An Incremental Learning of Neural Network with Multiplication Units for Function Approximation

Li, Dazi
Department of Electrical and Electronic Systems Engineering, Kyushu University : Graduate Student

Hirasawa, Kotaro
Graduate School of Information, Production and Systems, Waseda University

Hu, Jinglu
Graduate School of Information, Production and Systems, Waseda University

Wada, Kiyoshi
Department of Electrical and Electronic Systems Engineering, Faculty of Information Science and Electrical Engineering, Kyushu University

https://doi.org/10.15017/1515845

出版情報：九州大学大学院システム情報科学紀要. 8 (2), pp.135-140, 2003-09-26. 九州大学大学院システム情報科学研究院
バージョン：
権利関係：

九州大学大学院
システム情報科学紀要
第8巻 第2号 平成15年9月

Research Reports on Information Science and
Electrical Engineering of Kyushu University
Vol.8, No.2, September 2003

An Incremental Learning of Neural Network with Multiplication Units for Function Approximation

Dazi LI* , Kotaro HIRASAWA**, Jinglu HU** and Kiyoshi WADA***

(Received June 12, 2003)

Abstract: This paper presents a constructive neural network with sigmoidal units and multiplication units, which can uniformly approximate any continuous function on a compact set in multi-dimensional input space. This network provides a more efficient and regular architecture compared to existing higher-order feedforward networks while maintaining their fast learning property. Proposed network provides a natural mechanism for incremental network growth. Simulation results on function approximation problem are given to highlight the capability of the proposed network. In particular, self-organizing process with RasID learning algorithm developed for the network is shown to yield smooth generation and steady learning.

Keywords: Higher order neural networks, Function approximation, Multiplication units, Sigmoidal units, Random search

1. Introduction

Using neural networks to approximate functions is an extremely broad topic. The necessary to learn a function with neural networks is lying in that we have a function that is difficult to compute, or even if the computation is possible, but is very slow. A neural network accepts one or more inputs and produce one or more outputs. It is well known that conventional neural networks are intrinsically function approximators.

More recent results also show that radial basis function networks, recurrent neural networks et. al can be also uniform approximators. Difficulties of these networks in function approximation and mapping lie in that with the increase of the input dimension and the complexity of the mapping, learning process will become more unpredictably long.

Another class of neural network model which is worth stressing is nonlinear model. Nonlinear models of neural networks are certified to have higher performances. The development of nonlinear models of neural networks has been approached by many researchers. The most well-known units that comprise multiplicative synapses perhaps are higher-order neurons(HONs)[1)2)]. It is well known that higher order neural networks perform better than networks of first order. Actually it is well known that multiplications in the unit increase the computation power and capacity of the neural networks. 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 h_o and inputs $h_j, h_k, h_l, \ldots$ can be computed as:

$$h_o = f(w_0 + \sum_j w_j h_j + \sum_{j,k(j\leq k)} w_{jk} h_j h_k + \sum_{j,k,l(j\leq k\leq l)} w_{jkl} h_j h_k h_l + \ldots). \quad (1)$$

where $f(\cdot)$ 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[1)3)].

This paper studies mainly the approximation ability of a polynomial neural network which is mainly constructed by a kind of multiplication units proposed. The remaining of the paper is organized as follows. In section 2, the multiplicative connectionist neural network is formulated, and different combinations of multiplication units as well as summation unit will be investigated. In section 3, to achieve the incremental growth of networks, a ran-

* Department of Electrical and Electronic Systems Engineering, Graduate Student

** Graduate School of Information, Production and Systems, Waseda University

*** Department of Electrical and Electronic Systems Engineering

dom search method is described for training. Although there are many applications of HONNs, few papers are concentrated on function approximation problems, so in Section 4, some simulation results for parity problems and a function approximation problem by constructed HONNs will be given. Section 5 gives conclusions of this paper.

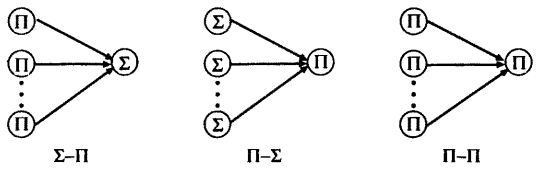


Fig.1 Combinations of Different Node Types.

2. Constructive Neural Networks

2.1 Description of the Model

It is well known that a general sigmoidal unit can be described as[4]:

$$h_j = \sum_i w_{ij} h_i + \theta_j. \tag{2}$$

where h_i is input to node j, w_{ij} is an adjustable parameter from node i to node j, θ_j is the threshold parameter of node j.

Differently from the conventional summation unit described above, instead of dealing with all the inputs by a linear summation, a developed multiplication unit multiplies all the inputs after subtracting an adjustable weight from them. Operation of the proposed multiplication unit for all the inputs h_i is expressed as:

$$h_j = z_j \prod_{i \in JF(j)} (h_i - w_{ij}) + \theta_j, \tag{3}$$

where z_j is the gain parameter of node j. w_{ij} is an adjustable parameter from node i to node j. $JF(j)$ is set of suffixes of nodes connecting to node j.

One problem is to decide the order of the network architecture required to implement the problem because it can directly affect the training time and generalization of the network. The form of a HONN is somewhat similar to the form of a polynomial with inputs. The network order can be identified by the polynomial formed, and when the polynomial includes products of up to n input terms, the network can be termed as a n-order HONN. For the speciality of our multiplication units, if full connections between nodes are expected, higher order of N (number of inputs) can be reached only by one multiplication unit. Differently from higher order networks of sigma-pi units[1] with tensor product as $\prod h_i$, higher order of more than N can be reached by more than one layer of multiplication units.

It is necessary for the incremental procedure to determine the basic structure to start. There are many ways of combining sigmoidal units and multiplication units(see **Fig.1**) in neural networks. Here we take $\sum$-$\prod$-$\prod$ structure as an example to explain the growth procedure of the networks. Because of the connection between the multiplication units, the order of the network will be decided on the basis of the number of input variables in the first hidden layer and the number of units in the layer behind the first hidden layer. The incremental mechanism of the $\sum$-$\prod$-$\prod$ networks was illustrated in **Fig.2**, which begins with a small network size. New units will be added one by one as more precise approximation is demanded considering to avoid constructing an oversized network concurrently(new added nodes are illustrated by shading).

The advantage of the new method is mainly based on the following points:

- Because of the special characteristics of the proposed unit, constructed networks with multiplication units can form higher order terms while eliminating the combinational explosion.

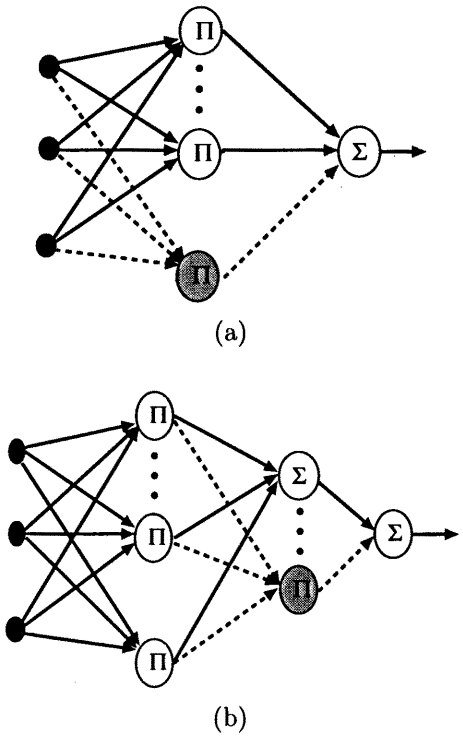


Fig.2 Mechanism for incremental network growth. (a) An incremental network of 3 layers; (b) An incremental network extended to 4 layers.

• The overall network will grow from a small structure, in one study, only one unit or layer can be added so as to avoid the oversized network structure for the problem.

Next we can define the $\sum - \prod$ structure by the proposed multiplication units and conventional sigmoidal units as:

$$h_k = \sum_j w_{jk}\Big(z_j \prod_{i \in JF(j)} (h_i - w_{ij}) + \theta_j\Big) + \theta_k, \quad (4)$$

and if there are more than one hidden layer with multiplication units, by means of Eq.(1) we have:

$$h_l = \sum_k w_{kl}\Big\{z_k \prod_{j \in JF(k)} \Big[\Big(z_j \prod_{i \in JF(j)} (h_i - w_{ij}) + \theta_j\Big) - w_{jk}\Big] + \theta_k\Big\} + \theta_l. \quad (5)$$

3. Learning of the Neural Network with RasID

3.1 Random Search with Intensification and Diversification

The gradient-based schemes such as well-known backpropagation(BP) are well used for neural networks training[5]. However, sometimes it is difficult to get the gradient information of the proposed units, and also the big amount of multiplications makes it easier to trap into local minima by searching the gradient direction. In this paper, we train the networks with multiplication units by using a modified random search method called RasID (Random Search with Intensification and Diversification)[6].

RasID is a kind of random search optimization methods and executes intensified and diversified search in a unified manner using information on success and failure of the past searching. One of the distinguished features of RasID is that the probability density function(PDF) can be changed adaptively based on the past success and failure information of the searching in order to realize the intensified search and diversified search. We introduce a sophisticated PDF $g(\theta_m)$ for generating the random search variable θ_m, defined by

$$g(\theta_m) = \begin{cases} (1 - q_m)\beta_m e^{\beta_m \theta_m} & \text{If } \theta_m \leq 0 \\ q_m \beta_m e^{-\beta_m \theta_m} & \text{If } \theta_m > 0, \end{cases} \quad (6)$$

where $q_m \in [0, 1]$ and β_m are two kinds of parameters used to perform intensification-diversification search. Therefore, it follows that a random search variable θ_m is generated by

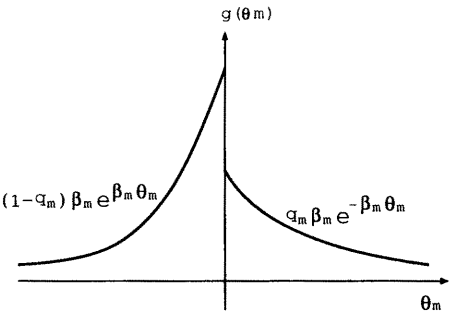


Fig.3 Probability density function for generating random variable θ_m.

$$\theta_m = \begin{cases} \frac{1}{\beta_m} \ln(\frac{z_m}{1-q_m}) & \text{If } 0 < z_m \leq 1 - q_m \\ -\frac{1}{\beta_m} \ln(\frac{1-z_m}{q_m}) & \text{If } 1 - q_m < z_m < 1.0, \end{cases} \quad (7)$$

where z_m's are random values uniformly distributed between 0 and 1.

So, by using RasID, faster computation is expected and escaping from local minima becomes possible. As shown in **Fig.3**, the RasID parameter β_m can be used to control the local search range (the variance of search variable θ_m). The larger the β_m is, the smaller the local search range will be, and vice versa. On the other hand, the parameter q_m can be used to control the search probability in positive or negative direction. The larger the q_m is, the higher the search probability in positive direction is. Typically, when $q_m = 0.5$ there is the same search probability in positive and in negative direction. In this way, intensified and diversified search can be realized by adjusting the parameters β_m and q_m effectively based on the past success-failure information and local information.

Basic idea of RasID is that it continues to iterate the searching in the following way. When there is quite a possibility of finding a better solution around the current one, intensified search is executed near the current solution, and when finding a better solution cannot be expected because of falling into local minima, then diversified search is executed looking for a better solution. Determination of β_m is introduced by

$$\beta_m = \underline{\beta}_m + (\bar{\beta}_m - \underline{\beta}_m)e^{-\phi I} \quad (8)$$

where $\bar{\beta}_m$, ϕ and I are adjustable indexes with initial values $\bar{\beta}_{m0}$, ϕ_0 and I_0, respectively. **Figure 4** shows the relation between β_m and $\bar{\beta}_m$, $\underline{\beta}_m$, ϕ, I.

In the area where it is possible to find a good

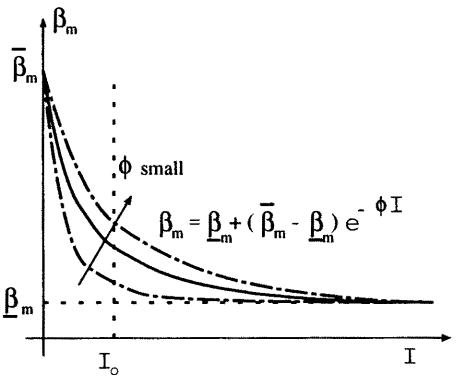


Fig.4 Relation between β_m and $\bar{\beta}_m$, $\underline{\beta}_m$, ϕ, I.

solution locally, an intensified search is performed. The intensified search is performed with keeping the success-failure ratio, P, in a moving window close to a pre-assigned value P_0. This can be realized by fixing I and adapting ϕ according to the relation between P and P_0. The diversified search is carried out in order to escape from the local minimum by fixing $\phi < \phi_{\min}$ and adapting I based on success and failure information of the searching.

In our scheme, the same function will be realized more efficiently by adapting the parameter q_m. Obviously, the local gradient, if available, is useful information for adjusting q_m. Let us first give q_m an initial value of 0.5 and adjust it only when the past search is success in the following way

$$q_m = \begin{cases} \alpha q_m & \text{If } \theta_m(n) < 0 \text{ or } \frac{\partial^+ f}{\partial \theta_m} > 0 \\ q_m & \text{If } \theta_m(n) = 0 \text{ or } \frac{\partial^+ f}{\partial \theta_m} = 0 \\ \alpha q_m + (1-\alpha) & \text{If } \theta_m(n) > 0 \text{ or } \frac{\partial^+ f}{\partial \theta_m} < 0 \end{cases} \tag{9}$$

where $\alpha \in (0, 1]$ is an appropriate value, and $\frac{\partial^+ f}{\partial \theta_m}$ is the ordered derivative of the objective function f with respect to the object variable θ_m of the nth search.

3.2 Constructing HONN with Multiplication Units

For the input-output data set given in the form:

$$(X_i, y_i) = (x_{i1}, x_{i2}, \ldots, x_{iN}, y_i), i = 1, 2, 3, \ldots, I. \tag{10}$$

the incremental process of the constructive networks by RasID can be described as the following steps:

1. Initialization: The network is initialized with 2 layers besides the input layer (that is, a 3 layers network) at first, with one unit in each layer. Output node is a summation unit, and the first hidden node is a multiplication unit. For I training patterns, criterion function of RasID training is defined as the average relative variance(ARV) measure[7][8] of the neural networks:

$$ARV = \frac{\sum_{i=1}^{I}(\hat{f}(X_i) - f(X_i))^2}{\sum_{i=1}^{I}(f(X_i) - \bar{f})^2} \tag{11}$$

where $f(\cdot)$ is the function to be approximated, $\hat{f}(\cdot)$ is the outcome estimated, such as $\hat{f}(X_i)$ is the estimated value with the ith input vector X_i and $\bar{f}$ is the mean value of $f(\cdot)$.

2. Judgement: If the network error defined above is smaller than a pre-specified value, the constructive process will be terminated.

3. Constructing: A new multiplication unit is added in the hidden layer. On the other hand, if there is no significant change in the monitored ARV, that is, the absolute value of the changing rate of $ARV(CR_{ARV}$(see Eq.(12)) is less than some prespecified value, then add a multiplication unit(the mth unit in the network) in a new hidden layer. The added unit receives its inputs from units in the last hidden layer. Comparison will be done after several iterations to decide whether or not the added unit will be accepted. The learning process is iterated to step 2 until the satisfactory estimation is available.

$$CR_{ARV} = \left| \frac{ARV(m-1) - ARV(m)}{ARV(m-1)} \right| \times 100\% \tag{12}$$

where $ARV(m)$ is the training error ARV after the mth unit is added.

4. Stop The constructive process without convergence until the prespecified iteration will be denoted failed and then be terminated.

The number of hidden layers and the number of nodes in one hidden layer decide the order of the neural network being constructed.

The outstanding feature of the constructive algorithm with multiplication units is its high flexibility. Not only the order of the network but also the coefficients of the product terms may vary with the multiplication units located at different layers. Therefore the multiplication units as well as sigmoidal units can be utilized effectively according to the order of the problem.

4. Experiments and Results

Different combinations of multiplication units and summation units have been tried and evaluated

Table 1 Comparisons of simulation results for the parity check problem

order	Epochs		MSE		No. unit	
	MLP	CNN	MLP	CNN	MLP	CNN
5	2000	1000	0.023	0.004	3	1
6	6000	1000	0.018	0.012	4	1
7	14000	2000	0.021	0.007	4	1

for different problems. For example, in the experiments for Boolean function learning, a $\sum$-$\prod$ structure is enough; But for function approximations, the $\sum$-$\prod$-$\prod$ structure was derived out. Simulation results show that this method can solve certain problems better with more compact structures.

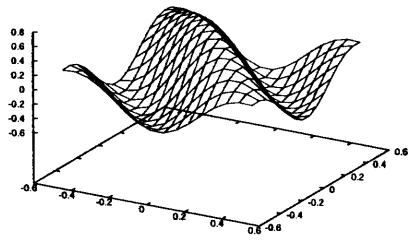
4.1 Parity Problems

Boolean function learning is studied many times in the research of neural networks. Some of the problems, such as parity problems, also act as benchmarks for algorithms of neural networks . A parity problem is a very demanding classification task for neural networks to solve, because the target-output changes whenever a single bit in the input vector changes. This makes generalization difficult, and a general feed-forward neural network which tries to solve parity problems does not always converge easily because of the local minima problem.

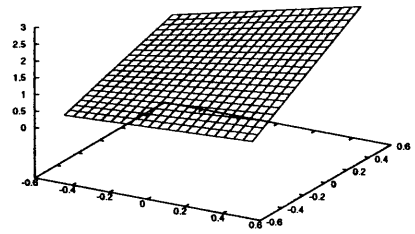
The networks to solve the problems were constructed with one multiplication unit only in the hidden layer and one sigmoidal output unit that uses the sigmoidal activation function: $s(h) = \frac{1}{1+e^{-h}}$. They all can converge very faster than conventional sigmoidal neural networks which need at least 3, 4 and 4 sigmoidal units in the hidden layer respectively for parity 5, 6 and 7 problems. Comparisons of the constructed neural network(CNN) with conventional MLP(trained by fast-BP) for 5, 6 and 7 bits parity problems are shown in **Table 1**. This result can be extended to parity problems of more higher orders, that is, the parity problem of order n can be represented by a network constructed with one multiplication unit in the hidden layer while forming higher order terms until order n.

4.2 Function Approximation

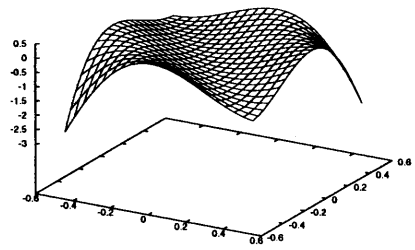
In order to study the approximation ability of the networks constructed with multiplication units, a simulation was executed by using the following two-dimensional function approximation problem. The function $f(h_1, h_2)$ is defined on $-0.5 \leq h_1 \leq 0.5, -0.5 \leq h_2 \leq 0.5$.



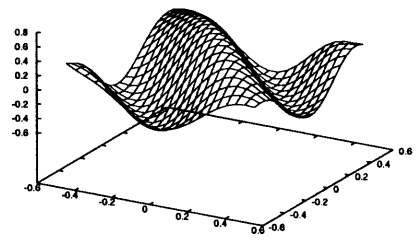
(a) Learning data of f



(b) Output of the unit in N_1



(c) Output of the unit in N_2



(d) Generation result of f

Fig.5 Results for function approximation.

$$f(h_1, h_2) = \frac{1}{2\pi(0.5)^2} \cdot e^{-[(h_1^2+h_2^2)/2(0.5)^2]} \cdot \cos(2\pi(h_1 + h_2)). \quad (13)$$

The training data set contain 256(16 × 16) data patterns, and the testing data set are composed of

$441(21 \times 21)$ data patterns, which are different from the 256 training data set.

Constructive process was executed for the function approximation problem. The network used to learn this function looks like the structure shown in **Fig.2(b)**. The following neural network structure was obtained after training: 4 layers with two inputs and one output: $2{:}N_1{:}N_2{:}1$, output unit is a summation unit with a tangent function, N_1, N_2 obtained for this problem is 6 and 2, respectively. The generated surfaces of the units in different layers along with its original and produced surfaces are shown in **Fig.5**.

Table 2 gives a simple comparison on the generalized performance of the constructed network and conventional sigmoidal network which is trained with fast backpropagation after 200,000 iterations. For training of the constructed network, initial value of weights were initialized within the range[-0.2, 0.2], the parameters of RasID method were set to $\beta_0 = 0.1$, $\beta_1 = 5000$, $\alpha = 0.95$, $\phi_0 = 0.1$, $P_{sf0} = 0.5$, $I_{sfmax} = 100$, $\Delta I_{sf1} = 0.02$, $\Delta I_{sf2} = 1.0$, $c_i = 1.05$, $c_d = 0.995$, etc. For conventional feedforward neural networks, the weights and biases were initialized within the range [-0.1, 0.1], the initial learning rate was set to 0.01. Results show that with more compact structure and less weight parameters, the constructed network with RasID training can achieve better generalization ability and faster learning than ordinary networks.

Table 2 Comparison of performances between constructed network(CNN) and conventional feedforward neural networks(FFNN)

	size of network	number of weights	MSE
CNN	2:6:2:1	35	2.78×10^{-4}
FFNN1	2:20:1	81	3.96×10^{-4}
FFNN2	2:50:1	201	2.27×10^{-4}

5. Conclusion

In this paper, a new constructed neural network by multiplication units and summation units with an efficient random search algorithm has been developed. The proposed network with multiplication units has a potential incremental mechanism to form higher order terms while avoiding the explosion problem of higher order neural networks. It is very flexible to organize different network structures with the proposed multiplication units and conventional summation units. Besides, the capability of the RasID method for training of higher order neural networks was also investigated.

Simulation results for parity problems and a function approximation problem show that this method can solve a certain problems faster with more compact structures. Future work will be focus on improving the success rate of the learning process, which is not competitive because of limitation of the search method as well as the local minimums caused by the multiplication units.

Acknowledgment

This research was partly supported by the 21st Century COE Program "Reconstruction of Social Infrastructure Related to Information Science and Electrical Engineering."

References

1) M. Heywood and P. Noakes : "A Framework for Improved Training of Sigma-Pi Networks", IEEE Trans. Neural Networks, Vol. 6, No. 4, pp. 893-903, 1995.
2) J. Ghosh, and Y. Shin,: "Efficient higher-order neural networks for classification and function approximation". International Journal of Neural Systems, **3** (1992) pp:323-350.
3) L. Spirkovska and M. B. Reid, "Coarse-Coded higher-order neural networks for PSRI Object Recognition", IEEE Trans. Neural Networks, Vol.4, No.2, pp. 276-283, March, 1993.
4) J. Hertz, A. Krogh and R. G. Palmer : Introduction to the Theory of Neural Computation, Addison.Wesley, 1991.
5) R.L. Watrous : "Learning algorithm for connectionist networks: Applied gradient methods of nonlinear optimization", Proc. of IEEE International Conference on Neural Networks, Vol.2, pp.619-628, 1987.
6) J. Hu, K. Hirasawa, and J. Murata, "RasID-Random Search for Neural Network Training", Journal of Advanced Computational Intelligence, Vol.2, No.4, pp.134-141,1998.
7) T.Y. Kwok and D.Y. Yeung, "Constructive algorithms for structure learning in feedforward neural networks for regression problems", IEEE Trans. Neural Networks, Vol. 8, No. 5, pp. 1131-1148, September 1997.
8) A. S. Weigend, B. A. Huberman and D. E. Rumelhart, "Predicting sunspots and exchange rates with connectionist networks", in Nonlinear Modeling and Forecasting, S. Eubank and M. Casdagli, Eds. Redwood City, CA: Addison-Wesley, pp. 395-432, 1992.