Paper

Hybrid Universal Learning Networks

Dazi Li* Non-member
Kotaro Hirasawa** Member
Jinglu Hu* Member
Junichi Murata* Member

A variety of neuron models combine the neural inputs through their summation and sigmoidal functions. Such structure of neural networks leads to shortcomings such as a large number of neurons in hidden layers and huge training data required. We introduce a kind of multiplication neuron which multiplies their inputs instead of summing to overcome the above problems. A hybrid universal learning network constructed by the combination of multiplication units and summation units is proposed and trained for several well known benchmark problems. Different combinations of the above two are tried. It is clarified that multiplication is an essential computational element in many cases and the combination of the multiplication units with summation units in different layers in the networks improved the performance of the network.

Keywords: hybrid Universal Learning Networks, multiplication units, summation units

1. Introduction

Universal Learning Network(ULN) is a general framework for modeling and control of the complex systems widely found in the real world[1]. Although it is generally used for dynamical systems that can be described by using a set of related equations, in this paper we concentrate mainly on static ULNs, that is, static feed-forward networks consisting of two kinds of elements: nodes and branches. The nodes correspond to equations and branches to their relations. The nodes may have continuously differentiable nonlinear functions, e.g., sigmoidal functions or neuro-fuzzy functions.

Design of a neural network is mainly motivated by analogy with human's brain, which is testified to be a good way in many fields, such as identification and control. Researchers always look to neurobiology for new ideas to solve problems and improve the existing methods and algorithms. The summation unit with sigmoidal activation function is now very generic organisms of feed-forward neural networks. Successful applications have been found in many fields such as pattern recognitions and control systems. Generally conventional ULNs consisting of sigmoidal units try to make all the units uniform, that is, all the units being used have the identical node type and activation function. Though such dealings make the networks' training and programming more easy, it is, to some extent, a little far away from the real nervous systems in the sense of diversities and nonlinearities.

The struggle to understand more about human brain has never stopped. Researchers have found that multiplication plays an important role in single neuron computation[2]. In recent years, also evidences have been accumulated that specific neurons in the nervous system of several animals work in a multiplicative way. This is not an unexpected result considering the complexity and sophistication of the chemical processing at the synapses. As the nonlinearity is one of the most useful and important specifications in neural networks, pure sigmoidal networks must emphasize the nolinearity in virtue of different activation functions and hidden layers. In particular, the interaction of synaptic inputs is known to be essentially nonlinear[3].

The work described in this paper addresses the use of multiplicative-like neurons in Universal Learning Networks. In the next section, we will give a general review of some kinds of networks where multiplication has been employed to increase the capabilities of the whole neural networks. In Section 3, we will describe hybrid Universal Learning Network constructed by proposed multiplication units and summation units and the influence on the learning speed. Simulations results to compare the proposed methods and conventional sigmoidal networks will be presented in Section 4. In the last section, we will summarize and discuss our methods.

2. A General View of Multiplication Units in Neural Networks

Summation unit which combines all the inputs by summation is not the only nonlinearity that plays an important role in information processing in the nervous systems. Over the years, networks with multiplicative computations of the inputs have been devel-

* Graduate School of Information Science and Electrical Eng., Kyushu University.
6-10-1, Hakozaki, Higashi-ku, Fukuoka 812-8581
** Graduate School of Information, Production and Systems, Waseda University.
Hibikino 2-2, Wakamatsu, Kitakyushu 808-0135

oped and applied to many applications involving function approximation, classification and control systems. The most well-known units that comprise multiplicative synapses perhaps are higher-order neurons(HONs). Higher-order neural networks(HONNs) constructed by the HONs have been developed to enhance the nonlinear expression ability of the feed-forward multiplayer networks. A basic HON, with the output y and inputs $x_j, x_k, x_l, \ldots$ can be computed as:

$$y = f(w_o + \sum_j w_j x_j + \sum_{j,k(j\leq k)} w_{jk} x_j x_k + \sum_{j,k,l(j\leq k\leq l)} w_{jkl} x_j x_k x_l + \ldots), \cdots\cdots (1)$$

where $f(.)$ is the sigmoidal activation function.

Although higher-order correlations enable the networks to learn geometrically invariant properties more easily, it was noticed that the number of hidden units in the fully connected HONNs increases exponentially with the number of inputs. In fact, the number of parameters, that is, the weights, increases rapidly with the number of inputs and becomes unacceptably large for use in many situations. Consequently, typically only second or third order networks are mostly considered in practice[4].

Another class of higher-order networks, known as Sigma-Pi networks also suffer from the same problem that a combinatorial increase in the number of product terms will lead to the increase of the number of weights. A Sigma-Pi network contains higher-order terms at each layer, often the layers have summation units fed via weighted connections by outcomes of intermediate multiplication units. Sigma units compute the sum of weighted inputs h_i from the lower layer,

$$\alpha_j = \sum_i w_{ij} h_i + \theta_j, \cdots\cdots (2)$$

while the pi units compute the product of weighted inputs h_i from the lower layer:

$$\alpha_j = \prod_i w_{ij} h_i. \cdots\cdots (3)$$

The output h_j of the unit j is passed through the sigmoidal activation function:

$$h_j = f_j(\alpha_j);$$

M. I. Heywood developed a framework for improved training of Sigma-Pi networks by implementing only a subset of the total number of product terms to avoid excessive weight counts[5]. However, for many cases, higher-order terms play a very important role and cannot be easily ignored.

Pi-Sigma network[6] developed by Yoan Sin et. al is said to be efficient for pattern classification and function approximation. The name came from the structure that the output layer is a simple multiplication unit processing all the outputs from the hidden layer by a product. The connections from summation units to an output have fixed weights of 1. The output y of the output layer is given by:

$$y = f(\prod_j h_j), \cdots\cdots (4)$$

where h_j is the output from linear summation units in the hidden layer.

$$h_j = \sum_i w_{ij} h_i + \theta_j. \cdots\cdots (5)$$

Pi-Sigma network uses products of sums of input components instead of sums of products as in Sigma-Pi networks. The main difference of Sigma-Pi and Pi-Sigma networks lies in the combination sequence of the summation units and multiplication units, which gives rise to the two names of networks.

Another type of multiplicative-like unit we want to mention, product unit, was proposed by Richard Durbin and David. E. Rumelhart[7]. Differently from higher-order networks mentioned above, computation executing in the unit is as follows:

$$h_j = \prod_i x_i^{p_i}, \cdots\cdots (6)$$

where p_i will be treated in the same way as variable weights. Network constructed by product unit can, in one different way, automatically learn the higher order terms.

From the above, conclusion can be given that: 1)There are different ways to model the multiplicative-like computation which do exist in the neural networks; 2)There are many different combinations of multiplication units and summation units, which lead to better performance of the networks. On the basis of the above multiplication units, in this paper, a kind of multiplication units was proposed, which, inherit the higher-order properties of all the above units while avoiding the explosion of the number of weights with the increase of the inputs. In fact, our neuron model has exact the same number of weights as conventional summation neurons while reaching the order until the number of the inputs. The results show that multiplication unit can be considered as an alternative to summation unit without significantly increasing the complexity of computing and learning like higher-order networks. Influence of the proposed unit on the performance of the networks as well as the possibility to cooperate with different units was investigated. We also show that the combination of multiplication and summation units shall not be limited to several known formats.

3. Hybrid Universal Learning Networks

Universal Learning Network(ULN) has just been introduced[1]. Although the dynamical properties make it very successful in applications of control systems such as robust control and system identifications[8][9][10], in this paper, for simplicity, we mainly investigate the static multiplicative ULN, although the result is thought to be applicable to the dynamical ULNs. The hybrid ULN proposed here will be constructed by summation and multiplication units together, and different units will have different activation functions.

3.1 Models of Summation and Multiplication Units in Hybrid Universal Learning Networks

Model of summation units used mostly by ULNs is the model first proposed by MuCulloch & Pitts.

1) *summation unit:* With output value h_j of node j, generic summation unit used in ULNs can be expressed specifically as

$$h_j = f_j(\alpha_j), \quad \cdots\cdots (7)$$

$$\alpha_j = \sum_{i \in JF(j)} w_{ij} h_i + \theta_j, \quad \cdots\cdots (8)$$

where w_{ij} is the weight parameter from node i to node j; $JF(j)$ is the set of suffixes of nodes which are connected to node j; θ_j is the threshold parameter of node j. Function $f_j(.)$ that governs the operation of the nodes can be any continuously differentiable functions, typically sigmoidal function

$$f_j(\alpha_j) = \frac{z_j}{1 + e^{-\phi_j \alpha_j}}, \quad \cdots\cdots (9)$$

or hyperbolic tangent function

$$f_j(\alpha_j) = z_j \frac{1 - e^{-\phi_j \alpha_j}}{1 + e^{-\phi_j \alpha_j}}, \quad \cdots\cdots (10)$$

can be employed. ϕ_i is the slope parameter of node j and z_j is the gain parameter of the function. We mean by ϕ_j and z_j that different nodes can have different activation functions or different parameters in the basis function in ULN so as to make the networks have multiformities.

2) *multiplication unit:* Differently from the summation unit described above, in spite of combining all the inputs by a linear summation, multiplication unit developed in [11] multiplies all the inputs after subtracting an adjustable weight from them. The presentation of the new model mainly based on the following points:

- The overall network will not increase the computation complexity with embedding of the multiplication unit;
- By combining the multiplication unit with conventional summation unit, performance of standard sigmoidal networks can be improved so that more parsimonious structures can be realized.
- The combinational network should be trainable by a standard method, such as back-propagation regime, without a new training method needed.

Operation of the proposed multiplication unit is expressed as:

$$h_j = z_j \prod_{i \in JF(j)} (h_i - w_{ij}) + \theta_j \quad \cdots\cdots (11)$$

where z_j is the gain parameter of node j. No nonlinear activation function is applied to the output of multiplication unit.

From the model of the proposed unit, we can see that, it is in fact a different product unit. If the dimension of inputs is represented by N, the increase of the number of weights with the increase of inputs by multiplication units is the same as that of summation units while reaches until the order N, which is significantly less than that of higher-order units. The reason is that weights of higher-order terms in the representation are in fact combined with products of different weights respectively. In fact, we can form a polynomial of the inputs by the unit. From the resulting polynomial, we can show how the complexity of computation can be reached with the simple combination of all the inputs. With 2-dimentional inputs, if we take the threshold θ_j as 0, the output of a multiplication unit can be described as:

$$h_j = z_j (h_1 h_2 - \sum_i h_i \sum_{k \neq i} w_{kj} + w_{1j} w_{2j}) \cdots\cdots (12)$$

that for n-dimentional inputs will be:

$$h_j = z_j (\prod_{i=1}^{n} h_i - \sum_k w_{kj} \prod_{i \neq k}^{n} h_i + \ldots + (-1)^{n-1} \sum_i h_i \prod_{k \neq i}^{n} w_{kj} + (-1)^n \prod_{i=1}^{n} w_{ij}) \cdot (13)$$

which looks like the appearance of a higher order unit although the number of parameters is almost the same as that of a conventional summation unit. There is one additional parameter, i.e., the coefficient z_j, which acts as the gain of the whole unit.

3) *Structure of hybrid ULNs:* In our hybrid ULNs, multiplication units can cooperate with summation units freely with any kind of combination. Multiplication unit can be used in any hidden layers or output layers. A specific combination of multiplication units with summation units can be shown in Figure 1. All the multiplication units in different layers form a subnet in the network, and also different activation functions are used by different nodes and layers. Combination of summation and multiplication units by the hybrid network come to the forms of existing Sigma-Pi or Pi-Sigma networks.

3.2 Learning of the Proposed Multiplication Units

With a good cooperation with the summation units, a parameter λ_m of multiplication units can also be trained by minimizing a criterion function L based on the gradient method:

$$\lambda_m \leftarrow \lambda_m - \gamma \frac{\partial^\dagger L}{\partial \lambda_m}, \quad \cdots\cdots (14)$$

where γ is the learning coefficient assigned a small positive value and $\frac{\partial^\dagger L}{\partial \lambda_m}$ is the ordered derivative defined by Werbos which means the net change of the criterion

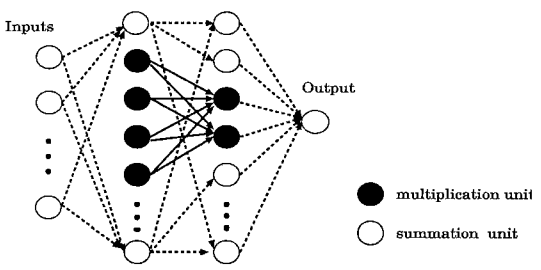


Fig. 1. An example of the structure of hybrid ULN.

function L caused by the change of λ_m with other variables being fixed.

The criterion function L chosen by back-propagation method usually is the sum of errors between the network outputs and their desired values. By calculating the derivatives of L with respect to a parameter, the gradient-based optimization can be executed as:

$$\frac{\partial^{\dagger} L}{\partial \lambda_m} = \sum_{d \in JD(\lambda_m)} \left[\frac{\partial h_d}{\partial \lambda_m} \delta(d) \right] + \frac{\partial L}{\partial \lambda_m}, \cdots\cdots (15)$$

where δ is defined to be:

$$\delta(j) = \sum_{k \in JB(j)} \left[\frac{\partial h_k}{\partial h_j} \delta(k) \right] + \frac{\partial L}{\partial h_j}, j \in J \cdots\cdots (16)$$

λ_m : value of m-th parameter,
$JB(j)$: set of suffixes of nodes which are connected from node j,
$JD(\lambda_m)$: set of suffixes of nodes which include parameter λ_m
J : set of suffixes of nodes

For multiplication units, calculation of the derivative of $\frac{\partial h_k}{\partial h_j}$ become:

$$\frac{\partial h_k}{\partial h_j} = z_k \prod_{j' \neq j} (h_{j'} - w_{j'k}) \cdots\cdots (17)$$

In general, there are two alternative methods to update the weights of multiplication unit, namely *off-line* and *on-line*. In the off-line method, sequential weight-updating values are stored during calculating all the updates. After all the updating values are calculated, all the weights are changed together. On-line method changes weights right after calculating the corresponding weight-update. So the concrete procedure of the gradient-based method is:

Initialize weights generated by small random values;
Repeat(for each epoch):
  Choose one input and present it to the input layer;
  Repeat (for each layer of the network)
    • Propagate the signal forward through the network;
      if (multiplication unit)
        Calculate h_j by equation (11)
      else if (sigmoidal unit)
        Calculate h_j by equation (7) and (8)
      end
    •Backpropagate the error throught the network;
      Especially when multiplication unit,
      Calculate δ by equation (16) and (17)
  Update all the weights.
Until terminal.

3.3 Influence on the Learning Speed For a visual evidence of the achievement by incorporating multiplication in ULNs, we investigated the error surface by multiplication unit and summation unit, respectively. We drew out only one multiplication unit and one summation unit from the hybrid network, respectively and show how different the two units are by their respective error surface(see Fig.2). This is difficult, but possible in simple two-dimensional problems, such as XOR problem. Consider, for example, a network with two inputs, a hidden layer of one unit, and one output node without bias, while fixing all the other parameters to be constant, then the corresponding cost function L of the sigmoidal node as a function of two weights w_1 and w_2 would be:

$$L(w_1, w_2) = \frac{1}{2} [z.f(w_1.h_1 + w_2.h_2) - Target]^2 \cdots\cdots (18)$$

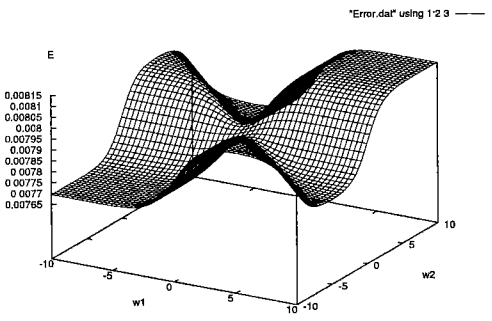
while that of the multiplication unit would be:

$$L(w_1, w_2) = \frac{1}{2} [z.(h_1 - w_1).(h_2 - w_2) - Target]^2 \cdots\cdots (19)$$

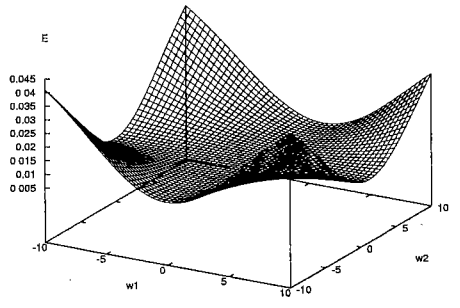
where $f(.)$ is the tangent function $f(h) = \frac{1}{1+e^{-\phi h}}$, and threshold $\theta = 0$.

With random initial values of weights, we can see from Fig.2 that the error surface of the multiplication unit is more smooth than that of summation unit. Cost function L is a 4th order function with respect to the weight parameters which will lead to fast learning even by the same gradient descent method.

In fact, by calculating the partial derivatives of the output with respect to its input of sigmoidal unit and multiplication unit respectively(that of sigmoidal unit described by equation (10) is shown in equation (20), refer to equation (17) for multiplication unit), we can see that even when $|w_{jk}|$, ϕ_k and z_k are set at large values,



(summation unit with sigmoidal function)



(multiplication unit)

Fig. 2. Error surface of the summation unit with sigmoid function and multiplication unit. The weights w_1 and w_2 run from -10 to 10 respectively.

Table 1. Simulation results for the XOR problem.

	Hybrid Network	Standard Network	
Algorithms	BP	BP	Quickprop
Learning Parameters	$\gamma = 0.5$ $\alpha = 0.9$	$\gamma = 0.5$ $\alpha = 0.9$	$\gamma = 0.5$ $\mu = 1.75$ $\alpha = 0.9$
Epochs	2000	2000	2000
Trials	1000	1000	1000
Successes(%)	100%	77.2%	80.2%
Failure(%)	0%	22.8%	19.8%

summation unit has difficulties to make $|\frac{\partial h_k}{\partial h_j}|$ change arbitrarily at any values. But from equation(17), we can see that it is easy to change $|\frac{\partial h_k}{\partial h_j}|$ arbitrarily based on the learning of $w_{j'k}$ and z_k.

$$\frac{\partial h_k}{\partial h_j} = \frac{w_{jk}\phi_k z_k}{2}(1 - (\frac{h_k}{z_k})^2) \cdots\cdots\cdots\cdots\cdots\cdots (20)$$

The model of the proposed multiplication units is supposed to be particularly well suited to situations where the modelled function or problem can be represented by a product function.

4. Simulations and Results

To evaluate the performance of the hybrid ULNs based on the proposed multiplication unit and conventional summation unit, two kinds of benchmark problems were selected as examples. All the problems have speciality that the network used has only one output. But without loss of generality, some of the conclusions may also be extended to other problems.

For the reasons of fair comparison of the network, all the results gained by hybrid ULN as well as pure sigmoidal ULN are under the same conditions except the node type. All the weights are initialized with uniform distribution in [-0.2, 0.2]. Learning algorithm was simplified to highlight the influence of the different node types and activation functions.

1)***XOR Problem:***

As one of the 2-dimensional examples, XOR is one of the traditional benchmarks which is frequently used for historical reasons. Also because it is can be solved by a very small network, so it becomes easier to explain the performance of the unit.

We use a 2-2-1 network to resolve the XOR problem. All the activation functions except that of multiplication unit are sigmoidal function $f(h) = \frac{1}{1+e^{-h}}$. Results shown in Table 1 compare a pure sigmoidal ULN with hybrid ULN with one unit of the hidden layer being multiplication unit. Training set is four general patterns. Learning parameters are named as: learning rate γ, momentum α, and the maximum growth factor μ for the quickprop learning rule. We use Scott Fahlman's quickprop program translated from Common Lisp into C by Terry Regier. Results reported are the average over 1000 runs for each configuration.

Learning curves of both a pure sigmoidal network and the proposed hybrid network with the same configuration are shown in Fig.3, where the curve by dash line is of hybrid network, and that by solid line is of a pure sigmoidal network. From the figure, the influence of

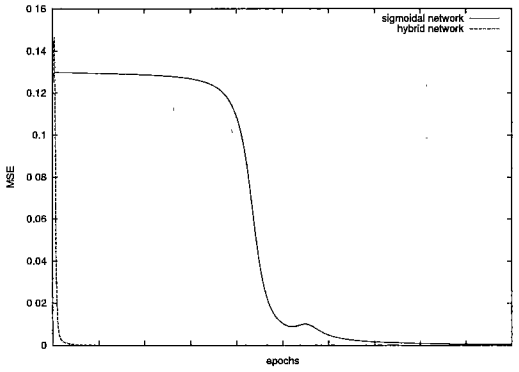


Fig. 3. Comparison of hybrid network and sigmoidal network with respect to learning curves for XOR problem.

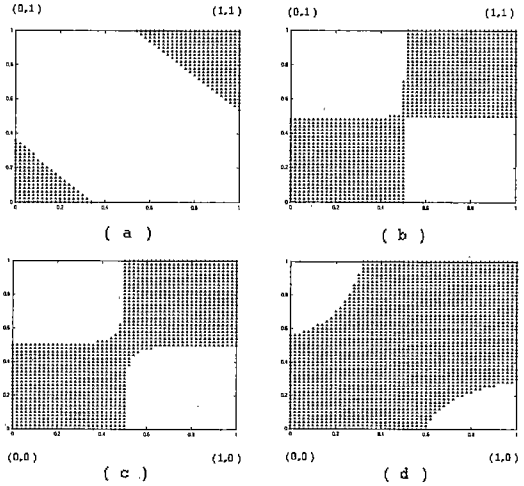


Fig. 4. Classification results of XOR problem.

multiplication unit on the learning rate can be explicitly presented.

Classification results for the XOR problem are shown in Fig.4, where Fig.4(a) shows the result by a standard 2-2-1 sigmoidal network. By changing one of the hidden nodes to multiplication unit, the result becomes like Fig.4(b). Fig.4(c) and Fig.4(d) are the results by 2-2-2-1 hybrid network where we change two or all four hidden nodes into multiplication units.

We found that with the embedding of multiplication units, the XOR problem can be classified by a hyperbola other than by straight lines.

2)***Parity Problem:***

An example used to test the speed of convergence is another benchmark, the parity check problem, which is often used for investigating the nonlinear discrimination capability of a network. Since if the dimension(or bit) changes, the parity will also be changed, so it can be looked as a multi-dimension example. It is a good test of the non-linear mapping and "memorization" capability of the network.

XOR problem, which is equivalent to a 2-bit parity problem, was already tested. Cases of parity problems chosen to be trained vary from 5 to 7 bits. In all cases, experimental results of hybrid universal learning network show better performance than standard feed-

Table 2. Comparisons of simulation results for the parity check problem.

Parity - N	Unit	Epochs	Minimal Hidden Nodes	Averge Sum Square Errors
5	pure	2000	3	0.023
	hybrid	1000	1	0.004
6	pure	6000	4	0.018
	hybrid	1000	1	0.012
7	pure	14000	4	0.021
	hybrid	2000	1	0.007

forward sigmoidal networks. Simulations are mainly centered on the minimal number of the nodes in hidden layer necessary to solve the above three cases by hybrid networks and pure sigmoidal networks, respectively. Results shown in Table 2 for standard networks are different from the results reported in [12], where at least N hidden units are necessary for N bit parity problem. The results show that the needed number of the nodes for the parity problem by hybrid network and pure sigmoidal network is far different.

Many researchers have proposed solutions for this problem using neural networks. Rudy Setiono[13] reported results using traditional sigmoidal neural network that, N-bit parity problem can be solved by a feedforward neural network having $N/2+1$ hidden units if N is even and by a network having $(N+1)/2$ hidden units if N is odd. Stork and Allen[14] showed that the problem can be solved by a network with just two hidden units. They made the assumptions that the network consists of three layers, activation function used is strictly monotonically increasing function and that no direct connection between the input layer and the output layer is allowed. The key to their results is the use of an transfer function of the form:

$$f(h) = \frac{1}{N}(h - \frac{\cos(\pi h)}{\alpha\pi}) \cdots\cdots\cdots\cdots\cdots\cdots (21)$$

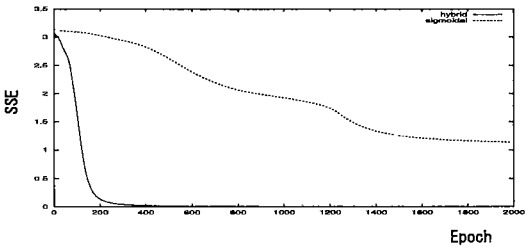
where α is any constant greater than one.

From I.G.Sprinkhuizen-Kuyper and E.J.W.Boers 's paper[15] we can see that if direct connections between the input units and the output unit are allowed, just one hidden unit is need.

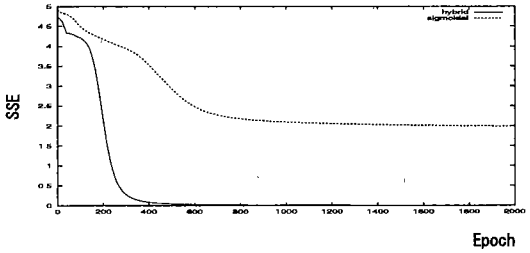
With the same learning parameters, hybrid network shows the capability for resolving the problem with less parameters and training time than traditional sigmoidal neural networks. Results can be compared to the results by Rudy Setiono. All three cases can be solved by N-1-1 hybrid network with only one multiplication hidden unit. So it is very clear that the number of parameters needed by the hybrid network(N*1+1) is less than that of pure sigmoidal networks(for example, for the 6-bit problem, totally $6/2+1=4$ hidden nodes are necessary, and at least $6*4+4=24$ parameters are necessary).

One solution for respective parity(5, 6, 7) problems are shown in Table 3. All the weights and biases were initialized within the range [-0.1, 0.1]. Output h_h of the one hidden unit for parity-6 problem can be represented as:

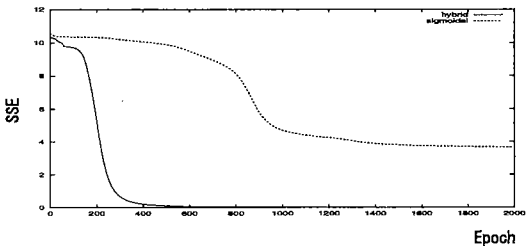
$$h_h = 18.856(h_1 - 0.504)(h_2 - 0.497)(h_3 - 0.495)$$



(parity-5 problem)



(parity-6 problem)



(parity-7 problem)

Fig. 5. Learning curves for parity problems.

$$(h_4 - 0.499)(h_5 - 0.505)(h_6 - 0.495)$$
$$\simeq 18.856\prod_{i=1}^{6}(h_i - 0.5) \cdots\cdots\cdots\cdots\cdots\cdots (22)$$

Sigmoidal function $f(h) = \frac{1}{1+\exp(-h)}$ is used as the activation function of the output unit in this solution. Learning curves of networks used to resolve the three parity problems(5, 6, 7) listed in Table.2 are shown in Fig.5.

Number of training datasets for the three cases is 80 for parity-7 problem, 37 for parity-6 problem and 24 for parity-5 problem respectively. From these figures, we can see that hybrid networks can solve the parity benchmark with faster speed while using less hidden nodes. Since the hybrid networks can find a simple solution for the parity problem which can represent the parity relation with one product term described by equation (22), hence it can generalize the untrained datasets well.

3)***Mirror Symmetry problem:***

Mirror Symmetry problem is to classify input strings as to whether or not they are symmetric about their centres. For the mirror symmetry problem, the output is 1 if the input pattern is exactly symmetrical about its center; otherwise the output is 0. A training case of a symmetry 3 problem can be listed as:

Generally, the problem can be resolved by traditional neural networks with two hidden units. While using

Table 3. Solutions for the parity problem(5, 6, 7).

$Parity-N$	w_{1h}	w_{2h}	w_{3h}	w_{4h}	w_{5h}	w_{6h}	w_{7h}	z_h	w_{ho}	θ_o
5	0.503	0.497	0.502	0.491	0.501			13.516	13.481	0.058
6	0.504	0.497	0.495	0.499	0.505	0.495		18.856	−18.834	0.038
7	0.500	0.494	0.498	0.499	0.502	0.493	0.504	26.766	26.751	0.028

Table 4. Training case for symmetry 3 problem.

x_1	x_2	x_3	y
0	0	0	1
1	0	0	0
0	1	0	1
0	0	1	0
1	1	0	0
1	0	1	1
0	1	1	0
1	1	1	1

hybrid networks, solutions can be found with one hidden multiplication unit. One of the solutions, that is, the output of the hidden multiplication unit can be described as follows:

$$h_h = 1.068(h_1 - 0.493)(h_2 + 6.978)(h_3 - 0.498) + 0.501 \cdots\cdots (23)$$

5. Conclusion

Based on the biological research and practice of pioneers, new type of multiplicative-like units have been already proposed and used in feed-forward neural networks. In this paper, a kind of hybrid universal learning network constructed by the multiplication units and summation units is proposed. We mainly investigated the learning speed by the cooperation of multiplication units with conventional summation units. In addition, the number of multiplication units and the ways of combination are also studied by several simple examples. It seems that multiplication units proposed can be used to improve the performance of traditional sigmoidal neural networks without changing the structure of the network. At the same time, hybrid Universal Learning Networks can be trained by standard back-propagation method as exactly in the same way as pure sigmoidal networks. By combing multiplication units with summation units, the structure of the networks used for certain problems can be simplified significantly while at the same time, the performance of the networks can be improved significantly for some cases.

(Manuscript received March 26, 2002, revised October 29, 2002)

References

(1) K. Hirasawa, X. Wang, J. Murata, J. Hu, and C. Jin: "Universal Learning Network and Its Application to Chaos Control", *Neural Networks*, Vol.13, pp.239-253 (2000)

(2) C. Koch and T. Poggio: "Multiplying with synapses and neurons ", In T.McKenna, J. Davis, and S.F. Zornetzer, Eds., Single Neuron Computation. Boston: Academic Press, pp.315-345 (1992)

(3) C. Koch: Biophysics of Computation. Oxford University Press, NewYork (1999)

(4) J. Hertz, A. Krogh, and R. G. Palmer: Introduction to the Theory of Neural Computation, Addison.Wesley (1991)

(5) M. I. Heywood: "A Framework for Improved Training of Sigma-Pi Networks", *IEEE Trans. Neural Networks*, pp.893-903, Vol.6, No.4, July (1995)

(6) Y.Shin and J. Ghosh: "The Pi-Sigma Network: An Efficient Higher-Order Neural Network for Pattern Classification and Function Approximation", Proceedings of IJCNN, Vol.I, pp.13-18, Seattle, July (1991)

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

(8) K. Hirasawa, J. Murata, J. Hu, and C. Jin: "Universal Learning Network and Its Application to Robust Control", *IEEE Trans. on System, Man, and Cybernetics-PART B*, Vol.30, No.3, pp.419-430 June (2000)

(9) K. Hirasawa, J.Hu, J.Murata, and C.Jin: "A New Control Method of Nonlinear Systems based on Impulse Responses of Universal Learning Networks", *IEEE Trans. on System, Man and Cybernetics - PART B*, Vol.31, pp.362-372 June (2001)

(10) K. Hirasawa, S.Kim, J.Hu, J.Murata, M.Han, and C.Jin: "Improvement of Generalization Ability for Identifying Dynamical Systems by using Universal Learning Networks ", *Neural Networks*, Vol.14, pp.1389-1404 (2001)

(11) D. Li, K. Hirasawa, J. Hu, and J. Murata: "Universal Learning Networks with Multiplication Neurons and its Representation Ability", *in Proc. of IJCNN*, Vol.1, pp.150-155, July (2001)

(12) D.E.Rumelhart, G.E. Hinton, and R.J.Williams: "Learning Internal Representations by Error Propagation", In D.E.Rumelhart and J.L.McClelland, Parallel Distributed Processing, Vol.1: Foundations. Cambrigde, MA: MIT Press (1986)

(13) R.Setiono: "On the solution of the parity problem by a single hidden layer feedforward neural network", Neurocomputing 16 pp.225-235 (1997)

(14) D.G.Stock and J.D.Allen: "How to solve the N-bit parity problem with two hidden units", *Neural Networks* 5 pp.923-926 (1992)

(15) I.G.Kuyper and E.J.W.Boers: "The Error Surface of the simplest XOR Network has no local Minima", *Neural Computation*, Vol.8, Issue 6, pp.1301-1320 August (1996)

Dazi Li (Non-member) She received the B.S and M.S degree in Industrial Control from Beijing University of Chemical Technology, Beijing, China, in 1992 and 1995 respectively. From 1995 to 2000, she worked in Beijing University of Chemical Technology, where she was a Research Associate and then Lecture. Since April 2001, she has been a doctor course student of Kyushu University.

Kotaro Hirasawa (Member) He received the M.S. degree in Electrical Engineering from Kyushu University in 1996. From April 1966, he served in Hitachi Lab. of Hitachi Ltd. and in 1989 he was a vice president of Hitachi Lab.. From August 1991 to November 1992, he served in Omika Factory of Hitachi Ltd.. From December 1992, he was a professor of the Graduate School of information Science and Electrical Engineering, Kyushu University. Since September 2002, he has been a professor of the Graduate School of Information, Production and Systems, Waseda University. Dr. Hirasawa is a member of the Society of Instrument and Control Engineers, a member of IEEE.

Jinglu Hu (Member) He received the B.S degree and the M.S degree in Electronic Engineering from Zhongshan University, China, in 1983 and 1986, respectively, the Ph.D degree in Computer Science and Engineering from Kyushu Institute of Technology, Japan, in 1997. From 1986 to 1993, he worked in Zhongshan University, where he was a Research Associate and then Lecture. Since 1997, he has been a research associate in Kyushu University, Japan. His current research interests are system identification, learning network, and control engineers, a member of the Institute of Electrical Engineers of Japan.

Junichi Murata (Member) He received the M.S and Ph.D degrees in engineering from Kyushu University, Fukuoka, Japan, in 1983 and 1986, respectively. He became a Research Associate and an Associate Professor with the Faulty of Engineering, Kyushu University. He is currently an Associate Professor in the Graduate School of Information Science and Electrical Engineering, Kyushu University. His current research interests are neural networks, self-organizing systems, and their applications to control and identification. Dr. Murata is a member of SICE, ISCIE and IEEJ.