A New Learning Method Using Prior Information of Neural Networks

Lu, Baiquan
Venture Business Laboratory Kyushu University

Hirasawa, Kotaro
Department of Electrical and Electronic Systems Engineering, Graduate School of Information Science and Electrical Engineering, Kyushu University

Murata, Junichi
Department of Electrical and Electronic Systems Engineering, Graduate School of Information Science and Electrical Engineering, Kyushu University

Hu, Jinglu
Department of Electrical and Electronic Systems Engineering, Graduate School of Information Science and Electrical Engineering, Kyushu University

https://doi.org/10.15017/1498417

出版情報：九州大学大学院システム情報科学紀要. 4 (1), pp.29-35, 1999-03-26. 九州大学大学院システム情報科学研究科
バージョン：
権利関係：

九州大学大学院
システム情報科学研究科報告
第4巻 第1号 平成11年3月

Research Reports on Information Science and
Electrical Engineering of Kyushu University
Vol.4, No.1, March 1999

A New Learning Method Using Prior Information of Neural Networks

Baiquan LU*, Kotaro HIRASAWA**, Junichi MURATA** and Jinglu HU**

(Received December 21, 1998)

Abstract: In this paper, we present a new learning method using prior information for three-layer neural networks. Usually when neural networks are used for identification of systems, all of their weights are trained independently, without considering their inter-relation of weights values. Thus the training results are not usually good. The reason for this is that each parameter has its influence on others during the learning. To overcome this problem, first, we give exact mathematical equation that describes the relation between weight values given a set of data conveying prior information. Then we present a new learning method that trains the part of the weights and calculates the others by using these exact mathematical equations. This method often keeps a priori given mathematical structure exactly during the learning, in other words, training is done so that the network follows predetermined trajectory. Numerical computer simulation results are provided to support the present approaches.

Keywords: Prior information, Neural network learning, Part parameter learning, Exact mathematical structure

1. Introduction

ARTIFICIAL neural networks(ANN's) have been used widely in many application areas in recent years. Most applications use feedforward ANN's and error backpropagation (EBP) training algorithm. There are three questions of great concern in the practical training of backpropagation networks, of which one to find training algorithms with fast convergence. Another vital question arises from the existence of local minima of backpropagation error surfaces. 3rd question is that relation of structure of neural networks with objective plant is not clear. Many promising results are reported [1)5)], but all these training algorithm in general assume that objective plant is often identified as a black-box, so it is difficult to use prior information in learning. Recently, new neural networks using prior information are many reported too [6)7)8)10)11)12)14)]. In these literatures,many approaches have been proposed to incorporate prior information,which can be roughly grouped into two kinds. The first one is the methods to incorporate prior information in neural network constructions. Bayesian neural network is one of the typical examples, which is based on a Bayesian estimation using information theory. Another typical example is fuzzy neural network which expresses prior information in IF-THEN rules and use them in the networks. In such approaches, it is difficult to incorporate prior information that is described by a set of explicit mathematical equations. The second one is the methods where prior information of neural networks are considered as a restriction that is usually added to the objective function [16)]. However, this kind of method does not make clear the relation between prior information and neural network weights. In this paper, we present a new method for three-layered neural network learning, in which not only the prior information expressed in a set of explicit mathematical equations are used,but also the relation between prior information and weights of neural networks is made clear.

Usually when neural networks are used as models for system identification, all weights in the network are trained together, without considering the relations among them. Conventional learning algorithms work as parallel calculations in some levels. However, the parallel calculation capability may become weak when the parameters in the network have leeway for identifying certain system. This leads to a poor training effectiveness because each parameter has its influence on others during the learning. For example, in the Steepest-descent method, each variable is not adjusted along coordinated fashion but along its own Steepest-descent direction. Furthermore, most of existing learning methods are based on local information in their learnings, so that they are easily to be stuck at a local minimum. Considering the information concerning overall situation in the learning may be a good way to solve the problem. For example, if we know

* Venture Business Laboratory Kyushu University
** Department of Electrical and Electronic Systems Engineering

some desired results of trained neural networks, we may use this information to improve the learning efficiency.Therefore, in this sense, conventional learning based on steepest-descent method use the local optimization technique rather than the global optimization technique. We here will study in two cases. One is that partial differentiation of teaching function with respect to inputs are assumed to be known. The other is that partial differentiation of teaching function with respect to inputs are assumed to be unknown. For a three-layered neural network, we found that weights of second-layer can be expressed by weights of first-layer in exact mathematical equations under certain conditions. We will present a new learning method based on the use of these exact mathematical equations, in which parts of parameter are learned from data, while others are calculated by using those exact mathematical equations. The proposed method not only keeps prior exact mathematical structure which include the information concerning overall situation in a coordinated fashion, but also uses local information such as error back-propagation (EBP) etc. in its learning. That is, the training is carried out based on a given trajectory. It can also make the mutual influence among the weights small and thus speed up the learning.

The remainder of paper is organized as follows. In Section 2, we will introduce in detail that a relation of parameters in neural networks, and give a exact mathematical equations. In Section 3, we propose a new learning method which use prior information of neural networks. In Section 4, simulation results are presented by this new method, the conclusions are given in Section 5.

2. Relation between Parameters of Neural Networks and Teaching Function

In this section, we will study relation among the weights in two cases. One case is that partial derivative of teaching function $f(x_1, x_2, \cdots, x_n)$ for a variate $x_i, i = 1, 2, \cdots, n$ is known, the other is to the contrary, the derivative is not known. In the first case, we study the relation between the weights from the first layer to the second layer and the weights from the second layer to the third layer and the relation among the weights from the first layer to the second layer. In the second case, we give relation of weights by using neural network outputs at some special points.

As any mapping of n-input-one-output can be approximated by a neural network with three layers [15] for given teaching function $f(x_1, x_2, \cdots, x_n)$, there is an integer m which satisfies the follows approximation equation with very small approximate error.

$$f(x_1, x_2, \cdots, x_n) = \sum_{k=1}^{m} \alpha_k \varphi(W^T X + \theta_k) \tag{1}$$

where α_k and W^T are weights from the second layer to the third layer and weights from the first layer to the second layer respectively, $X = (x_1, x_2, x_3, \cdots, x_n) \in R^n$ is input vector of the neural networks. Let $\theta_k = 1$, φ is node function such as sigmoid function. We assume $f(x_1, x_2, \cdots, x_n) \in C^m$, and let some coefficients of each term of Taylor expansion of both sides of (1) equal at $x_1 = x_2, \cdots, x_n = 0$, we have

$$\begin{aligned} f(0, 0, \cdots, 0) &= \sum_{k=1}^{m} \alpha_k \varphi(1) \\ f^{(j)}_{x_{i1}}(0, 0, \cdots, 0) &= \sum_{k=1}^{m} \alpha_k W^j_{i1,k} \varphi^{(j)}(1) \\ f^{(j+1)}_{x_{i1}, x^j_{i2}}(0, 0, \cdots, 0) &= \sum_{k=1}^{m} \alpha_k W^j_{i2,k} W_{i1,k} \varphi^{(j+1)}(1) \end{aligned} \tag{2}$$

where $f^{(j+1)}_{x_{i1}, x^j_{i2}}(0, 0, \cdots, 0)$ is the j_{th} order partial derivative with respect to x_{i2} after the first partial derivative with respect to x_{i1}, j=0,1,2,$\cdots$, i=1,2,$\cdots$, x_i is the i_{th} input of neural network, and m is the number of hidden layer in the neural network.

Equation (2) can be written by matrix as follows

$$A_1 B_1 = C_1 \tag{3}$$

where

$$A_1 = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ w_{i1,1} & w_{i1,2} & \cdots & w_{i1,m1} \\ \vdots & \vdots & \cdots & \vdots \\ w^{m1}_{i1,1} & w^{m1}_{i1,2} & \cdots & w^{m1}_{i1,m1} \end{bmatrix}, B_1 = \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_{m1} \end{bmatrix},$$

$$C_1 = \begin{bmatrix} \frac{f(0,0,\ldots,0)}{\varphi(1)} - \sum_{k=m1+1}^{m} \alpha_k \\ \frac{f^{(1)}_{x_{i1}}(0,0,0,\ldots,0)}{\varphi^{(1)}(1)} - \sum_{k=m1+1}^{m} \alpha_k w_{i1,k} \\ \vdots \\ \frac{f^{(m1)}_{x_{i1}}(0,0,0,\ldots,0)}{\varphi^{(m1)}(1)} - \sum_{k=m1+1}^{m} \alpha_k w^{m1}_{i1,k} \end{bmatrix},$$

$$A_2B_2 = C_2, \tag{4}$$

where

$$A_2 = \begin{bmatrix} \alpha_1 & \alpha_2 & \cdots & \alpha_{m1} \\ \alpha_1 w_{i2,1} & \alpha_2 w_{i2,2} & \cdots & \alpha_{m1} w_{i2,m1} \\ \vdots & \vdots & \cdots & \vdots \\ \alpha_1 w_{i2,1}^{m1} & \alpha_2 w_{i2,2}^{m1} & \cdots & \alpha_{m1} w_{i2,m1}^{m1} \end{bmatrix},$$

$$B_2 = \left[w_{i1,1}, w_{i1,2}, \cdots, w_{i1,m1} \right]^T,$$

$$C_2 = \begin{bmatrix} \frac{f_{x_{i1}}^{(1)}(0,0,\cdots,0)}{\varphi^{(1)}} - \sum_{k=m1+1}^{m} \alpha_k w_{i1,k} \\ \frac{f_{x_{i1}}^{(2)}(0,0,\cdots,0)}{\varphi^{(2)}} - \sum_{k=m1+1}^{m} \alpha_k w_{i1,k} w_{i2,k} \\ \vdots \\ \frac{f_{x_{i1},(xi2)^{m1}}^{(m1+1)}(0,0,\cdots,0)}{\varphi^{(m1+1)}} - \sum_{k=m1+1}^{m} \alpha_k w_{i1,k} w_{i2,k}^{m1} \end{bmatrix},$$

where $m1 \leq m$, $i1$=1,2,$\cdots$,n, $i2$=1,2,3,$\cdots$,n. For clarity, we assume $m1$=m in the following. Since the inverse of matrix A1 exists in (3) if and only if (5) is satisfied for $i1_{th}$ input of neural network, otherwise we can decrease m1 until the inverse of matrix A1 exists. Here we assume that inversion of matrix A1 exists for m1=m ,then we have (6).

$$w_{i1,k1} \neq w_{i1,k2}, \tag{5}$$

where k1$\neq$ k2, k1=1,2,$\cdots$,m,k2=1,2,$\cdots$,m.

$$B_1 = A_1^{-1} C_1 \tag{6}$$

From (6), we can know that B_1 can be expressed by $w_{i1,k1}$, $k1$=1,2,$\cdots$,m, in other words, weights from the second layer to the third layer can be expressed by weights from the first layer to second layer as shown in the **Fig.1**. This is import for learning, we can use this for help learning of networks.

We have studied that relation between weights from the second layer to the third layer and weights from the first layer to second layer as mentioned above, now we study a relation of among weights from the first layer to the second layer.

If (5) and (8) are satisfied for i_{th} input, then we get (8) from (4)

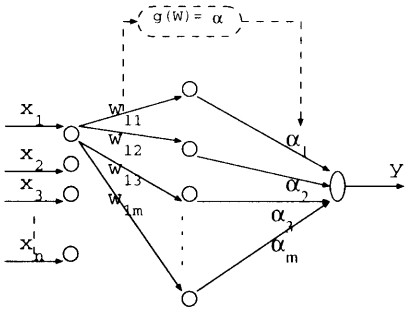

Fig.1 the relation between weights from the second layer to the third layer, g(W) is a function where express relation of the above

$$\alpha_k \neq 0, k = 1, 2, 3, ..., m. \tag{7}$$

$$B_2 = A_2^{-1} C_2 \tag{8}$$

From (8), we get that $w_{i1,k1}$ can be expressed by $w_{i2,k2}$ for i1 $\neq$ i2, for given α_i, i=1,2,$\cdots$,m, under the condition of (5) and (8). α_i for i=1,2,,...,m, is not usually equal to zero, otherwise i_{th} unit of hidden layer can be removed.

Now we discuss a condition that (5) is not satisfied, equation as $w_{i,k1} = w_{i,k2}$ have one at the least for every input,$k1 \neq k2$. For simplicity, we assume $w_{1,1} = w_{1,k(2)}$ for 1st input, $w_{2,k(3)} = w_{2,k(4)}$ for 2nd input, $\cdots$, $w_{n,k(2n-1)} = w_{n,k(2n)}$ for n_{th} input, if $w_{i,k(2i)} = w_{i,k(2i-1)}$, we have the following by (5).

$$A_3 B_3 = C_1 \tag{9}$$

where

$$A_3 = \begin{bmatrix} 1 & 1 & \cdots & 1 & 1 & \cdots & 1 \\ w_{i,1} & w_{i,2} & \cdots & w_{i,k(2i-1)} & w_{i,k(2i+1)} & \cdots & w_{i,m} \\ \vdots & \vdots & \cdots & \vdots & \vdots & \cdots & \vdots \\ w_{i,1}^{m-1} & w_{i,2}^{m-1} & \cdots & w_{i,k(2i-1)}^{m-1} & w_{i,k(2i+1)}^{m-1} & \cdots & w_{i,m-1}^{m} \end{bmatrix}$$

$$B_3 = \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_{k(2i-1)} + \alpha_{k(2i)} \\ \vdots \\ \alpha_m \end{bmatrix}$$

$i = 1, 2, 3, \cdots, n.$

We get equation similar to (6) as follows.

$$B_3 = A_3^{-1} C_1 \tag{10}$$

Form (10), we can know that α_i and $\alpha_{k(2j-1)} + \alpha_{k(2j)}$ can be expressed by $w_{j,i1}$ for i=1,2,$\cdots$,m, i1=1,2,$\cdots$,m, j=1,2,$\cdots$,n. For $\alpha_{k(2j-1)}$ and $\alpha_{k(2j)}$. Because weights from the second layer to the third layer can not be expressed by weights from the first layer to second layer, if and only if k1=k3=k5=,...,=k(2n-1) and k2=k4=,...,=k(2n), With this case, we can remove this node.

So far, we discussed relation among weights under the condition that the finite derivative of $f(x_1, x_2, \cdots, x_n)$ for $x_i, i = 1, 2, \cdots, n$ is known. Now we discuss the case where the finite derivative of $f(x_1, x_2, \cdots, x_n)$ for $x_i, i = 1, 2, \cdots, n$,is not known. Suppose we have some cardinal points of X where the network output should by exactly equal to the value of the teaching function $f(x_1, x_2, \cdots, x_n)$. Such cardinal points can be origin, points that give maximal values, or points corresponding to the minimal value of $f(x_1, x_2, \cdots, x_n)$. Our requirement on one of these points $X(i) = (x_1(i), x_2(i), \cdots, x_n(i))$,is written as

$$f(X(i)) = \sum_{k=1}^{m} \alpha_k \varphi(W^T X(i) + \theta_k) \tag{11}$$

This is another expression of prior information. From (11) or (3) and (4), we know that if all weights learn independently by descent method, then (11) or (3) and (4) are hardly to be guaranteed at every learning step, therefore learning result is not often good. In the following section, we present a new learning method that always keeps prior exact mathematical structure, such as (11) or (3) and (4), in other words, parts of parameters are trained by descent methods, and the others are calculated to give the required trajectories. for example ,(11) or (3) and (4).

3. Learning Using Prior Information

In this section, we will discuss how to use the prior information for learning of neural networks. We assume that the objective function be defined as the following

$$E(W, \boldsymbol{\alpha}) = \frac{1}{2} \sum_{k=1}^{P} [y_k - f(X(k))]^2, \tag{12}$$

where y_k and $f(x_1(k), x_2(k), ..., x_n(k))$ are k_{th} desired outputs and k_{th} actual output respectively, P is total number of train data, W a vector constituting of the weight from first layer to the second layer, $\boldsymbol{\alpha}$ a vector constituting of weights from the second layer to the third layer. Now first, we study and give learning method incorporating the prior constraints (3) and (4), then give learning method for (11). We know, since we assumed m1=m, that the right sides of equations (3) and (4) are constant with no relation to parameters of neural networks for a given function. If we differentiate and linearize the both sides of (3) and (4), then under the condition of (5) and (8), we have

$$\Delta \boldsymbol{\alpha} = T^{(i)} \Delta W_i, \tag{13}$$

$$\Delta W_{i1} = Q^{(i1,i2)} \Delta W_{i2}, \tag{14}$$

where

$$\Delta \boldsymbol{\alpha} = [\Delta\alpha_1, \cdots, \Delta\alpha_m]^T$$

$$\Delta W_i = [\Delta w_{i,1}, \cdots, \Delta w_{i,m}]^T$$

$$T^{(i)} = -(A1)^{-1} \begin{bmatrix} 0 & 0 & \cdots & 0 \\ \alpha_1 w_{i,1} & \alpha_2 w_{i,2} & \cdots & \alpha_m w_{i,m} \\ \vdots & \vdots & \cdots & \vdots \\ m\alpha_1 w_{i,1}^{m-1} & m\alpha_2 w_{i,2}^{m-1} & \cdots & m\alpha_m w_{i,m}^{m-1} \end{bmatrix},$$

$$Q^{(i1,i2)} = -Q1^{(i2)}(Q2^{(i1,i2)} T^{(i2)} + Q3^{(i1,i2)}),$$

$$Q1^{(i2)} = \begin{bmatrix} \alpha_1 & \alpha_2 & \cdots & \alpha_m \\ \alpha_1 w_{i2,1} & \alpha_2 w_{i2,2} & \cdots & \alpha_m w_{i2,m} \\ \vdots & \vdots & \cdots & \vdots \\ \alpha_1 w_{i2,1}^m & \alpha_2 w_{i2,2}^m & \cdots & \alpha_m w_{i2,m}^m \end{bmatrix}^{-1},$$

$$Q2^{(i1,i2)} = \begin{bmatrix} w_{i1,1} & w_{i1,2} & \cdots & w_{i1,m} \\ w_{i1,1} w_{i2,1} & w_{i1,2} w_{i2,2} & \cdots & w_{i1,m} w_{i2,m} \\ \vdots & \vdots & \cdots & \vdots \\ w_{i1,1} w_{i2,1}^m & w_{i1,2} w_{i2,2}^m & \cdots & w_{i1,m} w_{i2,m}^m \end{bmatrix},$$

$$Q3^{(i1,i2)} = \begin{bmatrix} 0 & 0 & \cdots & 0 \\ 1 & 1 & \cdots & 1 \\ \vdots & \vdots & \cdots & \vdots \\ mw_{i2,1}^{m-1} & mw_{i2,2}^{m-1} & \cdots & mw_{i2,m}^{m-1} \end{bmatrix}$$

$$\begin{bmatrix} \alpha_1 w_{i1,1} & 0 & \cdots & 0 & 0 \\ 0 & \alpha_2 w_{i1,2} & 0 & \cdots & 0 \\ \vdots & \vdots & \cdots & \cdots & \vdots \\ 0 & \cdots & \cdots & 0 & \alpha_m w_{i1,m} \end{bmatrix},$$

where i1 $\neq$ i2. $\Delta\boldsymbol{\alpha}$ and ΔW_i are small.
By Taylor expansion at W and $\boldsymbol{\alpha}$, we have $\Delta E(W,\boldsymbol{\alpha}) = \frac{\partial E}{\partial W}\Delta W + \frac{\partial E}{\partial \boldsymbol{\alpha}}\Delta\boldsymbol{\alpha} + \cdots$

$$= \sum_{i=1}^{m}\sum_{k=1}^{n}\frac{\partial E}{\partial W_{i,k}}\Delta W_{i,k} + \sum_{k=1}^{n}\frac{\partial E}{\partial \alpha_k}\Delta\alpha_k + \cdots \quad (15)$$

where m, n are the number of input units, the number of hidden units respectively. Furthermore if (5) is satisfied for $i2_{th}$ input unit, then we have

$$\begin{aligned}&\Delta E(W,\boldsymbol{\alpha}) = \\ &\textstyle\sum_{i\neq i2}^{m}\sum_{k=1}^{n}\frac{\partial E}{\partial W_{i,k}}(\sum_{k1}^{n}Q_{k,k1}^{(i,i2)}\Delta W_{i2,k1}) \\ &+\textstyle\sum_{k=1}^{n}\frac{\partial E}{\partial \alpha_k}(\sum_{k1}^{n}T_{k,k1}^{(i2)}\Delta W_{i2,k1}) \\ &+\textstyle\sum_{k1=1}^{n}\frac{\partial E}{\partial W_{i2,k1}}\Delta W_{i2,k1} + \cdots \\ &= \textstyle\sum_{k1}^{n}(\sum_{i\neq i2}^{m}\sum_{k=1}^{n}\frac{\partial E}{\partial W_{i,k}}Q_{k,k1}^{(i,i2)} \\ &\quad + \sum_{k=1}^{n}\frac{\partial E}{\partial \alpha_k}T_{k,k1}^{(i2)} + \frac{\partial E}{\partial W_{i2,k1}})\Delta W_{i2,k1} + \cdots, \quad (16)\end{aligned}$$

According to the gradient descending method, we get a learning algorithm as follows

$$\Delta w_{i2,k1} = -\eta\left(\sum_{i\neq i2}^{m}\sum_{k=1}^{n}\frac{\partial E}{\partial W_{i,k}}Q_{k,k1}^{(i,i2)} + \sum_{k=1}^{n}\frac{\partial E}{\partial \alpha_k}T_{k,k1}^{(i2)} + \frac{\partial E}{\partial W_{i2,k1}}\right), \quad (17)$$

where $k1 = 1, 2, ..., m$, η is the learning rate, i2 is the number of inputs which satisfy (5). Weights can be adjusted by (13), (14) and (17). First, we get $\Delta w_{i2,k1}$ by using (17), next get $\Delta w_{i1,k1}$ by using (14), then get $\Delta\boldsymbol{\alpha}$ by using (13).

Now we discuss a learning method using (11) with given L cardinal point data (without loss of generality, we let L$\leq$ m), we have

$$\begin{aligned} f(x_1(1), x_2(1), \cdots, x_n(1)) &= \textstyle\sum_{k=1}^{m}\alpha_k\varphi(1) \\ &\vdots \\ f(x_1(L), x_2(L), \cdots, x_n(L)) &= \textstyle\sum_{k=1}^{m}\alpha_k\varphi(L)\end{aligned} \quad (18)$$

where $\varphi(i) = \varphi(W^T X(i) + 1)$. If we differentiate and linearize the both sides of (18), then we have

$$\begin{aligned} &\textstyle\sum_{k=1}^{m}\varphi(1)\Delta\alpha_k + \sum_{k=1,j=1}^{m,n}\alpha_k\dot{\varphi}(1)x_j\Delta w_{j,k} = 0 \\ &\vdots \\ &\textstyle\sum_{k=1}^{m}\varphi(1)\Delta\alpha_k + \sum_{k=1,j=1}^{m,n}\alpha_k\dot{\varphi}(1)x_j\Delta w_{j,k} = 0\end{aligned} \quad (19)$$

where $\Delta w_{j,k}, \Delta\alpha_k, k = 1, 2, \cdots, m, j = 1, 2, \cdots, n$ are small, $\dot{\varphi}$ is the first order derivative of φ.
Among $\Delta w_{j,k}$ and $\Delta\alpha_k$, L parameters are chosen so that an inverse of their coefficient matrix exists and these parameters can be expressed by the others in (19). If the inverse matrix is not exist, then we decrease L until the inverse matrix exists. For simplicity, we assume that chosen L parameters are $\alpha_k, k = 1, 2, \cdots$,L, then we have

$$AB = -CD, B = FD, F = -A^{-1}C, \quad (20)$$

where

$$A = \begin{bmatrix} \varphi(1) & \varphi(1) & \cdots & \varphi(1) \\ \varphi(2) & \varphi(2) & \cdots & \varphi(2) \\ \vdots & \vdots & \cdots & \vdots \\ \varphi(L) & \varphi(L) & \cdots & \varphi(L) \end{bmatrix}, B = \begin{bmatrix} \Delta\alpha_1 \\ \Delta\alpha_2 \\ \vdots \\ \Delta\alpha_L \end{bmatrix},$$

$$C = \begin{bmatrix} \varphi(1) & \cdots & \varphi(1) & \dot{\varphi}(1)x_1(1) & \cdots & \dot{\varphi}(1)x_n(1) \\ \varphi(2) & \cdots & \varphi(2) & \dot{\varphi}(2)x_1(2) & \cdots & \dot{\varphi}(2)x_n(2) \\ \vdots & \cdots & \cdots & \cdots & \cdots & \vdots \\ \varphi(L) & \cdots & \varphi(L) & \dot{\varphi}(L)x_1(L) & \cdots & \dot{\varphi}(L)x_n(L) \end{bmatrix},$$

$$D = [\Delta\alpha_{L+1}, \Delta\alpha_{L+2}, \cdots, \Delta\alpha_m, \Delta w_{1,1}, \cdots, \Delta w_{n,m}]^T,$$

From (15),we can get (21) by (20).

$$\begin{aligned}&\Delta E(W,\boldsymbol{\alpha}) = \\ &\sum_{i1,i2}^{n,m}(\frac{\partial E}{\partial w_{i1,i2}} + \sum_{k=1}^{L}\frac{\partial E}{\partial \alpha_k}F_{k,(i1*m+i2+m-l)})\Delta w_{i1,i2} \\ &+ \sum_{j=L+1}^{m}(\frac{\partial E}{\partial \alpha_j} + \sum_{k=1}^{L}\frac{\partial E}{\partial \alpha_k}F_{k,(j-L)})\Delta\alpha_j + , \cdots\end{aligned} \quad (21)$$

According to the gradient descending method, we get the following algorithm for learning.

$$\Delta w_{i1,i2} = -\eta \left(\frac{\partial E}{\partial w_{i1,i2}} + \sum_{k=1}^{L} \frac{\partial E}{\partial \alpha_k} F_{k,(i1*m+i2+m-l)} \right)$$

$$\Delta \alpha_j = -\eta \left(\frac{\partial E}{\partial \alpha_j} + \sum_{k=1}^{L} \frac{\partial E}{\partial \alpha_k} F_{k,(j-L)} \right), \qquad (22)$$

where j=L+1,$\cdots$,m,η is the learning rate. Weights of neural networks except for $\Delta\alpha_j, j = 1, 2, \cdots, L$, can be adjusted by (22), while $\alpha_j, j = 1, 2, \cdots, L$ can be adjusted by (20).

In the above, we have got learning method by using prior information. This means that part of parameters are adjusted by the learning method which uses local information, other part of parameters are adjusted by exact calculation which uses global information. With more accelerating learning, we as long as change (22), while (20) is not changed, such as moment method. A procedure of parameter adjustment based on the above method is the following:

1) choose L or m1 parameters among weights of neural networks so that they satisfies (5) or (20) if possible, otherwise decrease L or m1 and again choose the parameters. Weights of neural networks are initialized by random, except for parameters number L or m1, since parameters number L or m1 can be obtained from (18) or (3) and (4).
2) part of parameters (except for parameters number L or m1) are adjusted by (17) or (22), other part of parameters are adjusted by (20) or (13) and (14).
3) choose L or m1 parameters among weights of neural networks so that they satisfies (5) or (20) if possible, otherwise decrease L or m1 and again choose the parameters, then go to step 2), until error function is changed to satisfactory value.

4. Simulation

In this section, the simulations are carried out for identification of a dynamic plant in order to verify the above idea. In this example, the plant is assumed to take the from :$y_p(k+1) = P[y_p(k), y_p(k-1), y_p(k-2), u(k), u(k-1)]$ where the unknown function P has the form:

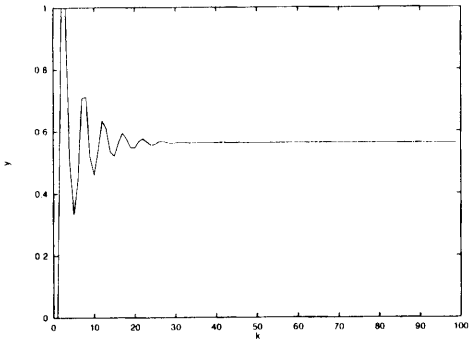

Fig.2 a plot of simulation function

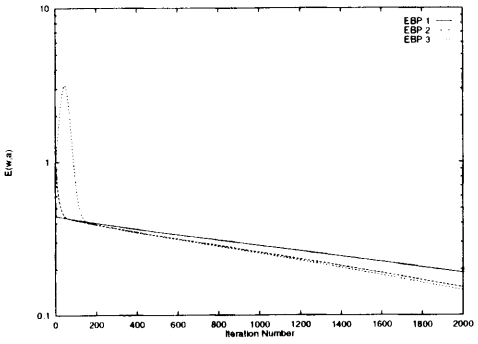

Fig.3 plot 1,2,3 are conventional EBP where learning rate is 0.0012,0.0015,0.00155 respectively

$$P[x_1, x_2, x_3, x_4, x_5] = \frac{x_1 * x_2 * x_3 * x_5 (x_3 - 1) + x_4}{1 + x_3^2 + x_2^2}, \qquad (23)$$

In the condition of $u(k) = 1, k > 0$, the plot of $y_p(k+1)$ is shown in **Fig.**2, where the number of samples is 100. Neural network is formed as 5-60-1, node function is $f(x) = (1 - e^{-x})/(1 + e^{-x})$, the error function is (12). Because points of k=1,2,5,7,10,12,80 is used as special points of the plot, L=7. initial value of weights from the first layer to the second layer and from second to third layer except for $\Delta\alpha_i (i = i_1, i_2, i_3, i_4, i_5, i_6, i_7)$ are generated from the uniform distribution (0.5,0) and (0.2,0) respectively, and $\Delta\alpha_i (i = i_1, i_2, i_3, i_4, i_5, i_6, i_7)$ are calculated by(20). In this simulation, i of $\Delta\alpha_i$ is generated from the uniform distribution (1,60) in every step in order to make (20) holds. We take 3 different values of learn-

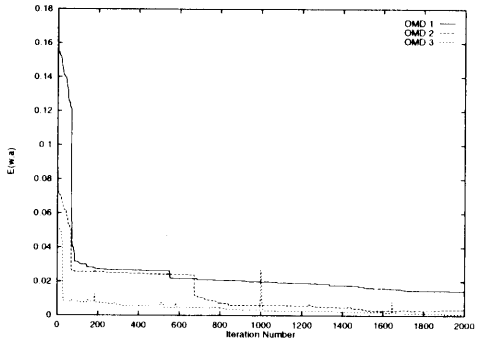

Fig.4 plot 4,5,6 are proposed method where learning rate is 0.00001,0.00002,0.00005 respectively

ing rates $\eta = 0.00001, 0.00002, 0.00005$ and compare with error backpropagation (EBP) algorithm with learning rates $\eta = 0.0012, 0.0015, 0.00155$. The plots of error function are shown in **Fig.3** and **Fig. 4**. We are easily known from plots that learning of neural network is much faster with a small identification error and jumps down sometimes. This is produced by $\Delta\alpha_i$ (i$\leq$ L), The results of simulation indicate that proposed method is very useful.

5. Conclusion

In this paper, a set of formulas have been introduced for the relation of weights of neural networks and learning method which is done according to given learning trajectories using prior information of neural networks with three layers. The proposed method can make each other's influence of weights as small as possible and shows good performance in the training. It has the following characteristics : prior information is effectively used and is much faster than the conventional EBP method. Its effectiveness and applicability have been demonstrated by computer simulations.

References

1) Hyun Myung and Jong-Hwan Kim, "Time-Varying Two-phase Optimization and Its Application to Neural-networks Learning " *IEEE TRANSACTIONS ON NEURAL NETWORKS*, VOL.8,NO.6,NOVEMBER 1997.

2) Tin-Yan Kwok and Dit-Yan Yeung, "Objective Functions for Training New Hidden Units in Constructive Neural Networks " *IEEE TRANSACTIONS ON NEURAL NETWORKS*, VOL.8,NO.5,SEPTEMBER 1997.

3) Xin Yao,Yong Liu, "A New Evolutionary system for Evolving Artificial Neural Networks " *IEEE TRANSACTIONS ON NEURAL NETWORKS*, VOL.8,NO.3,MAY 1997.

4) Xiao-Hu Yu,Gao-an Chen,Shi-Xin Cheng,"Dynamic learning Rate Optimization of the Backpropagation Algorithm " *IEEE TRANSACTIONS ON NEURAL NETWORKS*, VOL.6,NO.3,MAY 1995.

5) Gordon Lightbody and Georgre W.Irwin,"Nonlinear Control Structures Based on Embedded Neural System Models " *IEEE TRANSACTIONS ON NEURAL NETWORKS*, VOL.8,NO.3,MAY 1997.

6) Daniel T and Jenq-Neng Hwang, "Solving Inverse Problem by Bayesian Neural Network Iterative Inversion with Ground Truth Incorporation ," *IEEE TRANSACTIONS ON SIGNAL PROCESSING*, VOL.45,NO.11,NOVEMBER 1997.

7) Mehdi Mostaghimi, "Bayesian Estimation of a Decision Using Information Theory ," *IEEE TRANSACTIONS ON SYSTEM,MAN,AND CYBERNETICS -PART A :SYSTEMS AND HUMANS*, VOL.27,NO.4,JULY 1997.

8) D.J.C MacKay,"Bayesian interpolation ," *Neural compute.*, VOL.4,NO.3,pp415-447,May 1992.

9) P.M.Williams, "Bayesian regularization and pruning using a Laplace prior ," *Neural Computation*, VOL.7,pp117-143 1995.

10) D.J.C.MacKay, "a practical Bayesian framework for backpropagation," *Neural Computation*, VOL.4,No.3 pp448-478 1992.

11) Yan-Qing Zhang and Abraham Kandel, "Compensatory Neurofuzzy Systems with Fast Learning Algorithms ," *IEEE TRANSACTIONS ON NEURAL NETWORKS*, VOL.9,NO.1,pp83-105 1998.

12) G.A.Carpenter and S.Grossberg,rt al.,"A neural network architecture for incremental supervised learning of analog multidimensional maps ," *IEEE TRANSACTION ON NEURAL NETWORKS*, VOL.3,pp698-713,Sept. 1992.

13) P.J.Angeline,G.M.Sauders and J.B.Pollack, " An evolutionary algorithm that constructs recurrent neural networks ," *IEEE TRANSACTION ON NEURAL NETWORKS*, VOL.5,pp54-65, 1994.

14) Chao-Chee Ku,and Kwang Y.Lee, " Diagonal Recurrent Neural networks for Dynamic Systems Control," *IEEE TRANSACTION ON NEURAL NETWORKS*, VOL.6,NO.1,JANUARY 1995.

15) Tainping Chen and Hong Chen, " Approximation Capability to Functions of Several Variables, Nonlinear Functional and Operators by Radial Basis Function Neural Networks," *IEEE TRANSACTION ON NEURAL NETWORKS*, VOL.6,NO.4,July 1995

16) Chia-yiu Maaand Michael A.Shanblatt,"A Tow-Phase Optimization neural network" *IEEE TRANSACTIONS ON NEURAL NETWORKS*, VOL.3,NO.6,pp1003-1009, November,1992.