Characterization of a Higher-order Associative Memory That Uses Product Units

Jung-Hua Wang
Department of Electrical Engineering
National Taiwan Ocean University
No.2, Pei-Ning Rd.
Keelung, Taiwan

Abstract **The Characteristics of a novel 3-layer feedforward neural network that can be used as a higher-order associative memory is studied. The network structure consists of a hidden layer that contains product units for which each input is raised to a power determined by a trainable weight. The operation of the network consists of three steps:(1) preprocess the prescribed associative vectors and determine the principal connection weights (i.e., the first phase learning); (2) estimate the required number of product units and connections based on the results from (1); and (3) train the network using the backpropagation algorithm until satisfactory recall accuracy is achieved. The use of this two-phase learning is shown to enable us to achieve:(a) learning without requiring long training time (as often seen in a usual backpropagation-type network); (b) major reduction in the number of connection weights.Various interesting characteristics of the network, including input noise tolerance and fault tolerance, can be seen in this network.**

I. INTRODUCTION

A product unit is a feedforward network of the backpropagation [1,2] type proposed by Rumelhart and Durbin [3]. Instead of calculating a weighted sum this unit calculates a weighted product, where each input is raised to a power determined by a trainable weight. The idea behind the product unit is that it can learn to represent any generalized polynomial term in the input, and therefore help to form a better representation of the data in cases where higher order combinations of the inputs are significant. As with summing units such as perceptrons, there is only one weight per input. This is different from the Sigma-pi units [2] which not only provide a weight to each input but also to all higher order products of inputs. The output of a product unit is

$$Y_i' = \prod_{r=1}^{N} X_r^{w_{ri}} . \qquad (1)$$

Manuscript received May 13, 1993. This work was supported in part by National Science Council under Grant No. NSC 82-0408-E-019-003.

A possible form for product unit is shown in Fig 1. The weights W_{ri} are treated as variable weights, and are trained by gradient descent on the output sum square error. Since the W_{ri} can take fractional and negative values, product units provide much more generality than just allowing polynomial terms. Note that no nonlinear function is applied to the output of a product unit. The outputs of all summing units are compressed during training using a nonlinear, differentiable function such as the sigmoid $Sig=(1-e^{-X})/(1+e^{-X})$. i.e.,

$$Y_j = Sig\left[\sum_{i=0}^{N} Y_i' T_{ij}\right] . \qquad (2)$$

Note that, during the recall process, Y_j may be thresholded by a hard-limiting function to get binary values (if in binary applications).

It has been shown that network configured as in Fig 1 can be trained to learn as an associative memory, of which the memory capacity per neuron also estimated [2]. This paper proposes a 3-layer feedforward neural network that has similar look of Fig 1, but utilizes our previous results of higher order associative memory [4,5] as the first phase learning. The effects and advantages of breaking the learning process into two phases are described, and the characteristics of this special network are studied.

II. THE NETWORK CONFIGURATION

A. Preprocess (The First Phase Learning)

To illustrate, we use a 2nd order network as an example. The first step in forming the network is to estimate the capacity [4] of a 2nd order Hebbian-type associative memory according to (3)

$$M \approx \frac{N^2 + \left(6-\frac{4}{N}\right)[\ln(N)+5]}{\left(6-\frac{4}{N}\right)[\ln(N)+5]} , \qquad (3)$$

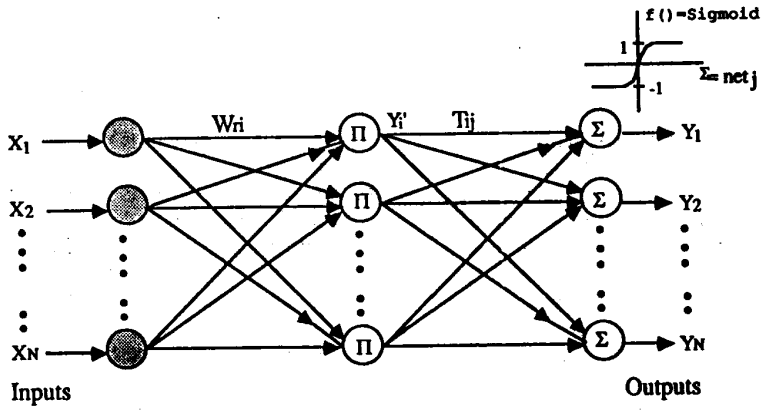

Fig. 1. A fully connected product units network.

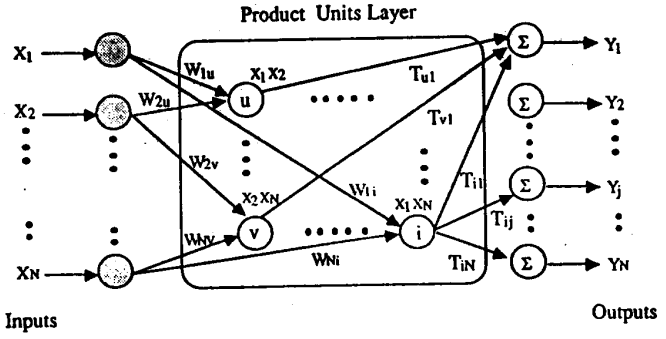

Fig. 2. A higher -order associative memory using product units.

where M is the number of prescribed patterns and N is the dimension of each pattern. The second step is to identify all principal connection weights and their associated XjXk input combinations. Note that the 2nd order connection weight Tijk is prescribed by

$$T_{ijk} = \sum_{\kappa=1}^{M} V_i^{\kappa} V_j^{\kappa} V_k^{\kappa}, \quad \text{where } 1 \le i,j,k \le N \text{ and } i \ne j \ne k. \qquad (4)$$

Assume T is the set of connection weights comprises all Tijk. By principal connection weights we mean there exists a subset of T, Tpr, that contain more information than the others. The recall capability of using only principal weights Tpr, where Tpr $\in$ T, is slightly less than if otherwise all weights in T are used (in fact, the accuracy difference can be compensated by the 2nd phase learning:backpropagation). In our previous work [5], it has been shown that these principal terms Tpr are shown to be located in the range $\sqrt{M} \le |Tijk| \le M$ and account for less than 50% of the total number of weights. Finally we implement these principal weights in the network by (i) generating the number of hidden units (product units [3]) by choosing these units from the XjXk input combinations; each hidden unit receiving only one input combination of XjXk (designated as an i,u,or v unit in Fig 2, e.g. X1XN =unit i and Tij= Tj1N=connection weight between unit i and the output neuron j) ; (ii) setting all connections wri from inputs to product units to be 1(maybe changed during training); (iii) setting all connections between product

units and output units to the values of the normalized principal weights (i.e., normalization by dividing Tijk by M).
Due to the sigmoid function, this normalization is necessary in order to avoid the error signal fed back from the output neuron from being too small.

Thus, the idea behind this special network configuration is twofold:(i) starting with only principal weights means that more than 50% of the total connection terms can be eliminated, yet fairly good recall accuracy Pdc can be achieved; and (ii) if the result of this significant reduction in connection weights is to be utilized in preparing the initial weights and product units for the net, the training time required should be much lower than if done otherwise, due to the fact that the net is to learn to improve from a certain value (e.g. 0.85) of Pdc to Pdc≥0.99, instead of starting from scratch. Also, since the computations of (4) and principal connection weights are just straight forward, the first phase learning will take no more than 10% of the total learning time!

B. The Second Phase Learning

After setting up the network and connections, the network can then be trained to learn, using backpropagation, to improve the recall accuracy Pdc from a certain value (due to the prior knowledge obtained from the first phase learning in (3)) to ≥0.99. Without going into math details, the update formulas of connection weights in Fig 2 can be derived as

$$T_{ij}(t+1) = T_{ij}(t+1) + \eta_1 \delta_j Y_i' , \quad \text{and}$$

$$W_{ri}(t+1) = W_{ri}(t+1) + \eta_2 \delta_i \frac{dY_i'}{dW_{ri}} , \qquad (5)$$

where $\sigma_i = \Sigma T_{ij}\sigma_j$, and $dY_i'/dWri$ is obtained by taking the derivative of Y_i' (i.e., the output of the product unit i) with respect to Wri . Note that for the special case of binary inputs, $dY_i'/dWri$ reduces to

$$\frac{dY_i'}{dW_{ri}} = -\pi \sin\left[\pi \sum_r w_{ri}\right], \quad \text{for } x_r = -1 ;$$

$$= 0, \quad \text{for } x_r = 1. \qquad (6)$$

Since the network is already capable of achieving a certain high value of recall accuracy before 2nd phase training (due to the first-phase learning by Eq.(2)), the average changes in the connection weights should be mild and a quick adaptation process can be expected.

III. RESULTS

In the past the problem of estimating the number of hidden units has never been solved. The proposed network avoids this difficulty by preprocessing the associative vector pairs, and provides the double advantages of speeding the learning and eliminating redundant connection weights. Our result has indicated (not shown here) that , given the full set of principal connection weights and their corresponding product units, the learning speed is at least 50% less than if no preprocessing is performed

Furthermore, It is even more interesting to know if the network can still learn if not all the principal connection weights is used. That is, by examining the trade-off between the number of product units and the required training time, it is possible to determine the best network configuration that meets the performance requirements. In the following simulation results, using N=30 and M=16 we show various properties of the network (mainly fault tolerance are shown here).

A. Decreasing Number of Hidden Units

Fig 3 illustrates that , given the number of iterations =6000 and input error bits b=2, the number of hidden units can be decreased down to 303, which is 70% of the original number, and the network is still able to learn to raise Pdc to more than 0.99. This is significant because the decreasing number of hidden units gives a reduction in the number of connections (T-links and W-links in Figure 1).

B. Training Time vs. Number of Hidden Units

Given that Pdc is required to reach ≥0.99 after training, Fig 4 shows the trade-off between training time and the number of hidden units required. Generally speaking, when the number of hidden units decreases, the needed training time goes up.

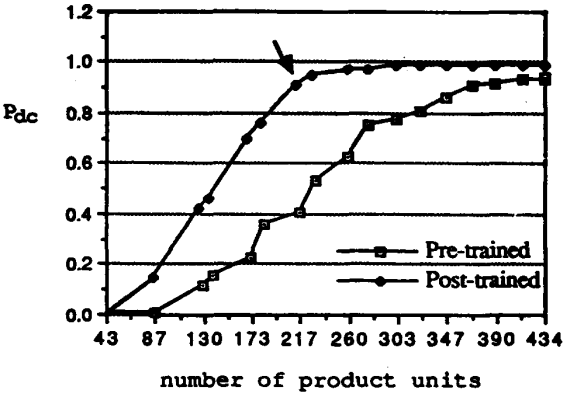

Fig. 3.

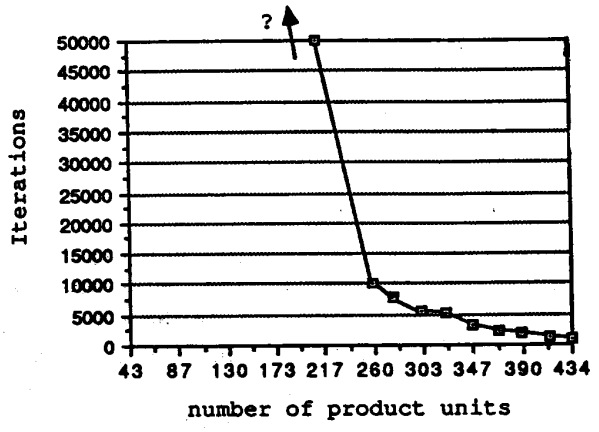

Fig. 4.

IV. CONCLUSIONS

The characteristics of a higher-order associative memory utilizing product units in the hidden layer and prior knowledge obtained from the hebbian association rule have been investigated. The application of the two-phase learning is shown to be effective, in terms of both learning speed and network complexity. The trade-off between the number of hidden units and the required training time has been shown. In particular, we have seen that the network shows strong fault tolerance capability.

REFERENCES

[1] P. J. Werbos, "Beyond regression:New tools for prediction and analysis in the behavioral sciences," Doctoral Dissertation, Appl. Math., Harvard Univ., 1974.

[2] D. E. Rumelhart and J. L. McClelland, Eds., *Parallel Distributed Processing, Explorations in the Microstructure of Cognition. Vol. 1: Foundations*, MIT press, 1986.

[3] R. Durbin and D. E. Rumelhart,"Product units: a computationally powerful and biologically plausible extensions to backpropagation networks", *Neural Computation, vol. 1*, pp.133-142, 1989.

[4] J. H. Wang, "On a neural network associative memory that uses indirect convergence ",*Proc. of 1992 IEEE Intl. Conf. Ion Systems, Man, and Cybernetics*. Vol. 2, pp.1033-1037. Chicago, IL,1992.

[5] J. H. Wang, T. F. Krile, and J. F. Walkup, "Reduction of interconnection weights in higher order associative memory networks", *Proc. of Intl. Joint Conf. on Neural Networks*, vol. 2, pp.177-182, Seattle, WA., 1991.