A Sensitivity Analysis Algorithm for Pruning Feedforward Neural Networks

AP Engelbrecht I Cloete

Computer Science Department, Stellenbosch University
Stellenbosch, 7600, South Africa

ian@cs.sun.ac.za

Abstract

A pruning algorithm, based on sensitivity analysis, is presented in this paper. We show that the sensitivity analysis technique efficiently prunes both input and hidden layers. Results of the application of the pruning algorithm to various N-bit parity problems agree with well-known published results.

1 Introduction

The determination of the optimal neural network architecture to solve a particular problem is a difficult task. Several research papers have presented bounds on the number of hidden units necessary to solve a particular problem [Cosnard 1992, Kamruzzaman 1992]. However, these results are based on unrealistic assumptions about the network and the problem to be solved. These bounds therefore do not predict the correct number of hidden units for a general class of problems.

Usually, a rather time-consuming method was used to overcome the model selection problem: several different network architectures were trained, and the architecture with best performance were selected. Recently, more efficient network construction and pruning algorithms were proposed to combat the model selection problem. Network construction algorithms start with a small number of hidden units and add new units, or split existing units if the performance of the network is not satisfactory [Lee 1991, Wynne-Jones 1992]. Conversely, the main approach of pruning algorithms is to train a network that is larger than necessary and then to remove redundant elements [Hagiwara 1993, Reed 1993, Weigend 1993]. In this paper we consider network pruning, specifically pruning of feedforward neural network architectures.

Pruning algorithms may elect to remove weights and/or units. We present a sensitivity analysis pruning algorithm to prune the input and hidden layers of a trained feedforward neural network. This approach of pruning removes units that have the least statistical influence on all units in the succeeding layers.

This paper focuses on the pruning of hidden layers. Results of the algorithm in the elimination of irrelevant input units have been reported elsewhere [Cloete 1994, Engelbrecht 1995, Zurada 1994]. A mathematical formulation of the sensitivity analysis model is presented in Section 2. We show in Section 3 that the sensitivity analysis algorithm correctly prunes the hidden layer for six N-bit parity problems. The sensitivity analysis pruning algorithm does not require the computation of additional network parameters. All the variables needed by the pruning algorithm is available through the training algorithm. The pruning algorithm can only be applied on well-trained neural networks.

2 Sensitivity analysis

Consider a neural network which consists of the $K+1$ layers $L_0, L_1, \ldots, L_K$, where L_0 and L_K respectively represent the input and output layers. Let the weight matrix between layer L_k and L_{k+1} be denoted by W_k. Without loss of generality, assume that layer L_k is to be pruned with respect to layer L_l, where $l > k$. The sensitivity matrix $S_{lk}^{(p)}$ defines the sensitivity of the units in layer L_l to architectural changes in layer L_k for a specific training pattern p [Zurada 1994]:

$$\begin{aligned} S_{lk}^{(p)} &= O_l^{'} W_{l-1} O_{l-1}^{'} W_{l-2} \cdots O_{k+1}^{'} W_k \\ &= \prod_{m=0}^{l-1} O_{m+1}^{'} W_m \qquad (1) \end{aligned}$$

where $O_m^{'}$ is the diagonal matrix

$$O_m^{'} \doteq diag(o_{1m}^{'}, o_{2m}^{'}, \ldots, o_{Nm}^{'}) \qquad (2)$$

with $o_{nm}^{'}$ the derivative of the output of unit n in layer L_m. The product $O_{m+1}^{'} W_m$ refers to matrix manipulation. In the case of output-input layer sensitivity analysis, the dimension of the resulting sensitivity matrix $S_{K0}^{(p)}$ for pattern p will be (*number of outputs*$\times$*number of inputs*). An element of the matrix $S_{lk}^{(p)}$ is denoted by $S_{ij,lk}^{(p)}$ where from [Zurada 1994]

$$S_{ij,lk}^{(p)} = \frac{\partial O_{i,l}}{\partial O_{j,k}} \qquad (3)$$

and measures the sensitivity of unit k in layer L_j to small perturbations in unit i of layer L_l. A double subscript is used in equation (3) where the first part ij refers to a specific element of matrix $S_{lk}^{(p)}$, and the second part lk refers to the corresponding layers.

Since each training pair produces a different sensitivity matrix, a measure should be defined to calculate the sensitivity of layer L_l to changes in layer L_k over the entire training set. For the purposes of this exposition, we have used the *mean square average* sensitivity matrix A_{lk} for layer L_k with respect to layer L_l

$$A_{ij,lk} = \sqrt{\frac{\sum_{p=1}^{P} [S_{ij,lk}^{(p)}]^2}{P}} \qquad (4)$$

where $A_{ij,lk}$ is the mean square average sensitivity of unit j in layer L_k to unit i in layer L_l; P is the total number of training patterns.

Define the significance of unit $L_{j,k}$ over all units $L_{i,l}$ as

$$\Phi_{L_{j,k}} = \max_i \{A_{ij,lk}\} \qquad (5)$$

The pruning algorithm finds a large enough gap between the significance of two consecutive units in layer L_k. That is, if

$$\Phi_{L_{j+1,k}}/\Phi_{L_{j,k}} < constant \tag{6}$$

all units $L_{j+1,k}, \cdots, L_{Nk}$ are pruned. Equation (6) requires the significance vector $\vec{\Phi}$ to be sorted in descending order such that

$$\Phi_{L_{j,k}} \geq \Phi_{L_{j+1,k}}$$

The calculation of sensitivity measures in equation (1) does not require additional network parameters to be computed during training. The pruning algorithm therefore does not increase the complexity of the training algorithm. Since the pruning algorithm is a post-processing technique, it does not interfere with the training process. In the next section we illustrate the sensitivity analysis technique on various N-bit parity problems.

3 Experimental results

In this section we apply sensitivity analysis to six N-bit parity problems, with $N = 2, 3, \cdots, 7$. The main objective is to illustrate the efficiency of the sensitivity analysis algorithm for pruning the hidden layer of feedforward neural networks. We show that the results obtained from the pruning algorithm are similar to the results published by Rumelhart and McClelland [Rumelhart 1986]. For all the experiments in this section, we have used a three layer feedforward neural network which was trained using the standard back-propagation learning algorithm [Zurada 1992]. Training was stopped when all the examples in the training set were classified correctly. Both the hidden and output units had sigmoid activation functions. Weights were initialized to small random values in the range $[-\frac{1}{\sqrt{fanin}}, \frac{1}{\sqrt{fanin}}]$. The learning rate and momentum were taken as 0.1 and 0.9 respectively.

For each parity experiment, an initial data set of 320 input-target pairs were generated to create complete training and test sets. Random noise in the range $[0, 0.1]$ was added to each input and target value. These initial data sets were divided into training sets of 224 patterns and test sets of 96 patterns. A neural network with one hidden layer, consisting of 10 units, has been used for each experiment. For pruning purposes, the gap constant in equation (6) has been set to 0.5 (suggested value only).

Figure 1 depicts the sensitivity profiles for the output-hidden layer analysis of each parity problem. For each problem a clear distinction between significant and non-significant units is formed. At the time of convergence, a large enough gap exists between the group of significant units and the group of insignificant units to allow the pruning algorithm to eliminate $10 - N$ hidden units in the case of N-bit parity. For example, 6 hidden units are pruned when the sensitivity analysis algorithm is applied to the 4-bit parity problem (refer to Figure 1(c)). The results illustrated by Figure 1 agree with the statement by Rumelhart and McClelland that an N-bit parity problem requires at least N units to be solved [Rumelhart 1986].

Figure 1 also illustrates that premature pruning of hidden units will result in an insufficient small network architecture if 100% correct classification is required. Table 1 illustrates the effects of premature pruning for the 4-bit, 5-bit and 6-bit parity problems. For example, in Figure 1(c) a large gap forms between the three highly significant hidden units and the seven insignificant units at epoch 400. Pruning of the seven insignificant units at this time causes a

Parity problem	*Epoch*	*Training error*	*Test error*	*Hidden units*
4-bit	400	6.3%	6.3%	3
5-bit	800	8.5%	9.4%	4
6-bit	1000	2.7%	9.4%	5

Table 1: Effects of premature pruning

classification error[1] of 6.3% on both the training and test sets. For 100% correct classification, four hidden units are required as illustrated at epoch 1100. Similarly, premature pruning at epoch 800 for the 5-bit parity problem causes a 8.5% classification error on the training set and a 9.4% error on the test set. For real-world applications where the data sets usually contain incomplete data and outliers, the neural network will not be trained for 100% correct classification, since this will cause overfitting of the training set and poor generalization. In such cases pruning should be applied at a point where the network has been trained sufficiently and well before overfitting can occur. Further study is needed to determine when a network is sufficiently trained to allow pruning to be applied successfully.

4 Conclusion

We have presented a sensitivity analysis pruning algorithm that can be applied to the input and hidden layers of a feedforward neural network. A general formulation for the pruning of any layer (the output layer excluded) has been derived in equation (1). The results presented in the paper indicate that sensitivity analysis correctly prunes all hidden units which have little influence on the performance of the network. For the N-bit parity problems presented in this paper, pruning has resulted in a neural network architecture similar to that proposed by Rumelhart and McClelland [Rumelhart 1986].

Further investigation is needed to define the optimal gap size, and to investigate pruning during training. The performance of the sensitivity analysis pruning algorithm on real-world problems also need to be investigated.

References

[Cloete 1994] I Cloete and AP Engelbrecht, *New Tools for Decision Support*, AMSE International Conference on Intelligent Systems, Pretoria, September 1994.

[Cosnard 1992] M Cosnard, P Koiran and H Paugam-Moisy, *Complexity Issues in Neural Network Computations*, LATIN'92, Sao Paulo, Brazil, 1992.

[Engelbrecht 1995] AP Engelbrecht, I Cloete and JM Zurada, *Determining the Significance of Input Parameters using Sensitivity Analysis*, in J Mira, F Sandoval (eds), 'From Natural to Artificial Neural Computation', Proceedings of the International Workshop on Artificial Neural Networks, Torremolinos, Spain, June 7–9, 1995, in the series Lecture Notes in Computer Science, Springer-Verlag, Vol 930, pp 382–388.

[1]An example was correctly classified if the output of the network was greater than or equal to 0.7 (taken as 1), or if the network output was less than or equal to 0.3 (taken as 0).

[Hagiwara 1993] M Hagiwara, *Removal of Hidden Units and Weights for Back Propagation Networks*, Proceedings of 1991 IJCNN, Vol. 1, pp 351–354, 1993.

[Kamruzzaman 1992] J Kamruzzaman, Y Kumagai and H Hikita, *Study on Minimal Net Size, Convergence Behavior and Generalization Ability of Heterogeneous Backpropagation Network*, Artificial Neural Networks, Vol. 2, pp 203–206, 1992.

[Lee 1991] T-C Lee, *Structure Level Adaptation for Artificial Neural Networks*, Kluwer Academic Publishers, 1991.

[Reed 1993] R Reed, *Pruning Algorithms – A Survey*, IEEE Transactions on Neural Networks, Vol. 4, No. 5, pp 740–747, 1993.

[Rumelhart 1986] DE Rumelhart and JL McClelland, *Parallel Distributed Processing*, Vol. 1, 1986, MIT Press.

[Weigend 1993] AS Weigend, DE Rumelhart and BA Huberman, *Generalization by Weight-Elimination Applied to Currency Exchange Rate Prediction*, Proceedings of 1991 IJCNN, Vol. 1, pp 837–841, 1991.

[Wynne-Jones 1992] M Wynne-Jones, *Node Splitting: A Constructive Algorithm for Feedforward Neural Networks*, Neural Information Processing Systems, Vol. 4, pp 1072 – 1079, 1992.

[Zurada 1992] JM Zurada, *Introduction to Artificial Neural Systems*, West Publishing Company, 1992.

[Zurada 1994] JM Zurada, A Malinowski and I Cloete, *Sensitivity Analysis for Minimization of Input Data Dimension for Feedforward Neural Network*, IEEE International Symposium on Circuits and Systems, London, May 30 – June 3, 1994.

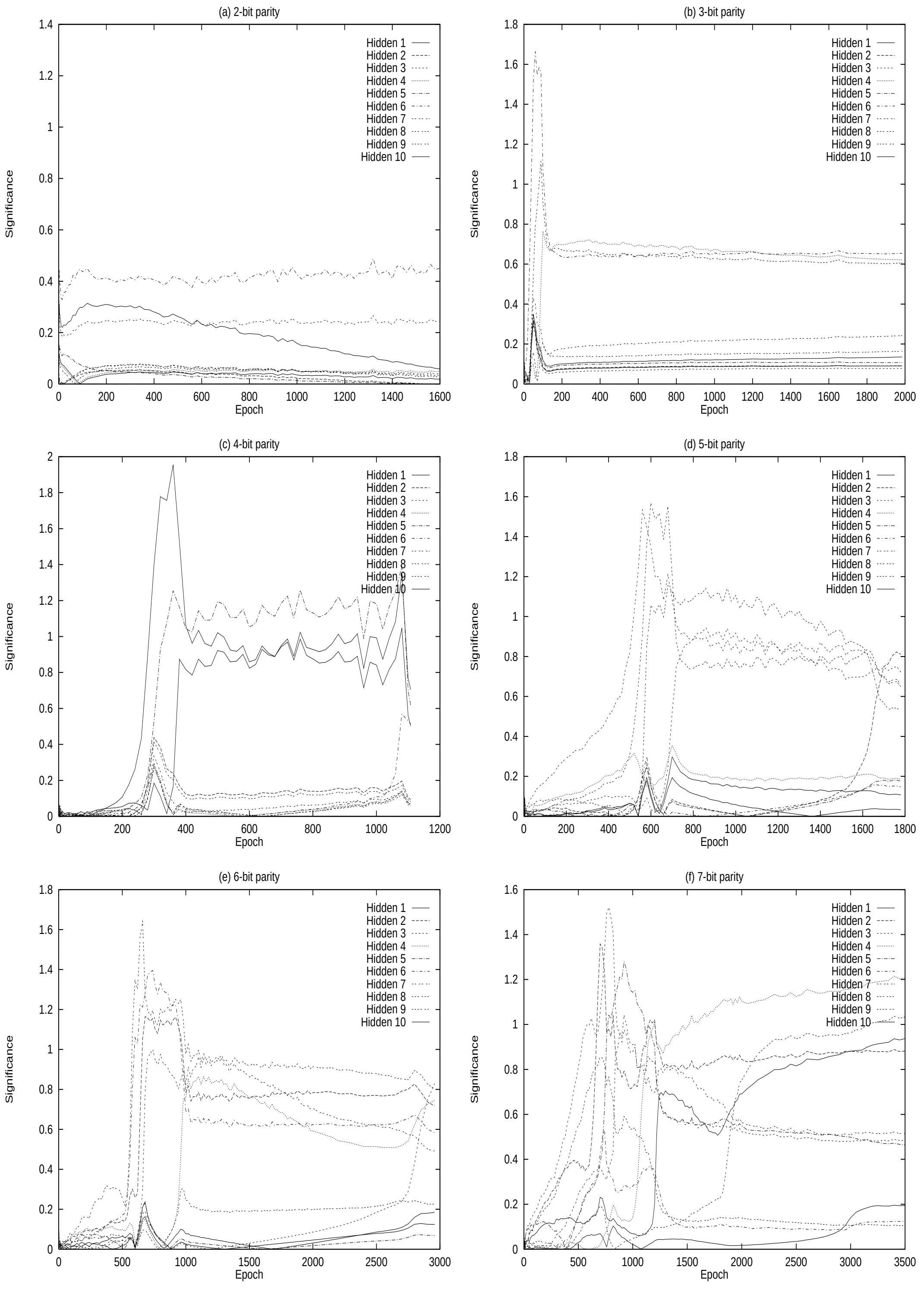

Figure 1: Output-hidden layer sensitivity profiles