Modeling of Hidden Layer Architecture in Feed-forward Artificial Neural Networks

N. M. Wagarachchi[1], A. S. Karunananda[2]
[1,2]Faculty of Information Technology, University of Moratuwa.
[1]mihirini@is.ruh.ac.lk, [2]asoka@itfac.mrt.ac.lk

Abstract— **Determining the appropriate architecture of a neural network is one of the main unsolved problems in artificial neural networks. The architecture has a great impact on its generalization power. More precisely, by changing the number of layers and neurons in each hidden layer, generalization ability can be significantly changed. Therefore, the architecture is crucial in artificial neural network and determining the hidden layer architecture has become a research challenge. In this paper, a pruning technique is presented to obtain an appropriate architecture by using the delta values of hidden layers. Pruning is done by using the delta values of hidden layers. The proposed method has been tested with three benchmark problem datasets in artificial neural networks and machine learning namely, breast cancer, Iris and car evaluation. The experimental results show that the modified architecture with lesser number of neurons performs better in generalization than that of the back-propagation algorithm.**

1. INTRODUCTION

Artificial neural networks (ANN) have been applied in many real world problems such as signal processing, pattern recognition, and medical diagnosis problems [1], [2], [3], [4]. Although they provide highly generalized solutions, we find several unanswered problems when using ANNs. Determining the most appropriate architecture of artificial neural network is identified as one of those major problems because, the performance of a neural network strongly depends on the size of the network. By increasing the number of layers generalization ability can be improved. But this solution may not be computationally optimized. On the other hand, too many hidden neurons may over-train the data which may cause poor generalization. Also, too few neurons under-fit the data and hence, network may not train the data properly. Thus, too many or too few neurons results in bad generalization. Therefore, determining the most suitable architecture is very important in artificial neural networks. As such, a large number of research have been carried out to model the hidden layer architecture by using various techniques. These techniques can be categorized as pruning methods, constructive methods and evolutionary methods. Pruning algorithms start with an oversized network and remove nodes until the optimal architecture occurs [5],[6],[7],[8],[9]. Constructive algorithms [10],[11],[12],[13] work the other way round. They build the appropriate neural network during the training process by adding hidden layers, nodes and connection weights to a minimal architecture. However, most of these methods are confined to networks with small number of neurons or a single hidden layer neurons and hence, they have not addressed the existing problem of hidden layer architecture properly.

In this paper, we have proposed a novel pruning algorithm to model hidden layer architecture based on back-propagation training. The algorithm reaches to the optimal architecture in two phases. Initially, it decides the number of hidden layers required for the best generalization. Usually, generalization power can be improved by increasing the number of hidden layers. However, to train a network with a large number of hidden layers, it needs more effort and more time. Therefore, it is essential to decide the number of hidden layers for the most effective neural network. To decide this number we use the 'Accuracy Ratio' (κ). The number of hidden layers corresponds to the greatest κ is to be considered as the number of hidden layers in the most effective network. In the second phase, algorithm prunes the network by removing unimportant nodes from the hidden layers. Accordingly, we selected to delta values of hidden layers to identify the unimportant nodes. This is based on the fact that the delta values of the hidden layers are used to compute the error term of the next training cycle. Hence, delta value is a significant factor in the error term. Thus, delta values are used to identify the less saliency neurons and remove them from hidden neurons, so that error term comes to desired limit faster than the backpropagation training. Moreover, delta depends on the several other parameters such as connection weights learning rate and the activation function. Hence, it automatically considers these parameters when removing the nodes from hidden layers. In [14] authors have discussed how to use delta values in optimization of hidden layer architecture. This paper presents a modified algorithm based on a pruning technique in designing artificial neural networks. This algorithm determines the optimal

solution by removing unimportant nodes from feed-forward multilayered artificial neural networks.

The rest of the paper is organized as follows. The related works done by other researchers are discussed in section II. Section III describes the new algorithm and how to use delta values in optimization of hidden layer architecture. The experimental method and results are discussed in section IV. Finally, section V presents the conclusions.

2. RELATED WORKS

As the architecture of artificial neural network is very important there are many attempts in modeling hidden layer architecture. These approaches are based on pruning techniques, constructive techniques or both pruning-constructive techniques and some other evolutionary methods.

2.1 Pruning Algorithms

The objective of a pruning algorithm is to obtain the optimum structure of network by incrementally removing low-saliency neurons from the architecture. Optimal Brain Damage (OBD), presented by Le Cun and Jhon Denker [5] is known as the most well-known and commonly used pruning algorithm, where parameters with low-saliency are detected based on the second derivative of the objective function and removed from the network. The intention of OBD is that, it is possible to obtain a network which perform in same manner (or better), by eliminating about half of its weights. However, a major issue arises with the enormous size of the Hessian matrix. This causes computational barriers and also takes considerable time to train. Hence, it assumes the Hessian matrix is diagonal.

To avoid the limitations of OBD, authors in [6] introduced Optimal Brain Surgeon (OBS) and claimed that Hessian matrix is non-diagonal for every problem and hence, weights elimination of OBD may not be accurate. Even though, it argues that OBS is better than OBD and yields better generalization, much computational limitations arises especially working with the inverse of the Hessian matrix.

Giovanna et al. [7] proposed a conjugate gradient algorithm in least-square sense to prune neurons after training the network. It reaches the optimal size by removing neurons successfully from a large trained network and then adjusting the remaining weights to maintain original input-output behavior. They claim that this algorithm yields better results relative to other existing methods. However, in this research it does not consider a multi-layered architecture and hence, it deviates from our aim. On the other hand, our goal is to identify and prune the neurons before training the network. In addition, we concentrate on removing a cluster of neurons rather than removing one neuron at a time.

Authors in [8] have used hybrid sensitive analysis and adaptive particle swarm optimization (SA-APSO) algorithm to determine the minimal architecture of artificial neural networks. Firstly, prune neurons with less saliency using the impact factor defined by,

$$\boldsymbol{ImF_i = \sum w_{ji}^2 \sigma_i^2}$$

The neurons with *ImF* less than a certain threshold value $\boldsymbol{\sigma_{ir}}$ have been identified as removable neurons with less saliency. Each removal is tried to replace with some other suitable neuron which has the similar effect. The similarity is defined by using the correlation coefficient. If the two neurons are highly correlated they can be merged. Anyhow, this research is restricted only for two layered networks. But there is no such rule that two layer network can solve almost all the problems. Also there are some problems which require three layer networks [15].

Reference [16] claims that for a particular network configuration, there is a continuous set of weights and biases that have the same value of the cost function

$$\sum_{n,p} (d_{pn} - o_{pn})^2.$$

Where the sum is taken over n outputs and p training patterns while d and o are desired and observed outputs respectively. This set of weights defines a contour of the cost function in the space of all possible weights. In order to determine the optimum architecture, a network trained by Back propagation algorithm can be obtained by eliminating weights which are close to zero.

2.2 Constructive Algorithms

In constructive neural networks the network structure is built during the training process by adding hidden layers, nodes and connections to a minimal neural network architecture. However, determining whether a weight should be added and

whether it should be added to the existing hidden layer of a new layer is not straightforward. Therefore in most algorithms, pre-defined and fixed number of nodes are added in the first hidden layer and the same number of nodes are added to second layer and so on [12]. This number is crucial for the better performance of the network and it reduces the network size as much as possible to avoid the complex computations during the training. The cascade correlation algorithm (CCA), proposed by S. E. Falhman [10] is a well known and mostly used constructive algorithm. This algorithm is proposed as a solution to problems such as local minima problem. The dynamic node creation (DNS) [11] algorithm is supposed to be the first constructive algorithm for designing single layer feed-forward networks dynamically. Authors in [13] improved the adaptive learning algorithm for multi-category tiling constructive neural networks for pattern classification problems. Md. Moniral et al [17] have proposed an adaptive merging and growing algorithm to design artificial neural networks. This algorithm merges and adds hidden neurons repeatedly or alternatively during the training, based on the learning ability or the training progress of the artificial neural network.

2.3 Evolutionary Methods

Other than the pruning and constructive methods some researchers have suggested evolutionary methods to model the hidden layer architecture based on various mathematical concepts.

An alternative method proposed by Stathakis [18] was to obtain a near-optimal solution by searching with a genetic algorithm. This research was carried out to determine the optimum architecture and it claims that obtained network is more efficient in problem classification. Although it gives a very small error in the training cycle, topology of network is comparatively large. Hence, it is difficult to apply especially when there is large number of inputs in the network. In [19] authors have presented an approach selecting the best number of hidden units based on some mathematical evidence. According to their suggestion number of hidden nodes can be given as

$$n = C\left(\frac{N}{dlogN}\right)^{1/2}$$

Where N is the number of training pairs and C is a constant. The approach is to increase C and train the feed-forward network for each n and then determine the optimal value for n which gives the smallest rooted mean square error. Although, they have applied this to medium-sized data set and there is no guarantee of applying the theory for large number of hidden neurons.

Even though methods to optimize the hidden layer architecture have been developed, they have not fully addressed the problem regarding the topology of multilayer artificial neural networks such as determining the number of required layers for better performances and then minimize the error in training.

3. NEW ALGORITHM

In this research we propose a new algorithm based on the Backpropagation training algorithm. Backpropagation algorithm is the most well-known and widely used among the optimization algorithms of artificial neural networks. Our proposed algorithm optimizes the artificial neural networks based on a pruning technique and reaches to the desired solution faster than the Backpropagation training. The main objective of the new algorithm is, while maintaining the same error rate; it decreases the size of the network and the number of training cycles with better performances.

At the first stage, it decides the number of hidden layers which gives the most efficient network. Then removes unimportant nodes from each layer to reach the optimal solution. Here we consider a network with n inputs and m outputs. The number of input/output training cycles is N. Let each network consists of M total number of hidden neurons. Then, train the different architectures of network by backpropagation algorithm. The number of hidden layers of network is h, where h = 1,2,3,... . The number of hidden neurons in each layer is $n_{H_i}, i = 1, 2, \dots, h$ where

$$\sum_i n_{H_i} = M$$

Define Accuracy ration (κ)

Where,

$$\kappa = \frac{Accuracy\ Rate}{Number\ of\ training\ cycles}$$

Then higher values of κ indicates that the network is more efficient. Hence, the number of hidden layers that provides the highest κ is to be considered as the most appropriate network.

After deciding the exact number of hidden layers (h), trim the network for better performance. Start with fully connected network with h hidden layers and M number of total hidden neurons. Initially, it identifies the less saliency neurons and removes from hidden layers during the training. After removing all the unimportant nodes network trains by using backpropagation algorithm until error $E(n)$ defines in (1) tends to zero.

Consider a fully connected network with n inputs, m outputs and h hidden layers. Hidden layers are denoted by $H_1, H_2, \dots H_h$ and the layer H_j contains n_{H_j} number of neurons.

The error of the n^{th} training cycle $E(n)$ can be written as

$$E(n) = \frac{1}{2}\sum_{k=1}^{m}(t_k(n) - o_k(n))^2 \qquad (1)$$

Where, t_k and o_k. denote the expected output and actual output on the neuron k respectively.

Let the error $e_k(n)$ be the difference between the desired and actual outputs of the k^{th} neuron at the n^{th} training cycle. That is

$$e_k(n) = t_k(n) - o_k(n) \qquad (2)$$

Then

$$E(n) = \frac{1}{2}\sum_{k=1}^{m}\left(e_k(n)\right)^2 \qquad (3)$$

When there are N input/output training sets total error of the network becomes

$$E_N = \frac{1}{2N}\sum_{p=1}^{N}\sum_{k=1}^{m}\left(t_{pk} - o_{pk}\right)^2 \qquad (4)$$

The delta value of the output layer is defined as

$$\delta_k = f_o'(net_k)(t_k - o_k)$$

$$= f_o'(net_k)e_k(n) \qquad (5)$$

Where f_o is the activation function of the output layer. The delta value of the i^{th} neuron of the j^{th} hidden layer can be written as

$$\delta_i^{H_j} = f_{H_j}'(net_i)\sum_{k=1}^{n_{H_{j+1}}} w_{ki}\,\delta_k \qquad (6)$$

Where f_{H_j} is the activation function defined for the hidden layer H_j. w_{ki} is the connection weight from i^{th} neuron of the j^{th} layer to k^{th} neuron of $(j+1)^{\text{th}}$ layer. δ_k is the delta value of k^{th} neuron in the hidden layer H_{j+1}. So that, the delta value of each neuron is computed by using the delta values of next adjacent hidden layer. After computing all delta values weights are updated according to

$$w_{kj}^{H_j}(n+1) = w_{kj}^{H_j}(n) + \eta\delta_k^{H_j}(n)f_{H_j}(n) \qquad (7)$$

Error $E(n+1)$ will be calculated by using the updated weights. Hence, $E(n+1)$ has been computed by using $\delta(n)$. Therefore, it is clear that there is a relation between $\delta(n)$ and $E(n+1)$.

The procedure begins with a fully connected multilayered artificial neural network. This approach is not restricted to only one hidden layer. Therefore, we consider a network with h number of hidden layers. Each layer contains a large number of hidden neurons. There are various upper bounds for the number of hidden neurons in ANNs [13],[15]. So that we have assigned a large number ($>N/h$) of neurons for each hidden layer. This number is much more than the maximum of the above upper bounds.

The delta values of hidden layers of the n^{th} training cycles $\left(\delta_i^{H_j}(n)\right)$ are used to compute the error of the $(n+1)^{\text{th}}$ training cycle $E(n+1)$. Thus, the correlation coefficient between the summations of delta values of hidden layer H_j at the n^{th} training cycle $\sum\delta_i^{H_j}(n)$ and $E(n+1)$ has been used to identify the less salient neurons. For our convenience, this correlation is denoted by r_{E,H_j}. i.e.o

$$r_{E,H_j} = corr\left(\sum_{i=1}^{n_{H_j}}\delta_i^{H_j}(n), E(n+1)\right) \qquad (8)$$

A positive r_{E,H_j} implies that eliminating positive nodes from the hidden layer H_j, summation of delta values decreases and hence, $E(n+1)$ can be reduced. If r_{E,H_j} is negative, by removing neurons with negative delta values summation of delta values increases. Therefore error can be reduced. Thus eliminating nodes with positive or negative delta values according their correlation error $E(n+1)$ becomes smaller than that of earlier. Therefore, it converges faster than the backpropagation. On other words number of the training cycle requires to reach the solution becomes lesser relative the backpropagation training.

Equation (7) implies that when delta is zero

$$w_{ki}^{H_j}(n+1) = w_{ki}^{H_j}(n)$$

i.e. there is no update of weight $w_{ki}^{H_j}$. Therefore, neurons with zero delta values do not have significant effect in minimizing the error. So that by removing nodes with zero delta values, size of the network can be reduced without degrading the performance of the network. Therefore, by removing neurons with,

Small positive values when $r_{E,H_j} > 0$

and

Large negative values when $r_{E,H_j} < 0$

The size of the network can make smaller while reaching the desired solution faster than the backpropagation training. The method of removing nodes is described in the following algorithm.

3.1 The New Algorithm

Step 1: Design fully connected networks for hidden layers , 2, 3, ... Each contains *n* input and *m* output vectors respectively. Initiate the connection weights which are randomly chosen and normalized. Train the network using backpropagation algorithm for *N* input/output datasets and computeκ. *h* is the number of hidden layer of the network, which provides them maximum κ.

Step 2: Compute $r_{E,H_j} \quad j = 1,2 \ldots h$

Step 3: Apply the first input and by using the same set of connection weights compute error,oo

$$E = \frac{1}{2}\sum_{k=1}^{m}(t_k - o_k)^2$$

Step4: Find the delta values of the last hidden layer H_h. If the correlation r_{E,H_h} is positive remove neurons with positive delta values those less than certain threshold value τ.i.e remove j^{th} neuron from H_h if ,

$$0 < \delta_j^{H_h} < \tau.$$

If r_{E,H_h} is negative, remove neurons with negative delta values if

$$\tau < \delta_j^{H_h} < 0$$

Let the remaining number of hidden olayers in H_h. be $n_{h'}$

Step 5: Use these delta to calculate delta of the neurons of (h-1)th layer. Remove all possible neurons as in step 4.

Step 6: update weights

$$w_{ki}^{H_j} = w_{ki}^{H_j} + \eta\delta_i^{H_j} f_{H_j}(net)$$

Step 7: Apply the next input and continue steps 3 - 6.

Step 8: Compute

$$E_N = \frac{1}{2N}\sum_{p=1}^{N}\sum_{k=1}^{m}(t_{pk} - o_{pk})^2$$

Step 9: Repeat the procedure until the optimum solution occur.

Step 10: After obtaining the optimum solutiono train the optimal architecture by using the back propagation algorithm.

4. EXPERIMENTS AND RESULTS

Experiments on deciding appropriate number of hidden layers and obtaining the optimal solution are discussed in this section. For these experiments, we used datasets chosen from three well-known benchmarking problems namely; car evaluation, breast cancer and Iris. More details on these data sets are available on [21] and [22]. These data sets have been used to explain many studies in artificial neural networks and machine learning. All the sets were divided in to two classes for training and testing purposes. Different architectures of the above problems were considered. The log sigmoid and linear functions were chosen as activation functions for hidden layers and output layer respectively. The learning rate in each case was 0.1. Network trained for *N* input/output training sets until

$$E_N = \sum_{p=1}^{N}\sum_{k=1}^{m}(t_{pk} - o_{pk})^2 < 10^{-4}$$

In order to determine the suitable number of hidden layers, training sets of size *N* were chosen from each problem and designed fully connected networks with layers 1, 2, 3 4 and 5. Each network was trained by Backpropagation algorithm. The accuracy of the results was tested by using the testing set and the accuracy ratio(κ) was computed.

Table1 shows the accuracy rate in each problem. h and M refer to the number of hidden layers and total number of hidden neurons respectively. TC denotes the number of training cycles. It is clear that generalization ability is improving when the number of layers is increasing. For example, in car evaluation problem, the accuracy rate of the output was 72.34% when there were two layers.

Table 1: Accuracy Ratio κ for Different Benchmarking Problems

Problem	N	h	M	TC	Accuracy Rate	Accuracy Ratio (κ)
Car Evaluation	100	2	120	16	72.34	4.52
		3	120	23	74.28	3.23
		4	120	521	78.53	0.15
		5	120	312	77.97	0.25
Breast Cancer	120	2	160	7	94.28	13.07
		3	160	7	96.89	13.84
		4	160	17	97.24	5.72
		5	160	55	97.24	1.77
Iris	50	2	60	7	74.46	10.64
		3	60	19	78.70	4.14
		4	60	306	79.33	0.26
		5	60	387	81.93	0.21

However, same data set agrees with 78.53%, for 4 hidden layers. A similar difference can be observed in breast cancer and Iris problems.

However, to train higher number of hidden layers it requires more time. When there are only 2 and 3 hidden layers in the car problem, network can be trained only by 16 and 23 training cycles respectively. But the numbers of training cycles require to train network with 4 or 5 hidden layer are much greater than that. Hence in some cases, the improvement of performance is negligible compared to the effort and time we put to train the networks. It can be described by the accuracy ratio. In this problem, when there are 2 or 3 hidden layers κ is much higher than that for 4 or 5 hidden layers. Hence, we can consider network with two hidden layers with the highest value of κ is the most appropriate network for this problem. Similarly, in cancer and Iris problems, networks with 3 and 2 hidden layers respectively provide the highest values for κ. The decision for hidden layers in each problem is shown in Table 2.

To test the modified algorithm consider the networks with hidden layers given in Table 2. The total number of hidden neurons (M) appearing in Table 1 is divided among the hidden layers as shown in Table 4. While training the data with Backpropagation algorithm, correlation coefficient was observed. Table 3 shows the correlation r_{E,H_j} for each layer. Then the network

Table 2. Number of hidden Layers in the most appropriate network

Problem	N	h
Car Evaluation	100	2
Breast Cancer	120	3
Iris	50	2

Table3. Correlation Coefficients

Problem	N	h	r_{E,H_1}	r_{E,H_2}	$r_{E,,H_3}$
Car Evaluatic	100	2	0.85	−0.23	
Breast Cancer	120	3	−0.29	0.52	0.79
Iris	50	2	0.07	0.42	

	N	Backpropagation Algorithm			Modified Algorithm		
		Initial Configuration	TC	Accuracy Rate	Modified Architecture	TC	Accuracy Rate
Car Evaluation	100	70 - 60	16	72.34	66 - 56	13	72.51
Breast Cancer	120	60 - 50 - 50	7	96.89	42 - 34 - 38	5	97.06
Iris	50	30 - 30	11	78.56	28 - 24	9	81.20

Table 4. Comparison between Backpropagation and New Algorithms

was trained by using the new algorithm. A benchmark comparison was done with Backpropagation algorithm. The required numbers of training cycles for Backpropation and the modified algorithm respectively, are shown in Table 4. Also it shows the number of hidden neurons in initial configuration and the modified architecture. Accuracy rate of each case was calculated. It shows that for every case, it was able to find an architecture which has lesser number of neurons and shows better performance. First example in car evaluation problem shows positive correlation for the 1st layer and negative correlation for the 2nd layer. Hence, by removing neurons with positive small delta values from 1st layer and negative large delta values from the 2nd layer, modified architecture was obtained. It needs 16 training cycles to reach error $E = 10^{-3}$ by backpropagation algorithm. But if we train it by using new algorithm within 13 training cycles, network tends to the desired limit and new model decreases number of hidden neurons by about 9%. Moreover, it has upgraded the accuracy rate from 72.34% to 72.51%. In the car problem it was able to remove about 20% of hidden neurons without degrading the performance.

Also in cancer problem with 3 layers of hidden neurons were reduced by 28% while improving the generalization ability. Fig. 1 shows how the error changes in two training processes. It is clear new algorithm decreases the error rapidly and reaches to the desired error faster than the Backpropagation training.

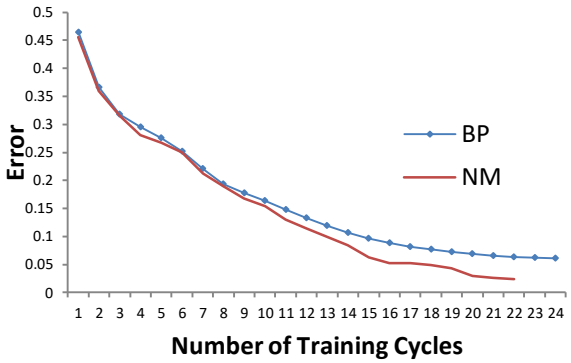

Figure1: The number of training cycles required in each training backpropagation and new algorithm.

5. CONCLUSION

In this paper, a new algorithm for multilayer hidden architecture was proposed. The algorithm is based on a pruning technique. First it decides the number of hidden neurons by using accuracy ratio. Then hidden neurons were pruned by using the delta values of hidden layers. The correlation between the summation of delta values of the n^{th} training cycle and error of the $(n+1)^{st}$ training cycle was considered to identify the less salient neurons. Moreover, neurons with zero delta values were recognized as unimportant neurons as they do not have much effect on updating the weights. Therefore, hidden neurons with small positive or large negative values (depending on the correlation) can be successfully used to reduce the size of the multilayer artificial network. In this paper a benchmark comparison was done with the backpropagation and it demonstrates that a new approach can be used to minimize the network by maintaining the same error rate as back propagation training with lesser number of training cycles. Further, the modified architecture can be obtained with very limited computations. Generally, 5% - 30% of neurons

can be removed from hidden layers without degrading the performance of the output.

However, the performance of the network depends on several parameters such as weights of the connections, learning rate η, number of hidden neurons etc. Hence, some moderate changes will be done to these parameters as future improvements.

REFERENCES

[1] R. Setiono, W. K. Leow and J. M. Zurada, "Extraction of Rules from Artificial Neural Networks for Nonlinear Regression", IEEE Transaction of Neural Network, vol.13 May 2002, pp.564-577.

[2] S. M. Kamruzzaman, A. R. Hasan, A. B. Diddiquee and Md. Ehsanul, "Medical Diagnosis Using Neural Networks" Proceedings of the International Conference on Electrical and Computer Engineering (ICECE-2004) BUET, Dhaka , 2004, pp. 185-204.

[3] S. M. Karuzzaman and Md. M. Islam, "An Algorithm to Extract Rules from Artificial Neural Networks for Medical Diagnosis Problems", International Journal of Information Technology, vol. 12 No. 8,2006, pp. 41-59.

[4] F. Amato,A. Lopez,E. M. Pena-Mendez, P. Vanhara, A. Hample, J Havol, "Artificial Neural Networks in Medical Diagnosis", Journal of Applied Bio-Medicine, pp. 47-58.

[5] Le Cunn Y.,Denkar, Solla S. A. "Optimal Brain Damage," *Advances in Neural Information Processing Systems* D.S. Touaretzky, Ed, San Mateo CA:Vol 2 pp 598-605, 1990.

[6] Hassabi . B., Stork. D. G., "Second Order Derivatives for Network Pruning: Optimal Brain Surgeon," *Neural Information Processing Systems*-vol 5,1993.

[7] Giovanna C., Anna M. F.,Marcello P., "An Iterative Pruning Algorithm for Feedforward Neural Networks," *IEEE Transactions on Neural Networks,* vol. 8, pp. 519-531, May 1997.

[8] Faisal Muhammad Shah, Md. Khairul Hasan, Mohammad Moinul Hoque, Suman Ahmmed, "Architecture and Weight Optimization of ANN Using Sensitive Analysis and Adaptive Paticle Swam Optimization," *IJCSNS International Journal of Computer Science and Network Security,* vol. 10, no. 8, August 2010.

[9] D. Sabo, X. H. Yu, "A New Pruning Algorithm for Neural Network Dimension Analysis", *IEEE International Joint Conference on Neural Networks*, IJCANN 2008, pp.3313 – 3318, Jun. 2008

[10] Fahlman S. E, "The Cascade-Correlation Architecture," May 1991.

[11] M. R. A. S, "Recursive Dynamic Node Creation in Multi Layer Neural Network," *IEEE Transactions on Neural Networks ,* vol. 4, no. 2, 1993.

[12] P. C. Sudir Kumar Sharma, "Constructive Neural Networks: A Review," *International Journal of Engineering Science and Technology,* vol. 2, no. 12, pp. 7847-7855, 2010.

[13] Sridhar. S.S, "Improved Adaptive Learning Algorithm for Constructive Neural Networks," *International Journal of Computer and Electrical Engineering,* vol. 3, no. 1, 2011.

[14] N. M. Wagarachchi, A. S. Karunananda, "Optimization of Multi-layer Artificial Neural Networks Using Delta Values of Hidden Layers," *IEEE Symposium on Computation Intelligence, Cognitive Mind and Brain* , pp. 80-86, April, 2013

[15] B. G. H. Don R. Hush, "Progress in Supervised neural networks," *IEEE Signal Processing,* vol. 10, pp. 8-39, 1993.

[16] A. N. Burkitt, "Optimization of the Architecture of Feed-forward Neural networks with hidden layers by Unit Elimination," *Complex Systems 5,* pp. 371-380, 1991J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68–73.

[17] Md. Monirul Islam, Md. Abdus S, Md. Faujul A, Xin Y., "A New Adaptive Merging and Growing Algorithm for Designing Neural Networksf," *IEEE Transactions on Systems Man and cybernetics,* vol. June 2009.

[18] Stathakis D., "How many Hidden Layers and Nodes," *International Journal of Remote Sensing,* vol. 30, pp. 2133-2147, April, 2009

[19] Sxuxinang Xu, Ling Chen, " Novel Approach for Determining the Optimal Number of Hidden LayerNeurons for FNN's and its Application in Data Mining," *5th International Conference on Information Technology and Applications,* pp. 683 - 686, ICITA 2008.

[20] Baum E. B, Haussler D., "What Size Network Gives Valid Generalization" *Neural Computations*-January,1989.

[21] AFrank, " UCI Machine Learning Repository [.," Irvine, CA: University of California, School of Information and Computer Science., 2010. [Online]. Available: http://archive.ics.uci.edu

[22] L. Prechelt, "PROBEN1—"A set of neural network benchmark problems and benchmarking rules," Faculty Informatics, Univ. Karlsruhe, Germany, Tech. Rep. 21/94,1994