Optimization of Number of Hidden Neurons in Neural Networks

Sarika Shukla[1], Jameel Ahamad[2]

[1]Student M. Tech, EED, Harcourt Butler Technical University, Kanpur

[2]Assistant Professor, EED, Harcourt Butler Technical University, Kanpur

Abstract: ***This paper carries study of Neural Network and analyses the effects of number of neurons in the hidden layer of the network on the output. This paper also contains study of conventional optimization methods for ascertaining the number of hidden neurons and analyses of neural network performance. Implementation of optimization criterion based on estimation of Signal to Noise Ratio Figure, Sum Squared Error and Delta Value method for ascertaining the number of hidden neurons and analyses of neural network performance. Comparison of results by using the methods as mentioned in the above points is presented here.***

Keywords: Neural Network, Signal to Noise Ratio Figure, Sum Squared Error, Delta Value method

1. Introduction

Study of Neural Network is done and structure of a neural network is studied. The significance of hidden neurons is analyzed and the effect of varying the number of hidden units is studied. A certain number of hidden units are taken initially. The number of hidden neurons are optimized using the criterion based on Sum Square Error. The number of hidden neurons are optimized using the criterion based on Signal to Noise Ratio Figure. The number of hidden neurons are optimized using the criterion based on Delta Value. Comparison of methods used above.

1.1 Constructive Algorithms

In constructive algorithms the network structure is built during the training process by adding hidden layers, nodes and connections to a minimal neural network architecture.

1.2 Pruning Algorithms

In pruning algorithms the network structure is reduced by cutting down hidden layers, nodes and connections to a large neural network architecture resulting into a minimal structure.

SSE Method

The input and output values are normalized.

Number of neurons in the hidden layer to lie between $l < m < 2l$, where 'm' is the number of nodes in the hidden layer, 'l' is the number of nodes in the input layer.

The weights of the network, i.e., the weights from input to hidden neurons [V] and weights from hidden to output neurons [W] are initialized to a small random value between -1 and 1.

For the training data, one set of inputs and outputs is presented. The pattern is presented to the input layer 'I', as inputs to the input layer, then the outputs of the input layers are evaluated as,

$$O_I = I_I$$

Inputs to the hidden layer are computed by multiplying corresponding weights to the synapses as,

$$I_H = V^T * O_I$$

The units of the hidden layer evaluate the output using the sigmoidal function.

$$O_H=1/(1+e^{-IHi})$$

The inputs to output layer are calculated by multiplying the corresponding weights of the synapses as

$$Io = WT * OH$$

The output layer units evaluate the output using sigmoidal function as,

$$O_o=1/(1+e^{-Ioj}).$$

This is the network output.

The value of error and the difference between the network output and the desired output for a training set is calculated,

$$\text{Error} = (T_o - O_o)^2$$

2. Problem Formulation

SNRF estimation for one dimensional function approximation is given below.

Training data are uniformly spaced.

$$\mathbf{X \subset R_1}$$

The error signal,

$$ei = si + ni = si + \beta mi \qquad (i = 1,2,...N)$$

The energy signal,

Es+n = Es + En

Es+n = C(ei ,ei)

$= \Sigma\, \mathbf{e_i}^2$

High level of correlation between two neighboring samples of s. Thus,

C(si,si-1) ≈ C(si,si)

Due to the nature of WGN, noise of a sample is independent of noise on neighboring samples:

C(ni,ni-1) = C(βm, βm-1)
= 0

where, ni-1 represents the circularly *shifted replica of ni.*

SNRFe

$$\frac{Es}{En} = \frac{C(ei,ei-1)}{C(ei,ei) - C(ei,ei-1)}$$

In order to detect the existence of useful signal in *e, the* SNRF of *e* has to be compared with SNRF of WGN estimated using the same number of samples.

SNRFWGN

$$\frac{Es}{En} = \frac{C(ni,ni-1)}{C(ni,ni) - C(ni,ni-1)}$$

SNRFWGN is independent of noise level β.
The average value of SNRFWGN can be obtained. Its program is as follows:

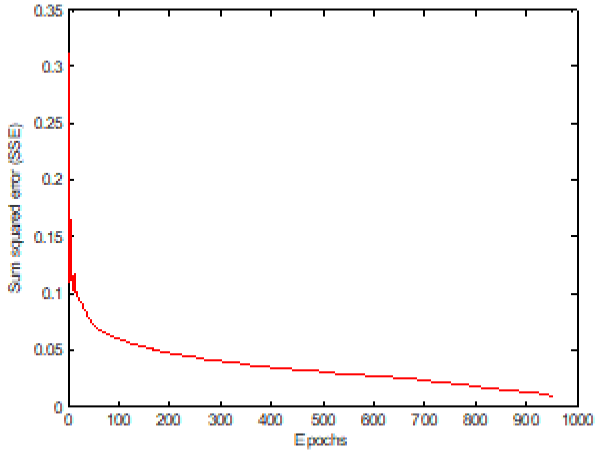
```
patterns = [0.2 0.8 0.006 0.747 0.0064 0.283; 0.6 0.4 0.0019 0.603 0.003 0.156; 0.9 0.1 0.0005 0.351 0.0018 0.041; 0.1 0.9 0.065 0.883 0.066 0.337];
desired_out = [ 0.814; 0.667; 0.358; 0.54];
sse_rec = [];
sse = 10;
eta = 0.6;
alpha = 0.8;
patterns = [patterns ones(size(patterns,1),1) ];
num_inp = size(patterns,2);
num_hid = 6;
num_out = size(desired_out,2);
w1 = 0.5*(1-2*rand(num_inp,num_hid-1));
w2 = 0.5*(1-2*rand(num_hid,num_out));
dw1_last = zeros(size(w1));
dw2_last = zeros(size(w2));
epoch = 0;
while sse > 0.01
winp_into_hid = patterns * w1;
hid_act = 1./(1+exp( - winp_into_hid));
hid_with_bias = [ hid_act ones(size(hid_act,1),1) ];
winp_into_out = hid_with_bias * w2;
out_act = 1./(1+exp( - winp_into_out));
output_error = desired_out - out_act;
sse = trace(output_error'*output_error);
sse_rec = [sse_rec sse];
deltas_out = output_error .* out_act .* (1-out_act);
deltas_hid = deltas_out*w2' .* hid_with_bias .* (1-hid_with_bias);
deltas_hid(:,size(deltas_hid,2)) = [];
dw1 = eta * patterns' * deltas_hid + alpha * dw1_last;
dw2 = eta * hid_with_bias' * deltas_out + alpha * dw2_last;
w1 = w1 + dw1; w2 = w2 + dw2;
dw1_last = dw1; dw2_last = dw2;
epoch = epoch + 1;
epoch < 5000;
 if rem(epoch,50)==0
 disp([' Epoch ' num2str(epoch) '  SSE ' num2str(sse)]);
 disp (['W1' mat2str(w1,3) 'W2' mat2str(w2,3)]);
end
end
figure(1);
plot(sse_rec,'r'); xlabel('Epochs'); ylabel('Sum squared error (SSE)');
```

3. Simulation and Testing

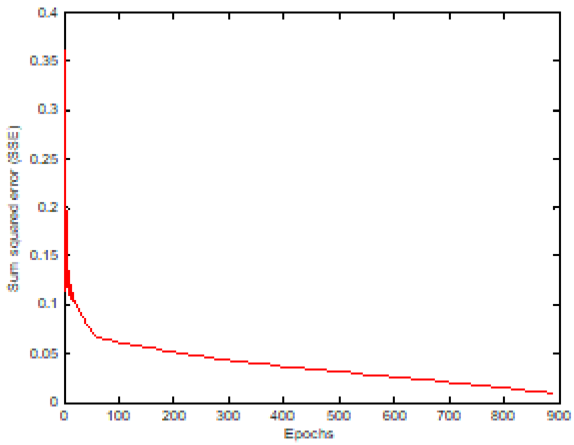
Simulation Data

Node1	Node 2	Node 3	Node 4	Node 5	Node 6	Output
0.2	0.8	0.006	0.747	0.0064	0.283	0.814
0.6	0.4	0.0019	0.603	0.003	0.156	0.667
0.9	0.1	0.0005	0.351	0.0018	0.041	0.358
0.1	0.9	0.065	0.883	0.066	0.337	0.54

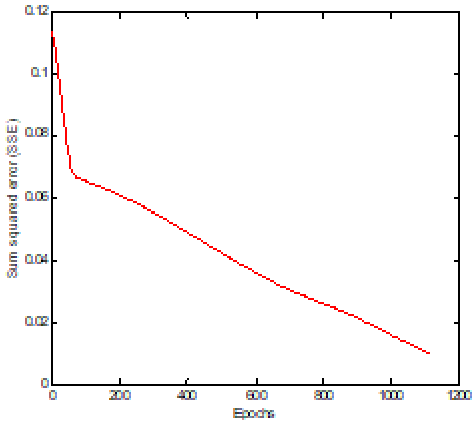
4. Results and Discussions



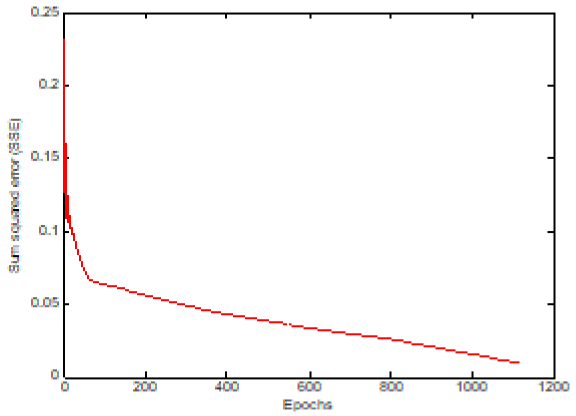
6 Hidden Units



7 Hidden Units



8 Hidden Units



9 Hidden Units

epochs No. of hid units	500	700
6	0.03769	0.027375
7	0.038782	0.031463
8	0.034565	0.027593
9	0.03341	0.023964
10	0.033518	0.024608
11	0.032286	0.019739
12	0.045251	0.028538

5. Conclusions

Values of sse computed for different hidden neurons suggest that the suitable number of hidden neurons here are 11 which gives lowest error. DV Method when applied to the network under consideration, suggests 11 neurons as the right number of neurons for the network for the given desired output.

References

[1] Ah Chung Tsoi, Markus Hagenbuchner and Alessio Micheli, "Building MLP Networks by Construction," *IEEE*, pp. 549-554, 2000.

[2] A. Lapedes and R. Farber, "How neural nets work," in Neural Information Processing Systems, 1988, pp. 442-456.

[3] Anthony N . Burkitt "Optimization of the Architecture of Feed-forward Neural Networks with Hidden Layers by Unit Elimination" University of Wuppertal, Complex Systems (1991) 371-380

[4] A. J. Owens and D. L. Filkin, "Efficient training of the back propagation network by solving a system of stiff ordinary differential equations,'' Proc. Of Int. Joint Conf Neural Networks, pp. I-381-II-386, June 1989.

[5] A. Waibel, T. Hanazawa, G. Hinton, K. Shikano, and K.J. Lang," Phoneme recognition using time-delay neural networks," IEEE Trans. Acoust., Speech, Signal Process., vol. 37, pp. 328-339, Mar. 1989.

[6] B. Choi, J. H. Lee, and D. H. Kim, "Solving local minima problem with large number of hidden nodes on two-layered feed-forward artificial neural networks," Neurocomputing, vol. 71, no. 16–18, pp. 3640–3643, 2008.

[7] B. Curry and P. H. Morgan, "Model selection in neural networks: some difficulties," European Journal of Operational Research, vol. 170, no. 2, pp. 567–577, 2006.

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

[9] C. A. Doukim, J. A. Dargham, and A. Chekima, "Finding the number of hidden neurons for an MLP neural network using coarse to fine search technique," Proceedings of the 10th International Conference on Information Sciences, Signal Processing and Their Applications (ISSPA '10), pp. 606–609,May 2010.

[10] D.E. Rumelhart, G.E. Hinton, and R.J. Williams."Parallel Distributed Processing", Cambridge MA MIT Press, vol. I, ch. 8, pp. 318-362, 1986

[11] D. Hunter, H. Yu, M. S. Pukish III, J. Kolbusz, and B. M. Wilamowski, "Selection of proper neural network sizes and architectures: a comparative study," IEEE Transactions on Industrial Informatics, vol. 8, no. 2, pp. 228–240, 2012.

[12] E. Barnard and D. Casasent, "Image processing for image understanding with neural nets," in Proc. Int. Joint Conf Neural Networks, pp. 1-111-1-116, June 1989.

[13] Etienne Barnard "Optimization for Training Neural Nets" IEEE transactions on neural networks, vol. 3, no. 2, March 1992.

[14] 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.

[15] G. B. Huang, "Learning capability and storage capacity of two-hidden- layer feedforward networks," IEEE Transactions on Neural Networks, vol. 14, no. 2, pp. 274–281, 2003.

[16] G. Strang. Introduction to Applied Mathematics. Cambridge, MA: Wellesley-Cambridge, 1986.

[17] G. W. Steward, Introduction to Matrix Computations. London: Academic Press, 1973.

[18] 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.

[19] H.C.Yuan, F. L. Xiong, and X.Y.Huai, "A method for estimating the number of hidden neurons in feed-forward neural networks.

[20] H. Robbins and S. Munroe, "A stochastic approximation method," Ann. Math. Statist., vol. 22, no. I, pp. 400-407, 1951.

[21] H. Bourlard and C. J. Wellekens, "Links between Markov models and multilayer perceptrons," in Advances in Neural Information Processing Systems 1, (D. S. Touretzky, Ed. Los Altos, CA: Morgan Kauffman, 1988, pp. 502-510.