A Simple Approach of Improving Back-Propagation Algorithm

H. ZHU*, K. EGUCHI**, T. TABATA**, N. SUN**
* Department of Computer Science, Faculty of Engineering,
Hiroshima Kokusai Gakuin University,
6-20-1 Nakano Aki-ku, Hiroshima 739-0321, JAPAN
** Kumamoto National College of Technology
2659-2, Suya, Nishigoshi, Kikuchi, Kumamoto 861-1102, JAPAN
Phone +81-82-820-2649, Fax +81-82-820-2640
Email kohe@wuchang.cs.hkg.ac.jp

Abstract: The enhancement to the back-propagation algorithm presented in this paper has resulted from the need to extract sparsely connected networks from networks employing product terms. The enhancement works in conjunction with the back-propagation weight update process, so that the actions of weight zeroing and weight stimulation enhance each other. It is shown that the error measure, can also be interpreted as rate of weight change (as opposed to ΔW_{ij}), and consequently used to determine when weights have reached a stable state. Weights judged to be stable are then compared to a zero weight threshold. Should they fall below this threshold, then the weight in question is zeroed. Simulation of such a system is shown to return improved learning rates and reduce network connection requirements, with respect to the optimal network solution, trained using the normal back-propagation algorithm for Multi-Layer Perceptron (MLP), Higher Order Neural Network (HONN) and Sigma–Pi networks.

1. Introduction

Networks involving the use of product terms, such as HONN's[1][2] or Sigma–Pi Networks[3] are known to be capable of returning much faster learning times than those returned by their single order counterpart (i.e., perceptron and MLP). However, the fundamental drawback in such systems results from a dramatic increase in the interconnect and weight density, as the product term order increases[1]. A further characteristic of product term network performance is that a sparsely connected network is returned[4][5], following completion of training from an initial fully interconnected network. Given this characteristic, it is evident that product term networks in particular only use a subset of the available weights.

Consequently, a requirement exists for an amendment to the learning algorithm which is capable of setting to zero those weights of low magnitude during the learning process. Hence, by zeroing weights representing incorrect descriptors, it is envisaged that weights representing useful descriptors receive more stimulus, so speeding the learning process. A final requirement of such a system is that should weights representing useful descriptors be zeroed, the system can correct the 'mistake'. The design parameters for such a system are strict due to the already large practical requirements of the product term networks. Consequently, the system should incur as low as possible additional storage requirements, and make use of the variables already used by the learning rule.

Application of a faster learning rate to the sparsely connected network allows much faster convergence through the dual mechanisms of

1. reduced network interconnect/weight requirement (i.e., reduced network hardware requirements),
2. faster learning parameters (higher than in the initial learning phase).

The design of such a system is presented in section 2, with experimental results in section 3.

2. Algorithm Design for Weight Zeroing during Sigma–Pi Network Training

There are three considerations which need addressing when extracting a sparsely connected network,

1. identification when a weight is of a stable magnitude.
2. whether a stable weight is of significant magnitude.
3. when to extract the sparsely connected network.

Given that the last point has been considered in the preceding paragraph, the first two points are addressed in the following sections.

2.1 Weight Stability

A stability measure is employed to determine at what point a weight is said to represent a descriptor, or more accurately the amount of a descriptor present in the I/O relationship. Practically the measure is such that whenever the weight rate of change diminishes below some threshold, then the corresponding weight magnitude is analyzed. The weights' rate of change is evaluated as the partial differential of the present weight change parameter.

Hence, the weights' rate of change from pattern 'p' for a first order weight is given by,

$$\frac{\partial W_{p_{ij}}}{\partial y_{p_j}} = \eta\delta_{p_i}(t) + \alpha\eta\delta_{p_i}(t-1) \qquad (1)$$

Table 1 Effect of Changing Zero–Weight Threshold for (4, 5, 3) MLP Network

Zero–Weight threshold	0.1	0.2	0.5	1	1.1	1.7
Iterations	174	159	211	182	181	245
weights	32	33	24	22	22	17
Performance factor	0.79	0.84	0.87	1.1	1.04	1.06

where

$$W_{p_{ij}} = W_{p_{ij}}(t-1) + \eta\delta_{p_i}(t)y_{p_j}(t) + \alpha\eta\delta_{p_i}(t-1)y_{p_j}(t-1) \qquad (2)$$

For a weight handling second order terms this becomes,

$$\frac{\partial W_{p_{ijk}}}{\partial(y_{p_j}y_{p_k})} = \eta\delta_{p_i}(t) + \alpha\eta\delta_{p_i}(t-1) \qquad (3)$$

where

$$W_{p_{ijk}} = W_{p_{ijk}}(t-1) + \eta\delta_{p_i}(t)y_{p_j}(t)y_{p_k}(t) + \alpha\eta\delta_{p_i}(t-1)y_{p_j}(t-1)y_{p_k}(t-1) \qquad (4)$$

p	-	indicates pattern 'p' from the training Set.
i	-	index of destination neuron.
j, k	-	indexes for incoming stimuli to the present layer.

Consequently, the weight rate of change is a function of the receiving node error, where all the weights feeding that node have the same stability factor (i.e., error).

2.2 Weight Significance

Once a weight has attained the required minimum 'stability' a weight magnitude comparison is performed with respect to a reference magnitude. Should the weight magnitude fall below this threshold (the weight having already been updated using the normal update process), then it is considered to deduct from the overall network performance, and is set to zero magnitude. If the threshold is exceeded, then the weight in question is left untouched.

On the next weight update, those weights which were not set to zero are enhanced further, so pushing the I/O characteristic closer to the descriptors they represent. Two mechanisms are at work. Firstly, the normal weight update process, and secondly, there is no longer an opposing effect from those weights which are set to zero. If the action of the weight zeroing has a decremental effect on network performance, then the node stability decreases (i.e., error increases) and the weights are updated in the normal manner (Generalized Delta Rule[3]).

2.3 Considerations when Selecting Stability and Weight Zero Thresholds

It is quite possible that, during training, a weights' stability may cycle between stable and volatile conditions, as the weight space is transversed. This means that setting a high weight zeroing threshold, below which weights having a 'stable' (i.e., small) error function are zeroed, will result in potentially promising weights never having the chance to develop beyond the threshold level. Having said this, 'high' weight zero thresholds are possible when used in conjunction with lower network learning rates. Conversely, setting a zero threshold too low results in 'bad' descriptors filtering through, and the performance is no better than that of the original Generalized Delta Rule.

3. Weight Zero Algorithm Performance

The performance of a system employing the Weight Zeroing algorithm outlined in the above section is measured in two ways,

1. Ability to speed convergence.
2. Ability to select a sparsely connected network.

These requirements will be investigated in the context of the three network topologies to which the Generalized Delta Rule[3] can be applied. In all cases, unless otherwise stated, networks are judged to have converged when all nodes have maximum output difference of 0.1. Likewise, a stability threshold of 0.00005 is used in all Weight Zero Enhanced (WZF) algorithm examples.

3.1 MLP Performance with Weight Zeroing Algorithm

It is already known that the optimal network configuration for the Two Bit Adder problem[4] using the MLP architecture is that of a (4, 4, 3) network. A convergence time of 157 iterations is obtained using the standard Generalized Delta Rule with a learning rate of 1 and a momentum of 0.9. This is used as a reference to judge the performance returned by the WZF algorithm.

Increasing the number of hidden layer neurons by one (4, 5, 3) tests the WZF algorithms ability to ex-

Table 2 Effect of Changing Zero–Weight Threshold for Third Order (4, 3) HONN NETWORK

Zero–Weight Threshold	0.1	0.2	0.3/0.4	1	1.7	1.8/1.9
Iterations	980	980	984	1525	879	no
Weights	19	17	17	14	10	convergence[1]
Performance Factor	2.22	2.5	2.44	1.92	4.78	

tract sparsely connected networks, before convergence is completed, and determines whether the optimal network solution can be returned, for the same learning rate as applied to the reference network.

Table 1 shows the effect of varying the zero-weight threshold from which the following general characteristic emerges. For low magnitudes of zero-weight threshold, faster learning times are returned, at the expense of the number of weights employed. The reverse is true for weight count minimization. Consequently, for zero–weight thresholds greater and equal to 0.5 (table 1), the number of weights employed is lower than that used in the optimum (4, 4, 3) net with the Standard learning rule.

Network Performance is a function of the number of iterations required for convergence and the number of weights used in the converged network. Hence the,

Net Performance = *numb. of iterations to convergence* × *number of weights*

In order to provide a comparison of network performance between the initially optimal (4, 4, 3) and redundant node (4, 5, 3) networks, the following Normalized Performance Factor, based on the Network Performance measure above, is used. Thus,

Performance Factor = *Reference Network Performance* \ *WZE Network Performance*

3.2 HONN Performance with Weight Zeroing Algorithm

In the investigation of the performance of Higher Order Neural Networks the Two Bit Adder problem is used (network requirement (4, 3)) with the same learning rate as the MLP. The network converged in 980 iterations, and 12 of the 42 available weights were over unity magnitude (average weight magnitude 2.33). Hence, it is expected that application of the Weight Zeroing Enhanced algorithm will provide substantial reductions in the required network weight / interconnect requirements, though not necessarily in speed, due to the optimal neuron count of the network (i.e., the (4, 3) network by default cannot have a lower neuron count).

Table 2 shows that, for the optimum WZE solution, the solution identified uses a total weight count of 10 for the same training time.

3.3 Sigma–Pi Performance with Weight Zeroing algorithm

Again the Two Bit Adder problem is used, this time using a network of intermediate characteristics (4, 3, 3), permitting first and second order product terms. This represents the optimum network configuration, in terms of nodes per layer. Applying the Generalized Delta Rule to provide a performance reference for the Sigma–Pi structure, returns a training time of 271 iterations, using a learning rate of 2 and momentum term of 0.2. Application of the WZE algorithm over a range of weight zero thresholds produces the same trade off between learning time and number of weights previously located. However, because more redundant terms exist within the minimum network configuration (the reference performance having 18 weights below unity magnitude, where the network average is 2.9), then all performance factors for the various thresholds in table 3, represent a performance increase over that available from the reference.

3.4 Optimum Sigma–Pi network extraction

As in the MLP example, the number of nodes in the hidden layer is increased by one, the aim is to extract the optimum (4, 3, 3) network. Again the performance factor remains better than that returned using the reference network with the same learning parameters. Of more concern is the inability to locate the optimum network configuration until the last weight zero threshold which returns a converging network. A sharp switch is observed, between the minimum number of weights providing convergence comfortably and a weight selection unable to provide convergence. Furthermore, the correct weight distribution, approaching the optimum network configuration, have not been selected until the last maximum difference threshold of 0.1, making sparsely connected network extraction for optimal network configurations difficultly.

This increased difficulty in locating the correct weight / interconnect distribution can be attributed to

[1]Convergence attempted for a network using 11 weights.

Table 3 Effect Of Changing Zero-Weight Threshold for (4, 3, 3) Sigma-Pi Network

Zero-Weight Threshold	0.1	0.2	0.5	1	1.5	1.6
Iterations	252	210	228	201	258	307
Weights	41	34	30	24	18	17
Performance Factor	0.79	0.55	0.53	0.37	0.36	0.4

a dramatic increase in the network complexity. In the MLP example, inserting an extra hidden layer node (4, 4, 3 to 4, 5, 3) resulted in a weight increase of 7 weights. For the second order Sigma-Pi network, increasing the number of hidden layer nodes by one, results in an extra 12 weights (or 28 interconnects), on a network which is already 70 complex than the (4, 4, 3) MLP system.

4. Conclusion

A simple addition to the Generalized Delta Rule has been demonstrated capable of zeroing weights judged to be stable and of an insignificant magnitude during the training process. This addition is made possible by first showing that the node error can be interpreted as a weight stability factor describing the rate of weight change for the weights feeding that node. The interactive nature of this system means that should weights representing useful descriptors be set to zero, the system is capable of correcting the mistake (assuming that the weight zero threshold is not set too high).

For the problems considered, simulation shows that the learning time and weight / interconnection requirements are improved, though as the network complexity increases, the ability to continuously return an optimum network configuration decreases. Furthermore, the extraction of sparsely connected networks is possible, allowing application of faster learning parameters, so resulting in faster training times, though care must be applied when extracting such networks in complex systems.

References

[1] Giles C., Maxwell T., "Learning Invariance and Generalisation in Higher Order Neural Networks", Applied Optics, Vol.26, pp.4972-4978, 1987.

[2] Yang H., Guest C. C., "High Order Neural Networks with Reduced Numbers of Interconnection Weights", Journal of Neural Networks, Vol.3, pp.281-286, 1991.

[3] Rumelhart D. E., McClelland J.L., "Parallel Distributed Processing, Explorations in the Microstructure of Cognition, Volume 1 Foundations", MIT, 1986.

[4] Karnin E.D., "A Simple Procedure for Pruning Back-Propagation Trained Neural Networks", IEEE Transactions on Neural Networks, Vol.1, No.2, pp.239-242, 1990.

[5] Maxwell T., Giles C. L., Lee Y. C., "Generalisation in neural Networks: The Contiguity Problem", IEEE First International Conference on Neural Networks, Vol.2, pp.41-46, 1987.