta_{rj})^{\beta}}{\sum_{l}(\tau_{rl})^{\alpha}.(\eta_{rl})^{\beta}} & if\ \ j\in Candidate_{k}\\ 0 & otherwise\end{cases} \qquad (7)$$

Where α and β define the relative importance of τ_{ij} representing the pheromone trail present on arc (i, j) and η_{ij} represents the visibility of the ant.

The parameter η_{ij} depends on the problem characteristics. We defined it by:

$$\frac{1}{\left\|F(X,U,W)-d\right\|^{2}+1} \qquad (8)$$

(we add the term +1 is to avoid the 0 case)

In the second step of the program, the ant has to select the next arc to modify, this time; the ant is going to modify both the weight value W^2 and the binary vector U. We use the same probability rule (9) presented below.

The pheromone trail is updated at the end of each iteration according to the following equation:

$$\tau_{ij}(t+1)=\rho\tau_{ij}(t)+\Delta\tau_{ij}(t) \qquad (9)$$

Where

- ρ is a coefficient of evaporation (ρ < 1 to avoid an unlimited accumulation of trace)
- Δτij is the quantity of trace left on the edge (i, j) by the colony at the end of an iteration. It is defined by:

$$\Delta\tau_{ij}=\begin{cases}\dfrac{Q}{C} & if\ \ (i,j)\in Best\ solution\\ 0 & otherwise\end{cases} \qquad (10)$$

Q : Updating constant

C: Cost function of the best solution in iteration t

Best solution: The minimal cost function found.

C. Ant colony algorithm applied to the problem

Our approach can be summarized as follows:

1. Initialization of the trail of pheromone by an initial value τ_0
2. For each ant
   Initialization of W^1, W^2, U
3. For a maximum number of iterations
   For each ant
   a. Selection of an arc (i, j) between input and hidden layers according to (7)
   b. Modification of $w_{i,j}^{1}$ in [-1, 1]
   c. Selection of an arc (j, k) between hidden and output layers according to (7)
   d. Modification of $w_{j,k}^{2}$ in [-1, 1] and U_j in {0,1}.
   e. Update of the pheromone trail according to (9).

V. Experimental Results of Training And Architecture Optimization To Solve The Classification Problem

To illustrate the performance of the proposed Ant Colony Algorithm, we apply it to a widely used data bases, namely, "Iris", "seeds" and "Cancer" Databases available on the UCI repository of machine learning databases [30].

Table below shows a summary of the databases used, including the number of samples, number of attributes and number of classes.

TABLE I
DATABASES CHARACTERISTICS

Data base	Samples	Attributs	Classes
Iris	150	4	3
Cancer	699	9	2
Seeds	210	7	3

To solve the proposed optimization model, we apply an Ant Colony System [4], [5] in order to determine the optimal number of hidden neurons (Architecture Optimization), and to obtain the matrix of weights (Training).

TABLE II presents for each Database used in our paper the initial Architecture and the optimal Architecture found by our method while maximizing the performance of the ANN. For example for a Peceptron of 40 hidden neurons, the optimal number of neurons is 34 neurons; which means that the number of neurons was reduced by 15%.

TABLE II
RESULTS OF ARCHITECTURE OPTIMIZATION

Data bases	Initial Architecture	Final Architecture
Iris	N_{max} = 20	*nopt* = 19
Seeds	N_{max} = 40	*nopt* = 34
Cancer	N_{max} = 60	*nopt* = 44

A. Preprocessing

Without loss of generality, we simplify the study by translating the Data from their natural to a normalized form using (11). After normalization, all features of these instances are between zero and one.

$$x_{i\ new}^{k}=\frac{x_{i\ old}^{k}-\min(x_i)}{\max(x_i)-\min(x_i)} \qquad (11)$$

The proposed Ant Colony Algorithm was implanted in C++, and tested on three databases. The program was run in a Intel® Core ™ i3 CPU M380 @ 2,53 GHz 2,53 GHz and 3 Go of RAM.

B. Database Iris

This database contains 3 classes of 50 samples, where each class refers to a type of iris plant. The samples in this database are characterized by 4 variables. We used half of the samples for training, and the rest for the test.

In the experiments, with our method, the average of the number of hidden neurons was computed relative to the maximum network architecture generated N_{max}.

In order to illustrate the effect of the Ant Colony Algorithm to solve the model of MLP Architecture optimization and training, the experimental results with 2000 iterations are presented in Table III and Table IV.

TABLE III
NUMERICAL RESULTS WITH 2000 ITERATIONS OBTAINED BY OUR APPROACH APPLIED TO TRAINING DATABASE

	Num. of Training data	Correctly Classified	Misclassified	Accuracy (%)	Num of hidden neurons
Setosa	25	25	0	100	19
Virginica	25	23	2	92	
Versicolor	25	25	0	100	
Total	75	73	2	97,3	

Table III presents the obtained classification results of training data. The proposed method permits to classify all the training data except for two elements which represents 97.3% of the total number of samples.

TABLE IV
NUMERICAL RESULTANTS OBTAINED BY OUR APPROACH APPLIED TO TEST DATABASE

	Num. of testing data	Correctly Classified	Misclassified	Accuracy (%)
Setosa	25	25	0	100
Virginica	25	23	2	96
Versicolor	25	25	0	100
Total	75	73	2	97,3

Table IV presents the obtained classification results of test data. The proposed method permits to classify all the test data except for two elements.

In order to give a better view of the performance, the MSE error of identification examples using our method is shown in Fig. 4, where the MSE error decreases in time.

The following Figure (Fig. 4) shows the evolution the mean square error (MSE) during training phase for 2000 iterations.

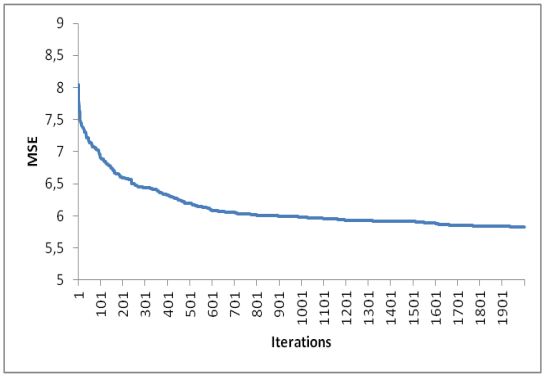

Fig. 4 MSE evolution during learning phase for 2000 iterations

A comparison of the average classification accuracy rate 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). We use half of the data examples (75 items) for training and the remaining (75 items) for testing as well.

TABLE V
NUMERICAL RESULTANTS OBTAINED BY COMPARISON OF THE PROPOSED METHOD WITH OTHER EXISTING NEURAL NETWORKS TRAINING ALGORITHMS

Methods	Num. Of Iterations	Misclassified for training set	Misclassified for testing set	Accuracy for training set (%)	Accuracy for testing set (%)	Num. of hidden Neurons
EBP	800	2	1	97.3	98.6	5
EBP	500	3	2	96	97.3	5
RBF	85	4	4	94.6	94.6	17
RBF	111	3	2	96	97.3	15
SVM	5000	3	5	96	93.3	-
Proposed Method	2000	2	2	97.3	97.3	19

From the results of the experiment with the classification problem, we notice that the performance of the Proposed Method is better than that of RBF and SVM. But the difference between the Proposed Method and the EBF Method is small.

These result shows that our method provides good solutions in terms of the misclassified elements. The advantage of our method happens to be better generalization performance and to propose automatically, for a given problem, the adequate neural architecture.

Our computational results suggest that the proposed method is capable of training neural networks, and of MLP architecture optimization.

C. Database Seeds

In order to illustrate the effect of using Ant Colony Algorithm to solve the model of MLP Architecture optimization and training, we continue our experiments using seeds database.

The seeds database represents measurements of geometrical properties of kernels belonging to three different varieties of wheat. A soft X-ray technique and GRAINS package ware used to construct all seven, real-valued attributes. The examined group comprised kernels belonging to three different varieties of wheat: Kama, Rosa and Canadian, 70 elements each, randomly selected for the experiment. The samples in this database are characterized with 7 variables.

For testing our method, the data set is divided into two parts: the training set and test set. We used 45 samples for training and the rest samples for the test.

TABLE VI
NUMERICAL RESULTANTS OBTAINED BY OUR APPROACH APPLIED TO TRAINING DATABASE "SEEDS"

	Num. of Training data	Correctly Classified	Misclassified	Accuracy (%)	Num of hidden neurons
Kama	15	13	2	86.67	34
Rosa	15	15	0	100	
Canadian	15	15	0	100	
Total	45	43	2	95.57	

Table VI presents the obtained classification results of test data. The proposed method permits to classify all the test data except for two elements.

TABLE VII
NUMERICAL RESULTS OBTAINED BY OUR APPROACH APPLIED TO TEST DATABASE "SEEDS"

	Num. of testing data	Correctly Classified	Misclassified	Accuracy (%)
Kama	55	51	4	92.73
Rosa	55	51	4	92.73
Canadian	55	51	4	92.73
Total	165	153	12	92.73

Table VII presents the obtained classification results of test data. The proposed method permits to classify all the test data except for 12 elements from 165 samples (which represents 7.2% from the test data).

In order to evaluate the performance of the proposed method, we compared our method with other learning algorithms: Support Vector Machine (SVM) and Self Organizing Map (SOM) for the classification of Seeds data. The results are presented in TABLE VIII.

TABLE VIII
NUMERICAL RESULTS OBTAINED BY COMPARISON OF THE PROPOSED METHOD WITH OTHER EXISTING NEURAL NETWORKS TRAINING ALGORITHMS FOR "SEEDS"

Methods	Num. Of Iterations	Accuracy for training set (%)	Accuracy for testing set (%)	Num. of hidden Neurons
SVM	5000	93.33	93.3	-
SOM	2000	91.11	92.12	-
Proposed Method	3000	95.55	92.73	34

The results are shown in the TABLE VIII; we can see that the proposed method gets a higher average classification accuracy rate than SVM and SOM methods.

D. Database Cancer

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.

TABLE IX
NUMERICAL RESULTANTS OBTAINED BY OUR APPROACH APPLIED TO TRAINING DATABASE "CANCER"

	Num. of Training data	Correctly Classified	Misclassified	Accuracy (%)	Num of hidden neurons
Benign	229	210	19	91.7	44
Malignant	121	116	5	95.86	
Total	350	326	24	93.14	

TABLE X
NUMERICAL RESULTANTS OBTAINED BY OUR APPROACH APPLIED TO TEST DATABASE "CANCER"

	Num. of Training data	Correctly Classified	Misclassified	Accuracy (%)
Benign	229	213	16	93.01
Malignant	121	116	5	95.86
Total	350	329	21	94

Based on these tables, we can conclude that the proposed approach in this paper gives better results compared to other methods for training data; but at the same time, we note that the accuracy rate of the average classification of the proposed method is almost the same as the other methods for testing data.

VI. CONCLUSION

This work demonstrates the hybrid training method: Optimization of MLP architecture and Training using Ant Colony Algorithm, where we have formulated this problem in term of mixed-integer problem with a linear constraint (P). In this way, solving the new model (P) using Ant Colony Algorithm was proposed.

Ant System is used to find an optimum solution, each solution represents both the architecture and the weights of an MLP network. This methodology searches for the global minimum, which represents an MLP network with low complexity and good generalization performance.

Near perfect results were attained when using our method to optimize the neural networks architecture.

The results of two experiments demonstrate the successful implementation of our method using Ant Colony Algorithm.

In future work, we plan to improve this Meta heuristic to find an optimal architecture and better quality training for MLP.

REFERENCES

[1] J. F. Benders, "Partitioning procedures for solving mixed-variables programming problems", Numer. Math., 4 – 238, 1962.

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

[3] Derong, L., Tsu-Shuan, C., Z. Yi, "A Constructive Algorithm For Feedforward Neural Networks With Incremental Training", IEEE Transactions on circuits and systems-I: fundamental theory and applications 49 (12), 2002.

[4] M. Dorigo, "Optimization, Learning and Natural Algorithms", PhD thesis, Dip. Elettronica e Informazione, Politecnico di Milano, Italy. 1992.

[5] M. Dorigo and L. M. Gambardella, "Ant colonies for the traveling salesman problem", BioSystems 43, 73-81, 1997.

[6] M. Duran, I. E. Grossmann, "An outer-approximation algorithm for a class of mixed-integer non linear programs", Mathematical Programming, 307-339, 1986.

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

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

[9] M. J. Er, L. Y. Zhai, X. Li and L. San, "A Hybrid Online Sequential Extreme Learning Machine with Simplified Hidden Network," *IAENG International Journal of Computer Science*, vol. 39, no. 1, pp. 1–9, February 2012.

[10] M. Ettaouil, Y. Ghanou, K. Elmoutaouakil, M. Lazaar, "A New Architecture.Optimization Model for the Kohonen Networks and Clustering", Journal of Advanced Research in Computer Science (JARCS), 3 (1), 14 – 32, 2011.

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

[12] J. A. Freeman and D. M. Skapura, "Neural Networks Algorithms, Applications and Programming Techniques", *Pearson Education*, 213 – 262, 2004.

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

[14] Hasham Shiraz Ali, Umar Nauman, Faraz Ahsan, Sajjad Mohsin, "Genetic Algorithm Based Bowling Coach For Cricket", *Journal of Theoretical and Applied Information Technology*, 37 (2), 171-176, 2012.

[15] K. Hornik, "Multilayer feedforward networks are universal approximators", *Neural Networks*, 2, pp. 359–366, 1989.

[16] K. Hornik, M. Stinchcombe and H. White, "Multi-layer feedforward networks are universel approximators", *Neural Networks* 2, 359-366, 1989.

[17] J. Hertz, A. Krogh and R. G. Palmer, "Introduction to Theory of Neural Computation", *Santa Fe Institute Studies in the Sciences of Complexity. Addison Wesley, Wokingham, England*, 1991.

[18] T.Y. Kwok and D. K. Yeung, "Constructive algorithms for structure learning in feed forward, neural networks for regression problems", *IEEE Trans. Neural Networks* 8, 630-645, 1997.

[19] X. Liang, "Removal of hidden neurons in multilayer perceptrons by orthogonal projection and weight crosswise propagation", *Neural Comput. & Applic.*, 16, 57-68, 2007.

[20] T. B. Ludermir, A. Yamazaki and C. Zanchettin, "An Optimization Methodology for Neural Network Weights and Architectures", *IEEE Transactions On Neural Networks*, vol. 17, no. 6, 2006.

[21] M. Minsky and S. Papert, "Perceptrons", *MIT Press, Washington DC.*, 1969.

[22] A. Mishra, Zaheeruddin, "Design of Fuzzy Neural Network for Function Approximation and Classification," *IAENG International Journal of Computer Science*, vol. 37, no. 4, pp. 326-340, 2010.

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

[24] D. E. Rumelhart, G. E. Hinton and R. J. Williams, "Learning internal representation by error propagation", *Prallel Distributet Processing, Rumelhart D. E and McClelland L., EDS. Cambridge MA, L+MIT Press*, vol. 1, pp 318-362, 1986.

[25] A. C. Subhajini and T. Santhanam, "Fuzzy Artmapneural Network Architecture For Weather Forecasting", *Journal of Theoretical and Applied Information Technology*, 34. (1), 022-028, 2011.

[26] T. Westerlund and F. Petersson, "A Cutting Plane Method for Solving Convex MINLP Problems", *Computers Chem. Eng.*, 19, 131–136, 1995.

[27] Yu J., Wang S. and L. Xi, "Evolving artificial network using an improved PSO and DPSO, , *Neurocomputing* vol 71, no 4-6, pp 1054-1060, 2008.

[28] X. Zeng and D.S. Yeung, "Hidden neuron pruning of multilayer perceptrons using a quantified sensitivity Measure", *Neurocomputing*, 69, 825-837, 2006.

[29] C. Zanchettin, T. B. Ludermir and L. M. Almeida, "Hybrid Training Method for MLP:Optimization of Architecture and Training", *IEEE Transaction On Systems, Man, And Cybernetics*.

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