Multilayer Feed-Forward Artificial Neural Network Integrated with Sensitivity Based Connection Pruning Method

Siddhaling Urolagin[1], Prema K.V.[2], JayaKrishna R.[1] and N.V. Subba Reddy[2]

[1] Dept. Comp Sc. & Engg., M.I.T., Manipal-576104. Karnataka, India
[2] Mody Institute of Technology and Science, Rajasthan, India
siddesh_u@yahoo.com,
{prema_kv,dr_nvsreddy}@rediffmail.com,
jayakrishnaa.r@gmail.com

Abstract. The Artificial Neural Network (ANN) with small size may not solve the problem while the network with large size will suffer from poor generalization. The pruning methods are approaches for finding appropriate size of the network by eliminating few parameters from the network. The sensitivity based pruning will determine sensitivity of the network error for removal of a parameter and eliminate parameters with least sensitivity. In this research a sensitivity based pruning method is integrated with multilayer feed-forward ANN and applied on MNIST handwritten numeral recognition. An analysis of effect of pruning on the network is compared with performance of a network without pruning. It is observed that the network integrated with pruning method show better generalization ability than a network without pruning method being incorporated.

Keywords: Sensitivity measure, pruning algorithm, generalization, global error.

1 Introduction

In theory, if a problem is solvable with a network of a given size, it can also be solved by a large net, which imbeds the smaller one, with all the redundant connections, or synapses, having zero strength. However, the learning algorithm will typically produce a different structure, with non vanishing synaptic weights spreading all over the net, thus obscuring the existence of a smaller size neural net solution. A network, which is too small, may never solve the problem, while larger network size may cause over fitting [1]. For a network to be able to generalize well, it should have fewer parameters than there are data points in training set [2], [3]. It has been observed that network with small size that fits the data well have good generalization capability [3]. Thus it makes sense to start with a large net and then reduce its size.

The pruning algorithms involves in removing network elements such as nodes, weights or biases selectively in order to reduce the size of the network. There are several pruning algorithms have been proposed in the literatures. The Optimal Brain

V.V. Das and J. Stephen (Eds.): CNC 2012, LNICST 108, pp. 68–74, 2012.
© Institute for Computer Sciences, Social Informatics and Telecommunications Engineering 2012

Damage [4] which estimates the saliency of connections and removes connections based on this estimation. The Bottom Up Freezing [5] is a method in which nodes are frozen from training if their contribution falls below a threshold. Nodes which are frozen are mostly removed from the network. A review on network pruning algorithms has been given in [6]. In this paper we integrated the sensitivity based pruning method of [7] with multilayer feed-forward neural network and applied it for handwritten numeral recognition. We analyze the behavior of the pruning integrated multilayer feed-forward neural network using experiments. During the training usual objective is to reduce the network error. The effect of pruning a connection on network error has been discussed in section 2. In section 3 the sensitivity based pruning method of [7] has been elaborated. In section 4 the feature extraction on handwritten numerals has been discussed. The experimental results are discussed in section 5. The section 6 covers the concluding remarks.

2 Effect of Pruning on Network Error

In a typical supervised learning method on presenting a pattern p let $t_{pi,}$ the desired output for unit i. The difference between the output produced O_{pi} and desired output t_{pi} is estimated usually as network error E. For a given training set, E is function of all w_{ij} weights. The learning is the process of modifying the weights such that the network error E will be decreased. The celebrated back-propagation learning algorithm which is variant of the steepest descent optimization method [8] updates the weights after each presentation of a subset of the training patterns. After the network undergoes sufficient training and network error E reaches its local minima, where all its weights are in the final state w_{ij}^{f}. By arbitrarily setting of a w_{ij} weight to zero, which is equivalent to elimination the synapse that goes from neuron j to neuron i will typically result in an increase of the error E. i.e., $E(w_{ij} = 0) > E(w_{ij} = w_{ij}^{f})$. So, efficient *pruning* means finding the subset of weights that, when set to zero, will lead to the smallest increase in E.

3 Sensitivity Based Pruning Method

Moze and Smolensky [9] have introduced the idea of estimating the sensitivity of the error function to the elimination of each unit. The sensitivity estimation s_{ij} for elimination of weight w_{ij} is

$$S_{ij} = E(w_{ij} = 0) - E(w_{ij} = w_{ij}^{f}) \quad (1)$$

Where w_{ij}^{f} is the (final) value of the connection upon the completion of the training phase, The approach as given in [7] for the pruning a connection is to estimate

sensitivity of error function to exclusion of each connection and remove connections having lower sensitivity. The sensitivity s_{ij}, defined in (1), can be rewritten as

$$S = -\frac{E(w^f) - E(0)}{w^f - 0} w^f \tag{2}$$

Where $w = w_{ij}$ and E is expressed as a function of w, assuming that all other weights are fixed at their final states, upon completion of learning. A typical learning process does not start with w = 0, but rather with some small randomly chosen initial value w^i. The error E as the function of weight w is depicted in Fig. 1a. In this figure, the training is begins at initial weight values w^i and as training progressing E of the network decreases. At w^f the network error reaches a local minimum. Since we do not know E(0), we will approximate the slope of E(w) (when moving from 0 to w^f) by the average slope measured between w^i and w^f, namely

$$S \approx -\frac{E(w^f) - E(w^i)}{w^f - w^i} w^f \tag{3}$$

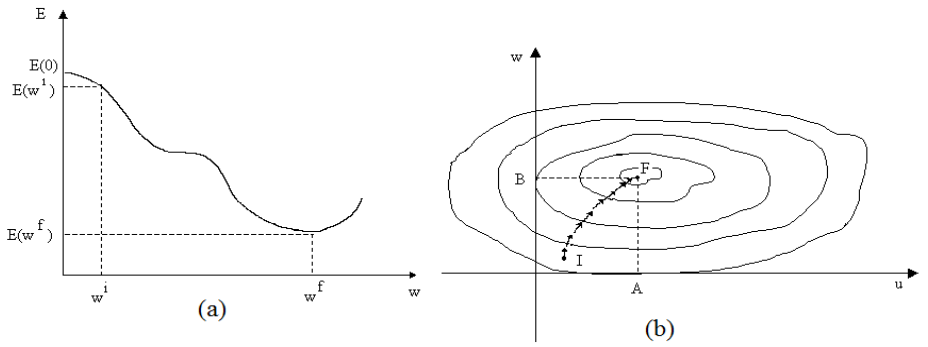

Fig. 1. (a) The error as a function of one weight, (b) Learning on an error function Surface

The initial w^i and final weights w^f,are quantities that are available during the training phase. However, for the numerator of (3) it was implicitly assumed that only one weight, namely w, had been changed, while all other weights remained fixed. This is not the case during normal learning. Consider for example a network with only two weights u and w. the numerator of (3) will be

$$E(u^f, w^f) - E(u^f, w^i) \tag{4}$$

The error E (u, w) is illustrated by the constant value contours in Fig. 1b. The initial point in the weight space is designated by I and the learning path is the dashed line from I to F, the final point. For a precise evaluation of S, the numerator of (2) can be evaluated as

$$E(w = w^f) - E(w = 0) = \int_A^F \frac{\partial E(u^f, w)}{\partial w} dw \tag{5}$$

The integral is along the line from point A, which corresponds to w = 0 while all other weights are in their final states, to the final weight state F. However, the training phase starts at point I rather than A, so we have to compromise on an approximation to integral above, namely, we will use

$$E(w = w^f) - E(w = 0) = \int_I^F \frac{\partial E(u, w)}{\partial w} dw \tag{6}$$

This expression will be further approximated by replacing the integral by summation, taken over the discrete steps that the network passes while learning. Thus the estimated sensitivity to the removal of connection w_{ij} will be evaluated as

$$\hat{S}_{ij} = -\sum_0^{N-1} \frac{\partial E}{\partial w_{ij}}(n)\, \Delta w_{ij}(n)\, w_{ij}^f \Big/ (w_{ij}^f - w_{ij}^i) \tag{7}$$

Where N is number of training epochs. The above estimate for the sensitivity uses terms which are readily available during the normal course of training. Obviously the weight increments w_{ij} are essence of every learning process, so they are always available. Also, virtually every optimization search uses gradients to find the direction of change, so the partial derivatives which are the components of gradient, are available. Therefore, the only extra computation demand for implementing is the summation of (7). For the special case of back-propagation, weights are updated according to [8], hence (7) reduces to

$$\hat{S}_{ij} = \sum_0^{N-1} [\Delta w_{ij}(n)]^2 \, w_{ij}^f \Big/ \eta (w_{ij}^f - w_{ij}^i) \tag{8}$$

Upon completion of training we are equipped with a list of sensitivity numbers, one per each connection. They were created by a process that runs concurrently, but without interfering, with the learning process. At this point a decision can be taken on pruning the synapses of the smallest sensitivity based on some criterion.

4 Experiment and Results

An experiment has been conducted multilayer feed-forward ANN integrated with sensitivity based connection pruning method of [7] for MNIST handwritten numerals recognition. 3000 samples for training set and 1500 for testing set is considered from large collection of handwritten digits.

The normalized numeral image is convolved with these filters to produce responses. For a specific θ_i, three responses at different fundamental frequencies are formed. From this averaged Gabor response we extract 32 directional features using a grid structure along horizontal, vertical, secondary diagonal, primary diagonal direction as given in [10].

For the experiment the neural network architecture selected as 32 input nodes, a hidden layer with 35 nodes and the output layer with 10 nodes in it. The integrated

neural network is trained for 5000 epochs with learning rate as 0.25. After 1000 epoch the sensitivity based pruning is carried out with different pruning thresholds. The parameters with sensitivity measures less than a chosen threshold are removed. The following Table 1 shows number of connections pruned at different epochs with pruning threshold as 0.0001, 0.0005 and 0.001.

Table 1. Total number of connections pruned during training phase

	Pruning Thresholds		
Epoch	T=0.0001	T=0.0005	T=0.001
1000	82	82	82
2000	16	21	27
3000	19	24	28
4000	13	56	61
5000	13	20	39

The analysis on the effect of connection pruning on a network as it undergoes training is carried out. This analysis is done for pruning the network with threshold 0.0005. At each 1000 epoch, MSE and classification rate before pruning and after pruning the connections on training set and testing set are obtained, which are tabulated in Table 2. At the end of 1000 epoch a classification rate of 96.83% and MSE of 0.023575 is observed on training set. When pruning is carried out and 82 connections with least sensitivity measure are removed a slight increase in MSE as 0.02392 is observed. On testing set at the end of 1000 epoch, a classification rate of 77.52% and MSE of 0.140259 are observed. Due to pruning the connections, an improvement in classification rate as 78.44% and MSE as 0.138445 is observed. On the other hand it is interesting to observe that when the network undergoes sufficient training, say at epoch 4000, the pruning has inverse effect and causing decrement in classification rate and increment in MSE, which can be observed in 8^{th} and 9^{th} rows of the Table 2. However as the training progresses, the network shows learning ability, which are indicated as decrease in MSE during further epochs.

A comparison of MSE and classification rate for network with different pruning thresholds and a network without pruning is carried out. The comparison results on training set are summarized in Table 3. The results for testing set are given in Table 4. With optimum number of connections being pruned, the improvement in MSE and classification rate is observed. At the end of 5000 epochs when no pruning is employed on the network a MSE of 0.142329 and classification rate of 77.06% are observed for testing set as shown in Table 4. At the end of 5000 epochs, with threshold as 0.0001 MSE of 0.141965 and classification rate of 77.98% and with threshold as 0.0005 MSE of 0.132123 and classification rate of 80.28% are observed as shown in Table 4. These results are better as compared to no pruning being used on the network.

Table 2. Effect of pruning on network during training phase

		On Training Set		On Testing Set	
Epoch	**Observation**	**MSE**	**Cls. Rate in%**	**MSE**	**Cls. Rate in %**
1000	Before Pruning	0.023575	96.83	0.140259	77.52
1000	After Pruning	0.02392	96.83	0.138445	78.44
2000	Before Pruning	0.02525	96.83	0.152521	76.61
2000	After Pruning	0.024942	96.83	0.160962	75.69
3000	Before Pruning	0.027919	96.83	0.143832	77.52
3000	After Pruning	0.030434	96.24	0.148899	77.06
4000	Before Pruning	0.02823	96.83	0.141866	77.52
4000	After Pruning	0.100993	88.71	0.18957	70.18
5000	Before Pruning	0.030268	96.83	0.138166	77.06
5000	After Pruning	0.030233	96.83	0.132123	80.28

Table 3. Results on training set

	No Pruning		**T=0.0001**		**T=0.0005**		**T=0.001**	
Epoch	**MSE**	**Cls. Rate in %**	**MSE**	**Cls. Rate in %**	**MSE**	**Cls. Rate in %**	**MSE**	**Cls. Rate in %**
1000	0.023575	96.83	0.02392	96.83	0.02392	96.83	0.02392	96.83
2000	0.025217	96.83	0.024746	96.83	0.024942	96.83	0.026581	96.83
3000	0.027924	96.83	0.028157	96.83	0.030434	96.24	0.037029	94.65
4000	0.028247	96.83	0.031294	96.24	0.100993	88.71	0.75471	51.88
5000	0.028439	96.83	0.028503	96.63	0.030233	96.83	0.026036	97.03

Table 4. Results on testing set

	No Pruning		**T=0.0001**		**T=0.0005**		**T=0.001**	
Epoch	**MSE**	**Cls. Rate in %**	**MSE**	**Cls. Rate in %**	**MSE**	**Cls. Rate in %**	**MSE**	**Cls. Rate in %**
1000	0.140259	77.52	0.138445	78.44	0.138445	78.44	0.138445	78.44
2000	0.151478	76.61	0.156431	75.69	0.160962	75.69	0.163175	75.69
3000	0.142597	76.61	0.144422	77.52	0.148899	77.06	0.148878	75.69
4000	0.142435	77.06	0.153685	76.61	0.18957	70.18	0.71225	41.28
5000	0.142329	77.06	0.141965	77.98	0.132123	80.28	0.137117	76.61

5 Conclusion

The pruning algorithms are approaches for finding appropriate size of the network by removing redundant parameters from the network. In the sensitivity based approach, the sensitivity of the network error due to elimination of each unit is estimated. Then several parameters with least sensitivity are removed from the network. In this research a sensitivity based pruning method of [7] is integrated with multilayer feed-forward ANN and effect of pruning is analyzed. The experiments have been conducted on integrated ANN to recognize MNIST handwritten numerals. It is interesting to observe that during the initial phase of learning more parameters are pruned than later stage. During the training of ANN, pruning the parameters will lead to increase in network error (i.e. MSE). However as the training progresses, then network shows ability learn even with fewer parameters in it. On unseen (test) data, it is usually observed that whenever pruning takes place it leads to improve in its generalization ability, which is evident from decrease in MSE and increase in classification rate on test data. When compared with a network with same topology and without pruning being integrated, the ANN integrated with pruning algorithm show better generalization results.

References

1. Steve, L., Lee Giles, C.: Overfitting and Neural Networks: Conjugate Gradient and Backpropagation. In: International Joint Conference on Neural Networks, pp. 114–119. IEEE Computer Society, Los Alamitos (2000)
2. Baum, E.B., Haussler, D.: What size net gives valid generalization? Neural Comuta. 1, 151–160 (1989)
3. Denker, J., Schwartz, D., Wittner, B., Solla, S., Howard, R., Jackel, L., Hopfield, J.: Large automatic learning, rule extraction, and generalization. Complex Syst. 1, 877–922 (1987)
4. Le Cun, Y., Denker, J.S., Solla, S.A.: Optima Brain Damage. In: Touretzky, D.S. (ed.) Advances in Neural Information Processing (2) (Denver 1989), pp. 598–605 (1990)
5. Farzan, A., Ghorbani, A.A.: The Bottom-Up Freezing: An Approach to Neural Engineering. In: Proceedings of Advances in Artificial Intelligence: 14th Biennial Conference of the CAIAC, Ottawa, Canada, pp. 317–324 (2001)
6. Reed, R.: Pruning Algorithms-A Survey. IEEE Trans. on Neural Network 4(5), 740–747 (1993)
7. Karnin, E.D.: A Simple Procedure for Pruning Back-Propagation Trained Neural Networks. IEEE Transaction on Neural Network 1(2), 239 (1990)
8. Luenberger, D.G.: Linear and Nonlinear Programming. Addison-Wesley, Reading (1984)
9. Mozer, M.C., Smolensky, P.: Skeletonization: a technique for trimming the fat from a network via relevance assessment. In: Advances in Neural Information Processing Systems 1, pp. 107–115. Morgan Kaufmann (1988)
10. Urolagin, S., Prema, K.V., Subba Reddy, N.V.: Illumination Invariant Character Recognition using Binarized Gabor Features. In: IEEE International Conference on CIMA, India, December 13-15, pp. 216–220 (2007)