Qualitative Analysis of the BP Composed of Product Units and Summing Units

Jung Hua Wang and Jia Hon Lin
Department of Electrical Engineering
National Taiwan Ocean University
No. 2 Pei-Ning Rd, Keelung, Taiwan
E-mail: jhwang@celab1.ee.ntou.edu.tw
Phone: +886-2-462-2192 Ext. 6207
FAX: +886-2-462-2192 Ext. 6203

ABSTRACT

In this paper, we qualitatively analyze networks that contain product units. By replacing the neurons in traditional back-propagation (BP) nets with product units in hidden layer gives us a different type of BP network called P-S model. We further extend P-S to P-S(in) by adding direct connections from input neurons to output neurons. By comparing with traditional BP nets that consists of ordinary summing units, we examine performance of product unit networks in solving TC, XOR, AOX, and other hard binary problems such as odd and even parity problems. The results show that product units outperforms traditional BP nets in terms of both hardware efficiency and training requirement.

1. INTRODUCTION

Study on the product unit [1] has not received much attentions, and publications on this subject are limited[1, 2, 3]. It is well known that traditional BP networks consist of summing units. The output of a summing unit is determined by (1), where X_i is the input for the i_{th} neuron and W_i is the associated connection weight.

$$y = f((\sum_{i=1}^{N} X_i W_i) - \theta) \tag{1}$$

Because the summing unit can only learn the linear combination of the inputs, we propose the product unit which can provide a BP network with nonlinear learning.

This work was supported by National Science Council (NSC 82-0408-E019-003)

The output of product unit is determined by (2), where X_i is the input and p_i is the associated weight.

$$y = \prod_{i=1}^{N} X_i^{P_i} \tag{2}$$

Because P_i is on the exponential term, learning arbitrary complex mapping function will make possible by adjusting values of P_i through training. For simplicity, we define a S-S model as a traditional BP network, whereas a P-S model is a BP net that contains only product units in the hidden layer and all summing units in other layers. To differentiate, we also denote S-S and P-S model that have direct connections from input neurons to output neurons as S-S(in) model and P-S(in) model. The remaining sections of this paper are organized as follows. In Section II, we demonstrate that P-S model outperforms S-S model in solving the well known XOR problem. P-S(in) model are discussed in Section III, we show that product unit is more capable in learning internal information than summing unit. A hard binary problem of parity is cited as another justification. To obtain better understanding of computational property of the product unit, we also examine other examples. The TC problem is solved in Section IV, followed by the problem of AOX (AND, OR, and XOR) in section V. Finally, conclusions are made in Section VI.

2. P-S AND S-S MODELS

From (2), we see that product units in P-S model potentially provide more nonlinearity than ordinary summing units, hence they are suitable for solving nonlinear-separable problem. In the following discussions, XOR is used to evaluate performance of P-S model and S-S model.

0-7803-2559-1/95 $4.00 © 1995 IEEE

Since XOR is nonlinear-separable, it is necessary to train thresholds in S-S model. The reason for doing this can be seen in Fig. 1.

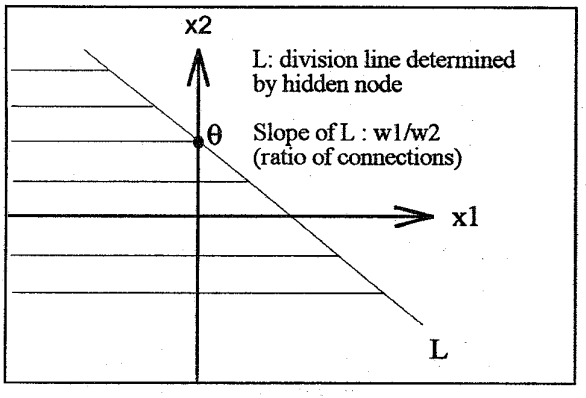

Fig.1 Division surface determined by a summing unit.

From Fig. 1, we can see that the solution of decision surface depends on the values of the connection weights and threshold θ. Thus, in the case of two input neurons, a hidden neuron provides a division line with slope w_1/w_2 . The intersection of division line on x2-axis is equal to θ (i.e., threshold). The division line is given by (3), where x1 and x2 are two input to hidden unit and w1 • w2 are the associated weights.

$$x_1 w_1 + x_2 w_2 - \theta = 0$$
$$\Rightarrow x_2 = -(w_1 / w_2) x_1 + \theta \qquad (3)$$

Now, it is clear that division line will pass through the original point (0,0) of the (x1, x2) plane, provided no training is given to summing unit. Therefore, it is impossible to solve XOR problem using this simple S-S model. But if P-S model is used, only single neuron is needed in hidden layer to solve the XOR problem. In addition, it requires no training on threshold. Thus, product unit not only outperforms summing unit in requiring less number of hidden-layer neurons, it is more efficient in training process. An example of a P-S network trained for XOR is shown in Fig. 2.

In this section we have shown benefits of using product unit, namely, good hardware efficiency and less training requirement. In fact, this is true for other training problems such as the parity and TC problem.

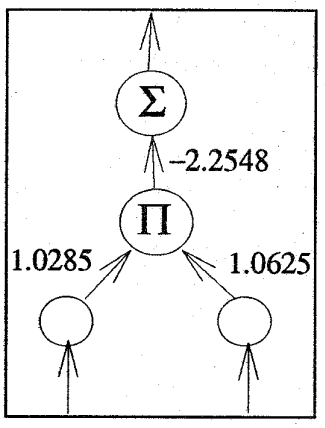

Fig. 2. The trained P-S model for XOR.

3. EXTENSION TO P-S(IN) MODEL

Product units also perform well in P-S(in) model in which the input neuron has direct connections to the output neurons[4]. To demonstrate, the network after training for six dimension parity problem is shown in Fig. 3 (initial weights are given randomly).

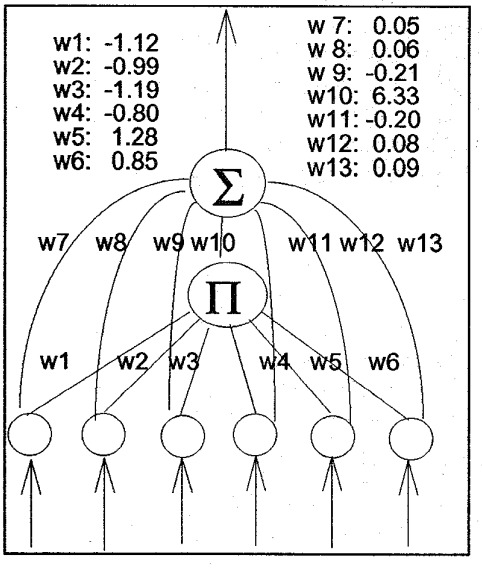

Fig. 3 The network after training for six dimensional parity problem.

What if dimension is larger than 6? In that case, training would become impractical due to immense number of the possible training patterns. To solve this difficulty, we have analyzed the parity problem and

devise the following algorithm useful in assigning the values for connection weights:

1. Use only one product unit in hidden layer and randomly assign either +1 or -1 to the weight between input unit and hidden unit.
2. Assign +1 to the weight between the output for recognizing *odd-parity* input strings; Assign -1 to the weight connecting the output unit and hidden unit, if the network is used for recognizing *even-parity* input strings.

To justify this algorithm, let us examine (4) where the input vector is (1,-1,-1,-1,-1,1). If all weights between the output and hidden units are set to 1, and all weights between input and hidden units are set to 1 or -1.

$$
\begin{aligned}
y &= \prod_{i}^{n} x_i^{Pi} \\
&= (1)^{-1}(-1)^{1}(-1)^{1}(-1)^{1}(-1)^{1}(1)^{1} \\
&= (-1)^{1+1+1+1} = (-1)^{4} \\
&= 1
\end{aligned}
\qquad (4)
$$

Thus, the output value of positive 1 indicates the input string is of odd-parity. This algorithm has the advantage of solving parity problem of arbitrary dimension and the solution comes without any training. It is noted in using this algorithm, there is no need to use direct connections from input units to output units as in Fig. 3.

4. THE T-C PROBLEM

P-S model exhibits better performance even for more complex problem such as the T-C problem. Fig. 4 gives an example of the 3x3 vector arrangements for rotational invariation. We simulate the TC problem on a P-S model with 3 product units, and the initial weights are set randomly. It is shown that about 1000 iterations is needed to obtain a solution (the convergence of training is shown in Fig. 5)

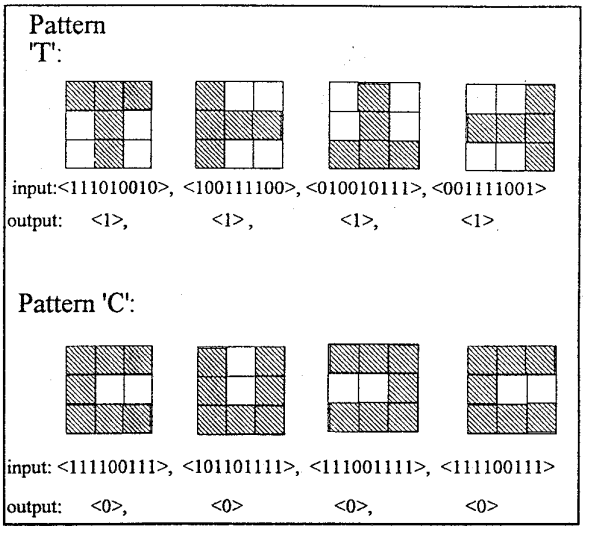

Fig. 4 Input vector arrangement for T-C problem

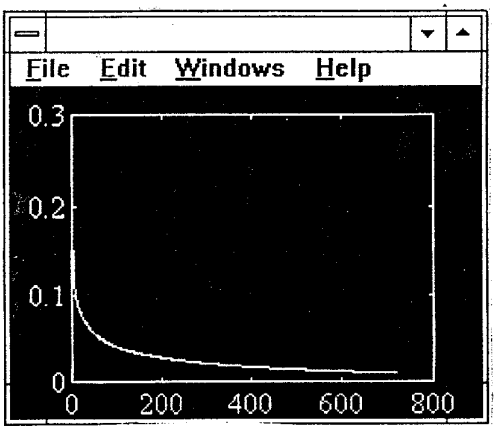

Fig. 5. Convergence for the TC problem

Note that we can speed up training time for the TC problem due to the sensibility of product unit in the number of 1 and 0 in input vector (as can be seen in previous parity problem). As such, we can rearrange 3x3 vector shown in Fig. 6 to improve the training speed. The network is still a PS model with 3 product units, and the initial weight are set randomly. It shown that only 100 iterations is need to get a solution (the convergence of training is shown in Fig. 7)

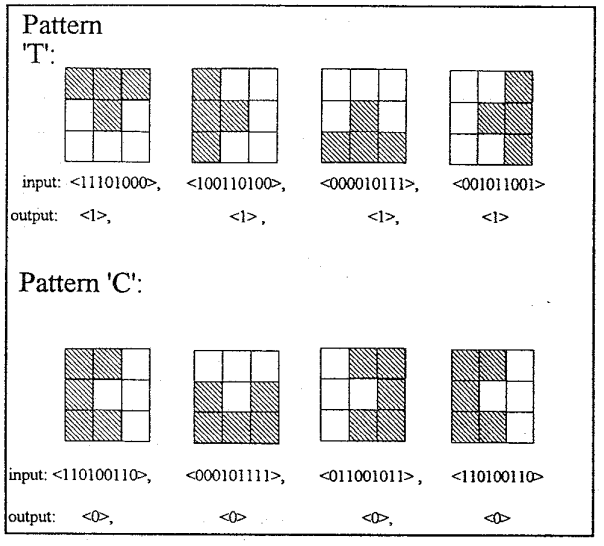

Fig. 6. Rearrangement of taining patterns for TC problem.

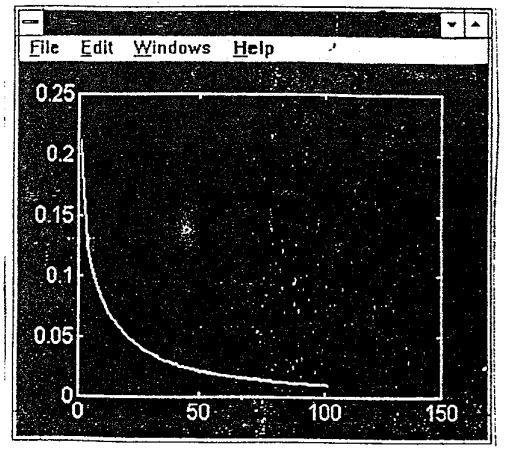

Fig. 7. Convergence of training for TC problem (after rearragement the input vector)

5. THE AOX NETWORKS

We have made an attempt to solve the problem of AOX, in which XOR and two other logic functions AND and OR are computed simultaneously (the network is shown in Fig.8). The problem are solved using different models previously described. The computed output error rates are compared for these models, and it turns out that among 4 different models (shown in Fig. 9), P-S(in) and S-S(in) are the best two choices for implementing AOX. They both gives error rate less than 0.001 when the number of hidden units>1.

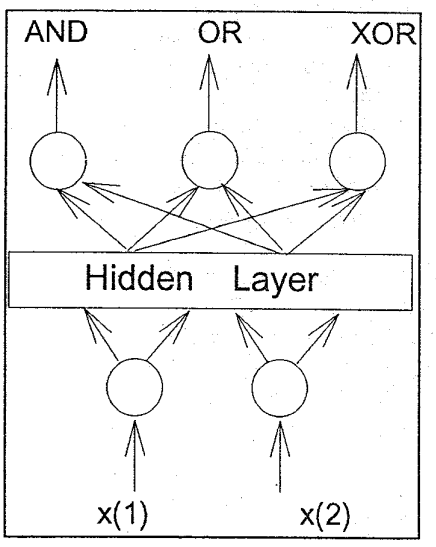

Fig. 8. The network for AOX.

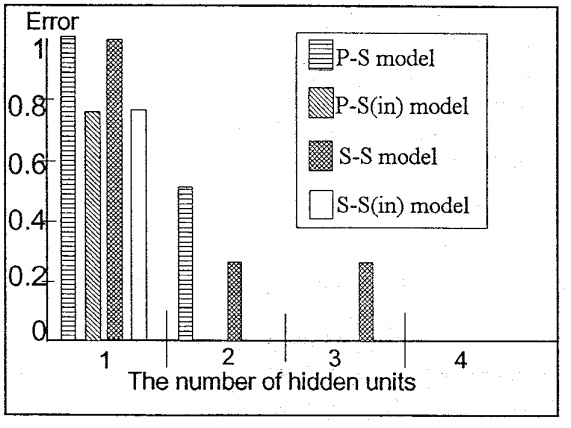

Fig. 9. Statistics (Mean Square Error) of the different implementations for AOX.

6. CONCLUSIONS

In this paper, we have seen that the including of product units in traditional back-propagation networks can result in additional nonlinearity, decreasing network size and increasing learning speed. In our simulations, we examine performance of product unit networks in

solving TC, XOR, AOX, and other hard binary problems (such as odd and even parity problems) . Our results show that product units outperforms traditional BP networks in terms of both hardware efficiency and training requirement.

7. REFERENCES

[1] Durbin, R. and Rumelhart, D. E., "Product Units: A Computationally Powerful and Biologically Plausible Extension to Backpropagation Networks", Neural Computation, Vol. 1, pp. 133-142, 1989

[2] Janson, D., Frenzel, J. "Application of Genetic Algorithms to the Training of Higher Order Neural Networks", Journal of system Eng. Vol. 2 No. 4 pp. 272-16, 1991.

[3] Chen Y., Bastani, F. "ANN with two-dendritite Neurons and its weight initialization, "Proceedings of IJCNN, 1992. Vol.3, pp. 139- 46.

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