www.ijecs.in
*International Journal Of Engineering And Computer Science ISSN:2319-7242*
*Volume - 3 Issue - 8 August, 2014 Page No. 7748-7763*

# Adaptive Constructive Algorithm for Artificial Neural Networks

***Prof. (Ms) A. B. Shikalgar[1], Prof. (Mrs) A. N. Mulla[2], Prof. T. A. Mulla[3]***

[1]Department of Computer Science and Engineering,
Dr. J. J. Magdum College of Engineering, Jaysingpur, India.
*annu.shikalgar@gmail.com*
[2]Department of Computer Science and Engineering,
Annasaheb Dange College of Engg. and Technology, Ashta, India
*mulla.anis@gmail.com*[3]
Department of Information Technology,
Dr. J. J. Magdum College of Engineering, Jaysingpur, India.
*mullatahseen@gmail.com*

**Abstract:** The artificial neural networks (ANNs) generalization ability is greatly dependent on their architectures. For a given problem constructive algorithms provide an attractive automatic way of determining a near-optimal ANN architecture. Many algorithms have been proposed in the literature and shown their effectiveness. In automatically determining ANN architectures our research work aims at developing a new constructive algorithm (NCA). NCA puts emphasis on architectural adaptation and functional adaptation in its architecture determination process as in most previous studies are determining ANN architectures. It uses a constructive approach to determine the number of hidden layers in an ANN and of neurons in each hidden layer. NCA trains hidden neurons in the ANN by using different training sets that were created by employing a similar concept used in the boosting algorithm, so as to achieve functional adaptation. The purpose of using different training sets is to encourage hidden neurons to learn different parts or aspects of the training data so that the ANN can learn the whole training data in a better way. In the research the convergence and computational issues of NCA are analytically studied. The experimental result in the research shows that, NCA can produce ANN architectures with fewer hidden neurons and better generalization ability compared to existing constructive and non constructive algorithms.

**Keywords:** NCA, ANN, Constructive approach, Functional adaptation.

## 1. Introduction

Learning from data with complex non-local relations and multimodal class distribution for widely used classification algorithms is still being researched. Even if accurate solution is found with complex classifying function, it may not be able to generalize to other situations. Artificial neural networks are biologically inspired models of computation. They are networks with elementary processing units called neurons massively interconnected by trainable connections called weights. ANN algorithms involve training the connections weights through systematic mathematical algorithms. Learning in ANN has two objectives: determining the optimal network topology and calculating weights by providing known outputs. The generalization ability of artificial neural networks (ANNs) is greatly dependent on their architectures. There are various classes of algorithms to determine the ANN architectures such as constructive, pruning, constructive-pruning and evolutionary algorithms. Thus, the problem of architecture determination for an ANN can be formulated as an optimization process. The solution of this optimization problem is the "complete" topological information of the ANN consisting the number of hidden layers and number of neuron nodes in each hidden layer.

Our research work aims at exploring the constructive algorithms to determine the complete topological information and some particular weight values of an ANN. This approach would be based on combining architectural adaptation with functional adaptation in one scheme. It starts adaptation with a minimal ANN architecture and the whole

training data. As adaptation progresses, NCA gradually adds hidden neurons or hidden layers to the ANN and excludes the examples of training data that are already learned. NCA's emphasis on architectural adaptation and functional adaptation can improve the performance of an architecture determination process [2].

## 2. Need of Work

Many types of neural network models have been proposed for function approximation (pattern classification and regression problems). Among them the class of multilayer feed-forward neural networks (FNN's) is the most popular due to the flexibility in structure, good representational capabilities (universal approximators), and large number of available training algorithms. In general, the learning accuracy, the generalization ability and training time of supervised learning in FNN's depend on various factors such as chosen network architecture (number of hidden nodes and connection topology between nodes), the choice of activation function for each node, the choice of the optimization method and the other training parameters (like learning rate, initial weights etc.). The architecture of the network is either fixed empirically prior to training or is dynamically adjusted during training of the network for solving a specific problem. If the chosen network architecture is not appropriate for the fixed size network, then under-fitting or over fitting takes place.

For better generalization performance and lesser training time, neither too small nor too large network architecture is desirable. We need sufficient number of trainable parameters (weights, biases and parameters associated with activation function) to capture the unknown mapping function from training data.

## 3. New Constructive Algorithm (NCA)

NCA is used to determine the complete topological information of feed-forward ANN architectures with sigmoid hidden neurons. Each hidden layer of such architecture receives signals from all preceding layers (i.e., the input plus the preceding hidden layers), whereas the output layer receives signals from all hidden layers (see Fig. 1). The reason for using these particular traversal paths for signals is to facilitate a fair comparison with previous work. In fact, NCA has no constraint on the type of ANNs. The feed-forward ANNs do not have to be strictly layered, fully connected between adjacent layers, or be of any other connectivity. They may also contain hidden neurons with different activation functions.

### 3.1 Overview of NCA

Unlike most previous studies on determining ANN architectures, NCA puts emphasis on architectural adaptation and functional adaptation in its architecture determination process. It uses a constructive approach to determine the number of hidden layers in an ANN and of neurons in each hidden layer. To achieve functional adaptation, NCA trains hidden neurons in the ANN by using different training sets that were created by employing a similar concept used in the boosting algorithm. The purpose of using different training sets is to encourage hidden neurons to learn different parts or aspects of the training data so that the ANN can learn the whole training data in a better way.

Our research work aims at exploring the constructive algorithms to determine the complete topological information and some particular weight values of an ANN. This approach would be based on combining architectural adaptation with functional adaptation in one scheme. It starts adaptation with a minimal ANN architecture and the whole training data. As adaptation progresses, NCA gradually adds hidden neurons or hidden layers to the ANN and excludes the examples of training data that are already learned. NCA's emphasis on architectural adaptation and functional adaptation can improve the performance of an architecture determination process.

### 3.2 Objectives of NCA

- To understand the working and application of Neural network
- To explore the architecture of Neural Network topologies
- To implement the new constructive algorithm for automatically determining in the architecture parameters of neural network.

- To test the NCA algorithm using suitable dataset and application based on classification and approximation problems
- To observe the effect of each parameters of architecture of neural network

### 3.3 Steps of NCA

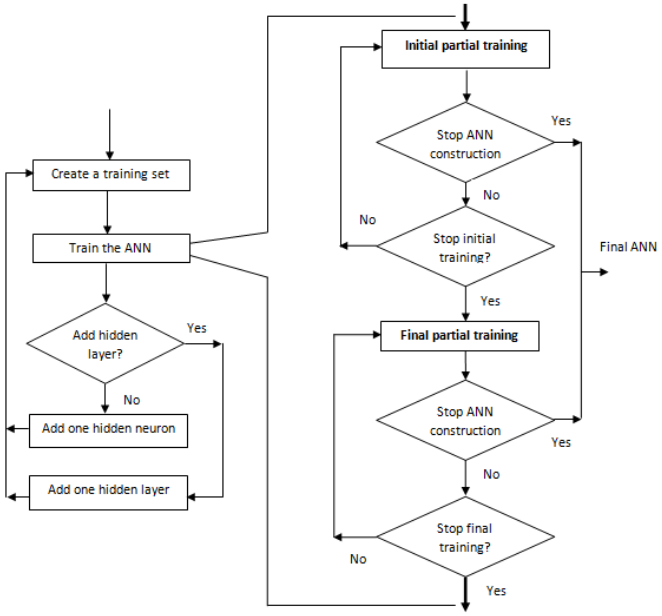


**Fig 1: Steps in NCA**

Step 1) **Create Initial ANN architecture**: Choose a minimal ANN architecture with three layers: 1) an input layer; 2) a hidden layer; and 3) an output layer. The number of neurons in the input and output layers is the same as the number of inputs and outputs of a given problem, respectively. Initially, the hidden layer contains one neuron. Randomly initialize all connection weights of the ANN within a small range. Label the hidden layer and its neuron with C and I, respectively.

Step 2) **Create a Training set:** Create a new training set for the I-labeled hidden neuron of the C-labeled hidden layer. Note that NCA does not create any training set for the first I-labeled hidden neuron of the first C-labeled hidden layer. The original training set is considered here as the new training set for this hidden neuron.

**Initial and final training:**

Step 3) Train the I-labeled hidden neuron of the C-labeled hidden layer on the new training set using the back propagation (BP) learning algorithm for a certain number of training epochs. The number of training epoch's $\tau$ is specified by the user. We call this training phase as the initial partial training for the I-labeled hidden neuron.

Step 4) Check the termination criterion for stopping the ANN construction process. If this criterion is satisfied, go to Step 11). Otherwise, continue.

Step 5) Compute the ANN error E on the training set. If this error reduces by a predefined amount _1 after training for $\tau$ epochs, go to Step 3). It is assumed that training is progressing well and further training is necessary. Otherwise, continue. E is calculated as

$$E = 100 \frac{O_{max} - O_{min}}{NPt} \sum_{p=1}^{Pt} \sum_{i=1}^{m} (Y_i(p) - Z_i(p))^2$$

Where, $O_{max}$ and $O_{min}$ are the maximum and minimum values of the output coefficients in a problem representation, Pt is the number of examples in the training set, and m is the number of output neurons. The value for $O_{max}$ and $O_{min}$ could be same as the maximum and the minimum target values respectively, i.e., 1 and 0, for classification problems.

Step 6) Add a small amount of noise to all input and output connection weights of the I-labeled hidden neuron of the C-labeled hidden layer. Gaussian distribution with a mean of zero and a variance of one is used to add a small amount of noise. It is worth mentioning that NCA adds noise to the connection weights of any I-labeled hidden neuron only once. Train the I-labeled hidden neuron using BP for $\tau$ epochs. We call this training phase as the final partial training for the I-labeled hidden neuron.

Step 7) Check the termination criterion for stopping the ANN construction process. If this criterion is satisfied, go to Step 11). Otherwise, continue.

Step 8) Compute E on the training set. If E is reduced by an amount _1 after training τ epochs, go to Step 6) for further training of the I-labeled hidden neuron. Otherwise, freeze (keep fixed) the input and output connection weights of the I-labeled hidden neuron, remove the label of the I-labeled hidden neuron, and continue.

Step 9) Add **hidden layer? :** Check the criterion for adding a new hidden layer. If the criterion is not satisfied, add a new neuron to the C-labeled hidden layer and go to Step 2). Label the new neuron with I and initialize its input and output connection weights with zero. Otherwise, remove the label of the C-labeled hidden layer and continue. The reason for initializing the weights with zero is to start further training from the previous error value.

Step 10) **Add one hidden neuron and layer:** Add a new hidden layer with one neuron above the existing hidden layer(s) of the ANN. Label the new hidden layer with C and its neuron with I. Initialize the input and output connection weights of the neuron with zero and go to Step 2).

Step 11) The existing network a architecture is the final ANN for the given problem.

## 4. Experimental Setup

The program is implemented in Maltab software with a smart IDE. The experiment is carried using Matlab (R2011b) on a single machine using with Windows 32-bit operating system.

## 5. Performance Evaluation

### 5.1 Back Propagation Neural Network (BPNN)

BPNN is a method of training ANN, so as to minimize the objective function. It reduces the number of parameters or makes them adaptive.

### 5.2 Back Propagation Algorithm

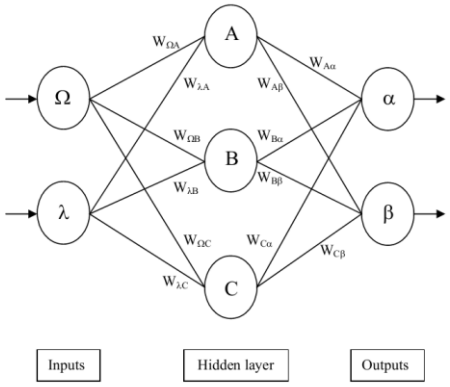


**Fig 2 : Back Propagation Algorithm**

### 5.3 Performance of BPNN

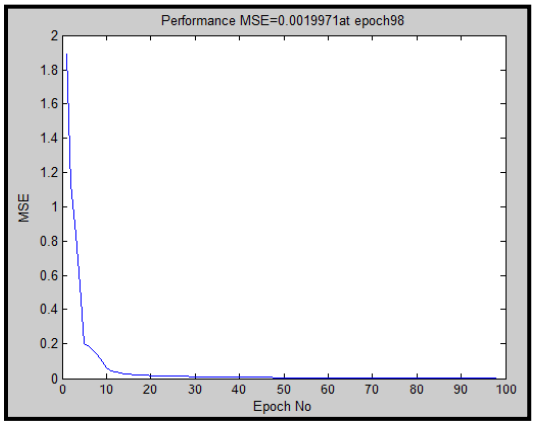


**Fig 3: BPNN performance**

### 5.4 Performance of IRIS dataset

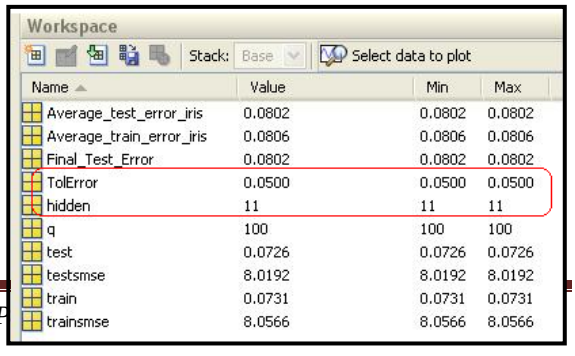

**Fig 4 : Performance of IRIS dataset**

### 5.5 Performance of KRKP dataset

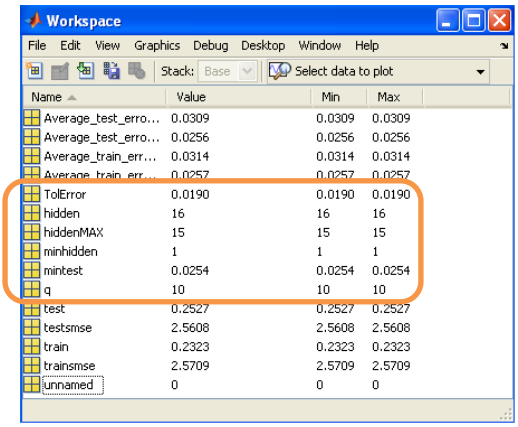


**Fig 5 : Performance of KRKP dataset**

## 6. Results and Analysis

### 6.1 Analytical Results based on the results of Neural Network achieved

After obtaining the Neural Network Parameters for the tested dataset, the work has carried out in the direction to the test the effect of number of epochs over the training results.

#### 6.1.1 Error Rates on various Datasets

Following table indicates the results of the error rates for the Iris and KrKp datasets taken over the scheme as follows

1. Iris Training Dataset – 90% . Testing Dataset – 10%
2. KrKp Training Dataset – 80%. Testing Dataset – 20%

**Table 1 – Error Rates on various datasets**

| Name of Dataset | Training Error Rates | Testing Error Rates | Epochs Considered |
|---|---|---|---|
| Iris | 14.46 | 14.3 | 80 |
| KRKP | 9.94 | 18.63 | 20 |

The above extracts of results are taken into consideration with fixed number of epochs as mentioned in the table. But the results are calculated as an average of 100 runs of the program execution.


**Fig 6 : Graph for Different Datasets v/s Error Rates**

#### 6.1.2 Reduction in Error with increase in number of epochs.

We can also analyze the system on the basis of Increase in number of epochs. If we increase the number of epochs and train the system in an effective manner it is observed that the error rate further reduces and reaches to its local minima. This feature of Back propagation is proven on the NCA Algorithm too by following results. The following results are with a consideration of 100% dataset utilization for the sake of training.

**Table 2 – Reduction in errors**

| Number of Epochs | IRIS | KRKP |
|---|---|---|
| 100 | 9.0202 | 9.8907 |
| 200 | 5.7275 | 6.4859 |
| 400 | 3.3741 | 4.1892 |
| 600 | 3.0026 | 3.3557 |
| 800 | 2.915 | 2.9401 |
| 1000 | 2.9224 | 2.6667 |

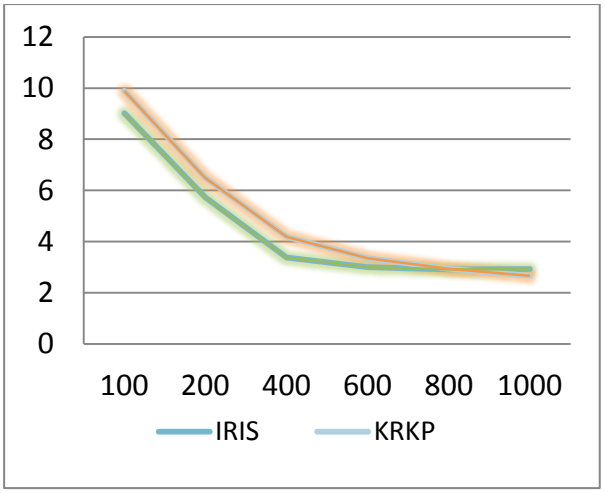


**Fig 7 : Reduction in Error with increase in number of epochs**

## 7. Conclusion

A new constructive approach is used to automatically determine ANN's architecture in our research work.

NCA is used to focus more on complete architectural adaptation and functional adaptation rather than only partial architectural adaptation. For Ex. The proposed work of our research determines not only the number of hidden neurons in an ANN but also the number of neurons in each hidden layer to achieve complete architectural adaptation. This work freezes the connection weights of a previously added hidden neuron and creates a new training set when a new neuron is added to the ANN. For emphasizing on unsolved parts of the training data a new training set is created. The hidden-layer-addition criterion of NCA incorporates the functionality of hidden neurons' along with training error.

## References


1. Islam, M.M.; Sattar, M.A.; Amin, M.F.; Xin Yao; Murase, K.; , "A New Constructive Algorithm for Architectural and Functional Adaptation of Artificial Neural Networks," Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on , vol.39, no.6, pp.1590-1605, Dec. 2009
2. L. Ma and K. Khorasani, "Constructive feedforward neural networks using Hermite polynomial activation functions," IEEE Trans. Neural Netw., vol. 16, no. 4, pp. 821–833, Jul. 2005
3. Kwok, T. Y.; Yeung, D. Y.: Constructive Algorithms for Structure Learning in feedforward Neural Networks for Regression Problems. IEEE Transactions on Neural Networks, 8 (3), 1997, 630-645.
4. Fahlman, S. E.; Lebiere, C.: The cascade correlation learning architecture. Advances in Neural Information Processing System 2, D. S. Touretzky, Ed. CA: Morgan Kaufmann, 1990, 524-277.
5. Ash, T.: Dynamic node creation in back-propagation networks. Connection Science, vol. 1, no. 4, 1989, 365-375.
6. Friedman, J. H.; Stuetzle, W.: Projection pursuit regression. J. Amer. Statist. Assoc., vol. 76, no. 376, 1981, pp. 817-823.
7. Platt, J. : A resource-allocating network for function interpolation. Neural Computation, vol. 3, 1991, pp. 213-225.
8. Farlow, S. J. Eds.: Self-Organizing Methods in Modeling: GMDH Type Algorithms, vol. 54 of Statistics: Textbooks and Monographs. New York: Marcel Dekker, 1984.
9. Nabhan, T. M.; Zomaya A. Y. : Toward generating neural network structures for function approximation. Neural Networks, vol. 7, no. 1, 1994, pp. 89-90.
10. Subirats, J. L.; Jerez, J. M. ; Franco L. : A new decomposition algorithm for threshold synthesis and generalization of Boolean functions. IEEE Transaction on Circuits and Systems I 55, 2008, pp. 3188-3196.
11. Fahlman, S. E.; Lebiere, C.: The cascade correlation learning architecture. Advances in Neural Information Processing System 2, D. S. Touretzky, Ed. CA: Morgan Kaufmann, 1990, 524-277.


## Author Profile

**Prof. (Ms.) A. B. Shikalgar –** Perceived BE degree in Information Technology from Padmabhooshan Vasantraodada Patil Institute of Technology, Budhgaon, Shivaji University, Kolhapur, and pursuing ME degree from same University, Currently working as an Assistant Professor in the Department of Computer Science & Engineering, Dr. J. J. Magdum College of Engineering, Jaysingpur, Shivaji University, Kolhapur. Research interests are in the field of Cloud Computing and Artificial Neural Network.

**Prof. (Mrs.) A. N. Mulla –** Perceived M.E. degree from Walchand College of Engineering, Sangli, Shivaji University, Kolhapur. Currently working as an Assistant Professor in the Department of Computer Science and Engineering, Annasaheb Dange College of Engineering and Technology, Ashta, Shivaji University, Kolhapur. Research interests are in the field of Image Processing and Artificial Neural Network

**Prof. T. A. Mulla–** Perceived BE degree in Information Technology from Padmabhooshan Vasantraodada Patil Institute of Technology, Budhgaon, Shivaji University, Kolhapur, and perceived M. Tech degree in Computer Science & Engineering from St. Mary's College of Engineering and Technology, Hyderabad, Jawaharlal Nehru Technological University, Currently working as an Assistant Professor in the Department of Information Technology, Dr. J. J. Magdum College of Engineering, Jaysingpur, Shivaji University, Kolhapur. Research interests are in the field of Cloud Computing and Artificial Neural Network.