Математички Билтен
23 (XLIX)
1999 (69–80)
Скопје, Македонија

ISSN 0351–336X

PRUNING ALGORITHM IN A DMA MODEL OF NEURAL NETWORKS

Lena Trajkovska

Abstract

Conventional network pruning techniques are uniquely established for multylayer neural networks in order to improve generalization or to prune processing units. The research presented in this paper is dealing with the modification of the architecture of an auto-associative neural network performed by progressive adaptation of the external stimuli. The proposed network pruning algorithm was obtained after an exhaustive experimenting with the Distributed Memory and Amnesia DMA model. The results of the experiments performed over a network with growing and constant architecture are presented. Finally, the dynamic shrink of the network by discarding the neurons one by one or simultaneously is compared.

1. Introduction

There are many ways for pruning neural networks by applying pruning functions during training process in order to eliminate the cells with minimum influence over network knowledge. The main goal of the research presented in this paper is to find a set of training patterns that will force the network to discard its own cells in the process of learning.

A cell is considered to be insignificant if its connections with all other cells are very week, i.e. the cell has no influence on other cells. In order to improve network performance this cell can be discarded, without complete lose of its influence during retraining. Variety of experiments with different training patterns were done over an augmented auto-associative network (Zdravkova, 1998) in order to discard all obsolete cells, starting from the last added cell. These experiments led to pruning algorithm presented in this paper.

The additional experiments for testing proposed algorithm are also included in this paper. They consisted of pruning the augmented network by choosing a different starting unit, pruning the augmented network for seven units at once and pruning a network with fixed architecture. The results were satisfactory and encouraging for future research in the area of modification of neural network architecture.

2. Preliminaries

The auto-associatiors are networks with an input layer of source nodes that project onto a hidden layer with connections of constant excitatory strength. All cells in the hidden layer are interconnected. The cells from the input layer are used only as an input port. Before the training starts, neutral values of the input activation are assigned to the initial state vector $a_i(0)$. The activation $a(t)$ in a discrete time t is changed according to the relation's (1) and (2):

$$\Delta a_i(t+1) = \begin{cases} E^*\mathrm{net}_i^*(1 - a_i(t)) - D^*a_i(t)\,, \mathrm{net}_i > 0 \\ E^*\mathrm{net}_i^*(a_i(t)+1) - D^*a_i(t)\,, \mathrm{net}_i \geq 0 \end{cases} \tag{1}$$

$$a_i(t+1) = a_i(t) + \Delta_i(t+1) \tag{2}$$

where net_i is the total net input to the processing unit i and E and D are parameters equal to 0.15, where D is the decay parameter and E is the scaling parameter.

The weights are modified by error-correcting rule adapted for auto-associative models to:

$$\Delta w_{ij} = \mathrm{lrate}^*\mathrm{err}_i^* a_j \quad i,j = 1,\dots,n\,. \tag{3}$$

The error for each unit is defined to be

$$\mathrm{err}_i = \mathrm{exinput}_i - \mathrm{input}_i \quad i = 1,\dots,n \tag{4}$$

where $input_i$ is value of the internal input based on the activations at the end of each preceding iteration (5).

$$\mathrm{input}_i = \sum_{k=1}^{n} w_{ik}^*\mathrm{act}_k \quad i = 1,\dots,n\,. \tag{5}$$

The error of recognizing the p-th pattern is computed by the following formula

$$\mathrm{SSE}_p = \sum_{j=1}^{n}(O'_{pj} - O_{pj})^2 \quad p = 1,\dots,n_p \tag{6}$$

where O'_{pj} is the value of the j-th cell from the world layer in the p-th pattern and O_{pj} is the value of the j-th cell from the learning layer in the p-th pattern. The average pattern error is calculated according to:

$$\text{Error} = \frac{\sum_{p=1}^{n_p} SSE_p}{n_p} \tag{7}$$

where n_p is the number of the training patterns.

The advantage of this model is that the activations are bounded in the segment $[-1, 1]$ which prevents them from infinite growth, but that is accomplished with equalizing the change force of the net input and the restoration force of the decay term D, so its stable state its not in the corners of the hypercube as in the BSB model.

The only objective for modification of the existing network architecture was the presence of neurons that have connection weights with the others with strength as much near zero as possible.

The main principle was accepted from Hebbian learning rule:

* When two units are correlated, which means that they are inhibited or excited at the same time then their interconnection should be increased, if they are with opposite activation then the connection should be decreased. When two units are not correlated, while one of them is idle and the other is excited or inhibited then the connection between them will be almost neutral.

This means that in order to allow redirection of the influence of the obsolete processing unit to the others and to neutralize its connections, the network has to be retrained while the unit is correlated with the others. After several training experiments it was concluded that if a neuron was totally inhibited, the absolute values of connection weights with other units were around 10^{-4}. So, the first conclusion was that in order to discard the last added neuron, the network should be trained with training patterns containing inhibitory activation for pruned processing unit and neutral activations for all other units, i.e. that all units should be left neutral and the last one should be constantly inhibited.

The next step was defining appropriate way of discarding next obsolete unit, bearing in mind that one has already been discarded. After several experiments it was concluded that the best pruning of two consecutive processing units, when one has already been discarded, was accomplished by training the network with total inhibition of the next neuron and neutralizing the others. This process was repeated for several units. It was concluded that the pruning could be done until one of the connections of the pruned cell has an absolute weight greater than or equal to 0.2.

The pruning was considered completed in the discrete moment t when connection weights of the i-th unit

$$C_i^{(t)} = \sum_{j=1}^{n} \mathrm{abs}\big(w_{ij}(t)\big) \quad \text{and} \quad R_i^{(t)} = \sum_{k=1}^{n} \mathrm{abs}\big(w_{ki}(t)\big) \tag{8}$$

started increasing:

$$C_i^{(t+1)} \geq C_i^{(t)} \quad \text{or} \quad R_i^{(t+1)} \geq R_i^{(t)} . \tag{9}$$

For pruning the network with fixed architecture, the criterion of the algorithm was relaxed, and the relation (10) was used:

$$C_i^{(t+1)} \geq C_i^{(t)} \quad \text{or} \quad R_i^{(t+1)} \geq R_i^{(t)} , \quad t > 0 . \tag{10}$$

The criteria is used only after the second epoch of pruning, because after the first epoch $C_i^{(1)} < C_i^{(0)}$ but $R_i^{(1)} \geq R_i^{(0)}$, but in the following epochs C_i and R_i, are in the same suite, i.e. they are both decreasing until one of them increased.

After successful pruning of one obsolete unit, the process can be repeated with other neurons satisfying the same condition. The next cell was chosen to be pruned when it had the strongest connection in absolute value with the last pruned unit is the next to be pruned. Finally, the criterion for ending of this iterative process was defined to be the moment when the last pruned unit had stronger absolute connection weights then those from the starting net.

```
$i = n + 1$
repeat
   $i = i - 1$
   repeat
      remember the network in N
      remember the weights of the connections of the unit $i$ in matrix A
      inhibit the unit $i$
      compare the weights of the connections of the unit i and the matrix
   A
   until (the connections of i have greater or equal weights with A   (*)
   the learning network is N
until (at least one of the connections of the unit $i$ have absolute
weights greater then or equal to 0.2)
refresh the values of the network
```

Algorithm 1. Pruning algorithm

```
remember the $\max_{i,j}(\mathrm{abs}(w_{ij}))$ in M
$i = n$
repeat
   $k = 0$
   if ($i <> n$) then the new value for i is the unit which
     has the strongest absolute weight with the $i$-th unit
     repeat
       $k = k + 1$
       remember the network in N
       remember the weights of the connections of the unit $i$ in matrix A
       inhibit the unit $i$
       compare the weights of the connections of the unit i and the matrix
   A
 until (the connections of i have greater or equal weights with A) and $k>1$   (*)
   the learning network is N
until (at least one of the connections of the unit $i$ have absolute
weights greater then or equal to M)
refresh the values of the network
```

Algorithm 2. Relaxed pruning algorithm

3. Experimental results

The network consisting of 18 processing units was augmented until it reached 30 processing units. The growing was performed in a DMA model with activation function (2) and the delta learning rule (3), following the criteria defined in (Zdravkova, 1998). The initial network was trained within 20000 epochs. Retraining series for each new unit passed through 5000 epochs, reaching final error of 2.414. The weights of the network lied in the interval $w_{kj} \in (-0.87887,\ 1.33241)$, with the weakest connection $13- > 3$ and the strongest $16- > 17$. In order to prune the last unit, this network was retrained by inhibiting the 30-th unit, and neutralizing the others during 10000 epochs (Table 1).

Before the pruning	
$w_{30j} \in (-0.13703,\ 0.11538)$	$R_{30} = 1.74517$
$w_{k30} \in (0.00000,\ 0.00000)$	$C_{30} = 0.00000$
After the pruning	
$w_{30j} \in (-0.05431,\ 0.05451)$	$R_{30} = 0.81633$
$w_{k30j} \in (-0.00003,\ 0.00003)$	$C_{30} = 0.00027$

Table 1. Connection weights with 30-th unit

Connection weights corresponding to the pruned unit were distributed to several others (Table 2). They were slightly increased for several units that accepted the role of the pruned unit, and decreased mainly in the area of recently added units. These units have started to become more mutually correlated.

From unit	Before the pruning	After the pruning	To unit	Difference
7	−0.102780	−0.037380	1	−0.065400
8	−0.306280	−0.215960	1	−0.090320
19	0.000000	0.104500	3	−0.104500
20	0.000000	0.102400	3	−0.102400
21	0.000000	0.115460	10	−0.115460
26	0.000000	0.118350	3	−0.118350
26	0.000000	0.134350	5	−0.134350
5	−0.121630	−0.072910	29	−0.048720
5	0.691110	0.576230	23	0.114880
22	0.637090	0.508300	23	0.128790
23	0.538470	0.406850	22	0.131620
22	0.621770	0.451170	24	0.170600

Table 2. The most affected units after the pruning of the 30-th unit

During the experiments for obtaining the best training patterns for pruning the 29-th unit, the best results were gained with total inhibition of the 29-th cell and neutralization of all the others. This was the moment when the algorithm (8) was established, except for the stop criteria, which was reached after inhibiting the 23-th processing unit. The connection from 22-th unit to 23-th unit had weight of 0.78, which could not be diminished even after 30000 epochs of training. That's why the pruning had to stop.

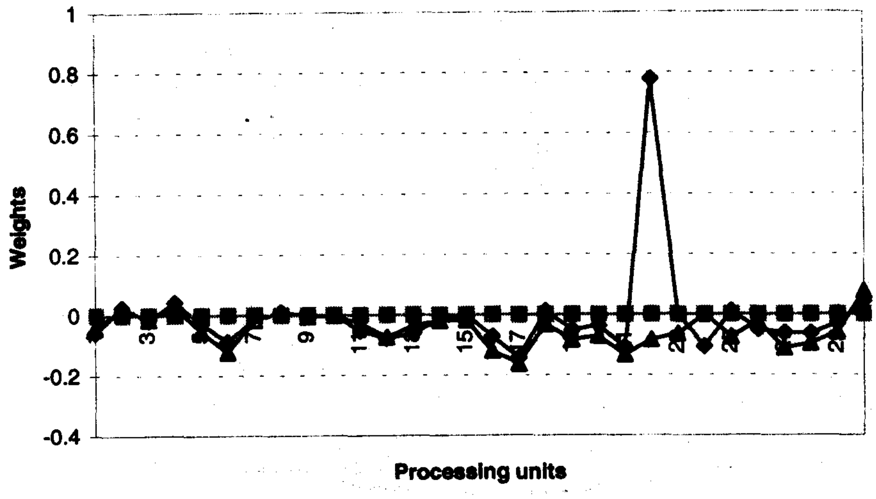

Figure 1: Weights of the connections of the 23-th and the 24-th unit after their pruning

The network was pruned for seven processing units following this algorithm, after 100000 epochs of training. As a last step in the algorithm, training the network with patterns containing the first 23 activation with which the network was grown made a renewing of the first 23 values. The inhibited units were idle.

The weights of all the pruned units varied in the interval $w_{kj} \in (-0.16278,\ 0.08135)$ while before the pruning they were on the interval $w_{kj} \in (-0.25316,\ 0.3725)$. Table 3. contains the number of epochs needed for pruning of the corresponding unit, the Error and C_i and R_i (9). The arguments B and A were introduced to represent the values of R_1 and C_1 before and after the pruning.

Unit	30	29	28	27	26	25	24
Epoch	10000	10000	20000	20000	15000	15000	10000
Error	0.25003	0.25033	0.24999	0.25000	0.25001	0.25001	0.25000
$R_i(B)$	1.74517	1.60366	1.37027	1.60644	1.66732	1.93119	1.89047
$C_i(B)$	0.00000	1.29848	1.32125	1.49395	1.31057	1.35015	1.68622
$R_i(A)$	0.81633	0.97321	1.27546	1.34659	1.01626	0.92205	1.67675
$C_i(A)$	0.00027	0.00018	0.00011	0.00021	0.00014	0.00018	0.00015

Table 3. Parameters for consecutive pruning of the units

This algorithm was compared with the pruning of the original augmented network for seven processing units simultaneously. That pruning was accomplished with training the network with a set of 56 patterns containing the first 23 values with which the network was grown and the last seven activations were repetitive permutations of six zeroes and one -1. The pruning was accomplished after 10000 epochs of training, but the weights of the pruned units lied in the interval $(-1.667480, 0.152730)$, where the weakest connection is from the 26-th to the 24-th unit, and the strongest connection is from the 25-th to the 27-th. So, the algorithm gave better results.

Experiments were taken about the starting node of the pruning. First the algorithm was tested on pruning the first processing unit of the network, and then the 18-th unit was taken to be the first to be pruned. The same result was achieved when the 22, 28 and the 11-th unit were taken as a starting unit in the pruning algorithm. The Fig.2 shows the weights of these pruned units.

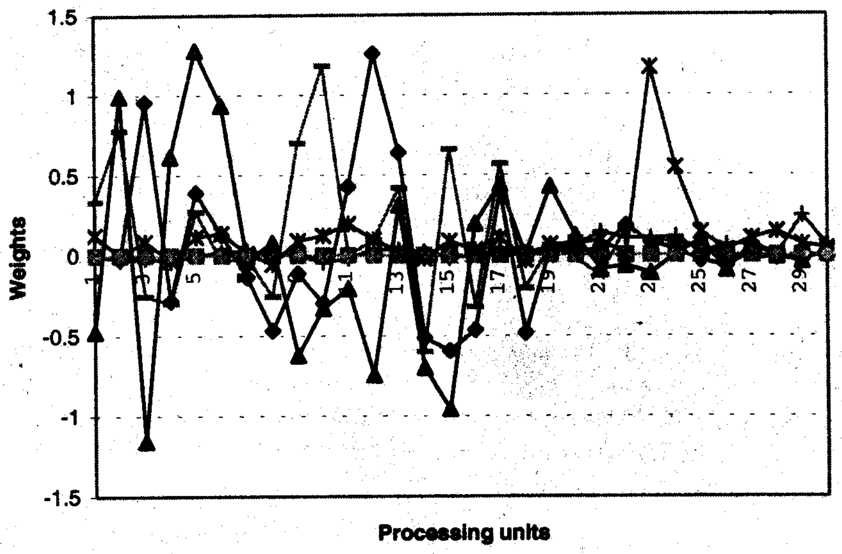

Figure 2: Weights of the connections of several starting points after individual pruning when LIFO criterion is not used

As shown in the figure, the connections from the inhibited units were always very near zero (0.0001), but the connections to the units sometimes had much greater weights than 0.2, and those weights could not be diminished, as it could be seen from the next table.

Unit	23	1	18	22	28	11
R_i	2.05454	8.09501	11.20842	3.85527	1.3866	7.62421
C_i	0.0001	0.00018	0.00023	0.00015	0.00029	0.0005

Table 4. C_i and R_i of the units after their pruning when the LIFO criterion is not followed

Additional experiment with pruning of a network consisting of 30 cells using the relaxed algorithm (10) was done. The network was trained within 10000 epochs with the same training patterns as the growing network was trained with. The weights of the connections lied in the interval (−0.5642, 0.99536), where the weakest connection is from the 13-th to the 16-th unit and the strongest connection is from the 17-th to the 16-th unit. The results of the pruning are shown in the following table.

Unit	30	13	16
Epochs	2000	2000	2000
Error	0.24998	0.24998	0.24998
C_i	2.31635	3.43087	3.42435
(Before/After)	0.00014	0.00009	0.00016
$R_i(B)$	1.71151	3.66850	3.28998
(Before/After)	3.25113	7.96819	7.44123
Before the pruning	(−0.19945, 0.15422)	(−0.406442, 0.36290)	(−0.85067, 0.69613)
After the pruning	(−0.276300, 0.244200)	(−0.850670, 0.790880)	(−1.056520, 1.44360)

Table 5. The results of unit pruning in a fixed architecture network

Although there were connections with weights greater than 0.2 in absolute value after pruning of the 30-th unit and R_i increased, the pruning continued with the 13-th unit, which had the strongest absolute connection with the 30-th unit. The pruning continued to the 16-th unit, because it has the strongest absolute connection with the 13-th unit. But, the pruning had to stop here because the weights of the pruned unit where greater than the weights of the starting network. The relaxed pruning algorithm (10), really helped the pruning, but the weights accumulated in the next pruned unit and the pruning had to stop after the third unit.

4. Final remarks

This paper has presented the possibilities of network pruning in autoassociative neural networks. It introduced an algorithm for pruning the network, starting from the last added processing unit. The pruning was made under the influence of the external stimuli. The algorithm was tested on different networks and was compared with the result of the simultaneous pruning of the network. Alterations of the algorithm, considering the position of the starting unit and the order of the units to be pruned was tested and showed that the results depend on the origin of the unit. Only the units from the part of the network gained by network growing algorithm could be pruned to some extent. When the algorithm was taken with relaxed conditions, the network with fixed structure could be also pruned.

5. References

[1] McClelland, J. L., Rumelhart, D. E. *Parallel Distributed Processing*, Vol.1. Foundations, The MIT Press, Cambridge, Massachusetts, 1986.

[2] Trajkovska, L. "*Modification of the Architecture of Auto-Associative Neural Networks*", graduation thesis, Institute of Informatics, Skopje, 1998 (in Macedonian).

[3] Zdravkova, K. "*Learning Architecture in Auto-Associative Neural Networks*", in Proceedings of the 20th International Conference on Information Technology Interface, Pula, June, 16-19, 1998.

[4] University of Stuttgart, Institute for parallel and distributed high performance systems, Applied Computer Science - Image Understanding, "SNNS - Stuttgart Neural Network Simulator- User Manual, Version 4.0", 1995.

ОТСЕКУВАЧКИ АЛГОРИТАМ ЗА ДМА МОДЕЛОТ НА АВТОАСОЦИЈАТИВНИТЕ НЕВРОНСКИ МРЕЖИ

Лена Трајковска

Р е з и м е

Конвенционалните техники за намалување на невронските мрежи се дефинирани само за повеќеслојните невронски мрежи. Истражувањето кое е претставено во овој труд се однесува на модификацијата на архитектурата на авто-асоцијативната невронска мрежа под влијание на прогресивна адаптација на надворешните стимули. Предложениот алгоритам за намалување на мрежата беше добиен по исцрпно експериментирање со моделот ДМА (Дистрибуирана Меморија и Амнезија). Прикажани се и резултатите од експериментите изведени над мрежа со растечка и фиксна архитектура. На крајот е направена споредба помеѓу мрежата добиена со динамичко намалување на почетната мрежата со отстранување на една по една клетка и резултантната мрежа добиена со директно намалување на почетната мрежа со истовремено умртвување на клетките.

Ognjan Prica 23,

1000 Skopje,

Macedonia

lena@artisoft.net