A Hidden Node Value Regulation Algorithm for Neural Network Training

LU Jin

(Department of Automation, Tsinghua University · Beijing, 100084, PRC)

Abstract: Error back propagation algorithm is widely used for the training of multi layer feed forward neural networks. But the convergency of this method is actually an open problem, which often leads to its poor generalization capacity. In this paper, we probe the situation and propose a Hidden Node Value Regulation (HNR) algorithm for approximating single output functions based on the mapping neural network existence theorem. This is useful in neural identification, for many industrial plants are multi-input-single-output. Theoretical problems of the convergence and generalization capacity of the HNR algorithm are also studied. Simulation results as well as an application case of modeling a catalytic reform process show some interesting characteristics of this algorithm, including its relatively high speed of sample learning and strong capacity of generalization under certain circumstance

Key works: neural network; learning algorithm; generalization capacity

1 Introduction

It is well known that a properly constructed multi layer feed forward neural network can implement any continuous function defined over a compact subset of Euclidean space. However, in many application cases, we find that the way by which we select samples and the number of hidden nodes greatly influence the property of the learned network. It is difficult to know how to decide the network structure and sample set in order to learn a given function. The only thing we can do is trial-and-error. In this paper, we study the mapping neural network existence theorem[1,2], and propose an algorithm that can make a neural network learn more efficiently.

This paper is arranged as following: In part 2, we survey the general multi layer neural networks and mapping neural network existence theorem. In part 3, we show the HNR structure and corresponding algorithm. In part 4, we offer some theorems on the convergency of this method and discuss its generalization capacity. And then in part 5, simulation results are given, together with some interesting features of the proposed HNR algorithm. Part 6 offers an application case. Finally, in part 7, we draw some conclusions and discuss problems for further research.

2 Multi Layer Neural Networks (MLN) and Mapping Neural Network Existence Theorem

An ordinary multi layer neural network is composed of many sigmoidal nodes (neurons)

Manuscript received Nov. 7, 1994, revised Jun. 11, 1997.

connecting among different layers. Each sigmoidal node has an identical structure, which includes: synapse function $f(x)$, activation function $g(x)$, and output function $o(x)$. With the neuron inputs $I_1, I_2, \cdots, I_n$, connection weights W_i, and threshold θ, we often set $f(x) = \sum_{i=1}^{n} I_i W_i + \theta$, and $o(x) = 1$. Thus, the neuron output $y = g(\sum_{i=1}^{n} I_i W_i + \theta)$. A general MLN consists of one or more hidden layers. The famous BP algorithm[2] solved the problem of hidden node training, and became widely used.

The original purpose of using neural network is to learn sampled input-output pairs in order to memorize them, so that when given an input, the learned network can recall the correct output. In 1957, Andrei Kolmogorov published an astounding theorem concerning the representation of arbitrary continuous functions[2]. With this theorem and another theorem published by D. A. Sprecher, the mapping neural network existence theorem was proposed[4].

Mapping Neural Network Existence Theorem: For any continuous function $f(x_1, x_2 \cdots, x_n)$ defined over a bounded subset of a n-dimensional Euclidean space and an arbitrarily given constant $\varepsilon > 0$, there exist a fixed continuous increasing function $\phi(x)$, constants $N, c_i, \theta_i (i = 1, \cdots, N)$, and $w_{ij} (i = 1, \cdots, N, j = 1, \cdots, n)$ such that with the function $\tilde{f}(x_1, x_2, \cdots, x_n)$ written in the form

$$\tilde{f}(x_1, x_2, \cdots, x_n) = \sum_{i=1}^{N} c_i \phi(\sum_{j=1}^{n} w_{ij} x_j + \theta_i),$$

we can expect $\max |f(x_1, x_2, \cdots, x_n) - \tilde{f}(x_1, x_2, \cdots, x_n)| < \varepsilon.$

This means a properly constructed and well trained 3-layer neural network can implement any continuous function defined over a bounded space. Hornik, Stinchcombe, and White[5] did a further study and proved that $\phi(x)$ can be even limited in the range of sigmoidal functions. From then on, we started to expect to learn not only the samples, but the function itself that generates these samples. We expect to get correct output of an untrained input, which means strong generalization capcity of a neural network.

3 HNR Structure and HNR Algorithm

As mentioned in part 2, the mapping neural network existence theorem tells us that a p-input-one-output function $f(x_1, x_2, \cdots, x_p)$ can be expressed by

$$f(x_1, x_2, \cdots, x_p) = \lim_{n \to \infty} \sum_{i=1}^{n} b_i \phi(\sum_{j=1}^{p} a_{ij} x_j + \theta_i).$$

Or in another form, if we define

$$\tilde{f}(x_1, x_2, \cdots, x_p) = \sum_{i=1}^{n} b_i z_i, \tag{1}$$

$$z_i = \phi(\sum_{j=1}^{p} a_{ij} x_j + \theta_i), \quad i = 1, \cdots, n, \tag{2}$$

then, we may find a constant N and corresponding coefficients a_{ij}, b_i, θ_i so that while $n \geqslant N$, $|f(x_1, x_2, \cdots, x_p) - \tilde{f}(x_1, x_2, \cdots, x_p)| < \varepsilon$. The problem is how to construct and learn the function $\tilde{f}(x_1, x_2, \cdots, x_p)$. We may notice from (1) and (2) that there are two stages to build

function $\tilde{f}(x_1, x_2, \cdots, x_p)$ under a given number n :

1) find a_{ij}, θ_i, so that z_i calculated by (2) are highly linear dependent on $f(x_1, x_2, \cdots, x_p)$.

2) find b_i, so that $f(x_1, x_2, \cdots, x_p) = \sum_{i=1}^{n} b_i z_i$ under some criterion.

This is the fundamental idea of HNR algorithm, which will be illustrated in detail hereafter.

3.1 HNR Network Structure

A HNR network has three layers, which are input layer, hidden layer and output layer. The hidden layer is composed of sigmoidal nodes, with activation function $g(x) = \frac{1}{(1 + e^{-x})}$. Other nodes are input-output equal, with $g(x) = 1$. A HNR network structure is shown as Fig. 1.

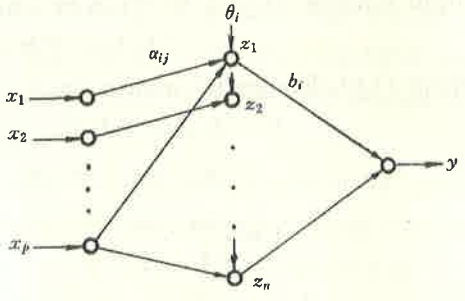


Fig. 1 HNR network structure

3.2 HNR Algorithm

For easy understanding, we first review the definition of linear dependence. With an equation

$$y = \varphi + \sum_{i=1}^{n} b_i z_i$$

and m sample pairs $(y_1, z_{i1}), (y_2, z_{i2}), \cdots, (y_m, z_{im}), i = 1, \cdots, n$, we can define a linear relationship coefficient as

$$\rho(y, z_i) = \frac{1}{L_{00}} \sum_{i=1}^{n} L_{i0} b_i,$$

where

$$L_{00} = \sum_{l=1}^{m} (y_l - \bar{y})^2, \quad \bar{y} = (\sum_{l=1}^{m} y_l)/m,$$

$$L_{i0} = \sum_{l=1}^{m} (z_{il} - \bar{z}_i)(y_l - \bar{y}), \quad \bar{z}_i = (\sum_{l=1}^{m} z_{il})/m, i = 1, \cdots, n.$$

Under this definition, $\rho = 1$ is the necessary and sufficient condition of $P(y = \varphi + \sum_{i=1}^{n} b_i z_i) = 1$. With this theorem, we may propose HNR algorithm as follows:

1) Give b_{i0}, a_{ij0} as initial values of b_i and a_{ij}.

2) Adjust a_{ij}, so that $\rho(y, z_i) \to 1$.

Define

$$E = \frac{1}{2}(\rho - 1)^2.$$

then

$$\frac{\partial E}{\partial a_{ij}} = (\rho - 1) \frac{1}{L_{00}} \sum_{i=1}^{n} \frac{\partial L_{i0}}{\partial a_{ij}} b_i, \quad \frac{\partial L_{i0}}{\partial a_{ij}} = \sum_{l=1}^{m} (y_l - \bar{y})(\frac{\partial z_{il}}{\partial a_{ij}} - \frac{\partial \bar{z}_i}{\partial a_{ij}}),$$

$$\frac{\partial \bar{z}_i}{\partial a_{ij}} = \frac{1}{m} \sum_{l=1}^{m} \frac{\partial z_{il}}{\partial a_{ij}}, \quad \frac{\partial z_{il}}{\partial a_{ij}} = g'(\sum_{j=1}^{p} a_{ij} x_{jl} + \theta_i) x_{jl}, \quad l = 1, \cdots, m.$$

Set

$$\Delta a_{ij} = -\lambda \frac{\partial E}{\partial a_{ij}}.$$

After convergence, start step 3.

3) With y_l and z_{il}, use LS algorithm to get b_i.

4) Repeat step 2, till convergence.

We may know from following parts, that a suitable choice of b_{i0} and samples will assure the convergence of HNR algorithm and give HNR network strong generalization capacity.

4 Results on Convergency of HNR Algorithm

4.1 Problem Formulation

We know from the mapping neural network existence theorem that any p-input-one-output function $f(\underline{x}), \underline{x} \in \mathbb{R}^p$ can be expressed by

$$f(\underline{x})_\varepsilon = \underline{b}^{\mathrm{T}} \underline{\phi}(A\underline{x}), \quad \underline{x} \in \underline{X}_d, \underline{X}_d \subseteq \mathbb{R}^{p+1}. \tag{3}$$

Where $f(\underline{x})_\varepsilon$ denotes the approximation value of $f(\underline{x})$ within precision ε, and

$$\underline{b} = [b_1, b_2, \cdots, b_n]^{\mathrm{T}}, \quad \underline{x} = [x_1, x_2, \cdots, x_p, 1]^{\mathrm{T}},$$

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1p} & \theta_1 \\ a_{21} & a_{22} & \cdots & a_{2p} & \theta_2 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{np} & \theta_n \end{bmatrix}, \quad \underline{X}_d \text{ is the definition domain.}$$

Therefore, we may construct a HNR network for learning function (3) as

$$\tilde{f}(\underline{x}) = \tilde{\underline{b}}^{\mathrm{T}} \tilde{\underline{z}}(\underline{x}), \quad \tilde{\underline{z}}(\underline{x}) = \underline{\phi}(\tilde{A}\underline{x}). \tag{4}$$

Our objective is to find optimal parameters $\tilde{A}$ and $\tilde{\underline{b}}$ by HNR algorithm based on a certain set of samples $(\underline{x}_l, y_l), l = 1, 2, \cdots, m$, and get a solution W_0^* of the learning problem with (4). Now, we regard $y_l = f(\underline{x})_l$ as the observation value of $f(\underline{x})_{\varepsilon_l}$ with precision ε_l. A solution of problem (4) is defined as

$$W_0^* \in W^*, \quad W(\tilde{\underline{b}}, \tilde{A}) = \{\tilde{\underline{b}} \in \mathbb{R}^n, \tilde{A} \in \mathbb{R}^{n(p+1)} : \| f(\underline{x})_\varepsilon - \tilde{\underline{b}}^{\mathrm{T}} \underline{\phi}(\tilde{A}\underline{x}) \| < \varepsilon, \underline{x} \in \underline{X}_s\},$$

where $\underline{X}_s$ is the sample set.

4.2 Theoretical Results

While studying a new algorithm, we mostly concern about its convergency. As to neural networks, we still concern about its generalization capacity after training. Because these two problems are very difficult to study, hereby, we just try to find some sufficient conditions on the convergency of HNR algorithm and discuss its generalization capacity.

We will hold the following three assumptions throughout this paper.

C0: Sample output $y_l, l = 1, 2, \cdots, m$ is not chosen as a constant.

C1: Sample input $\underline{x}_l, l = 1, 2, \cdots, m$ can be freely selected in the definition domain $\underline{X}_d$.

C2: Residual of equation (4) ε_l under randomly selected sample $\underline{x} \backslash -l, l = 1, 2, \cdots, m$ can be treated as white noise.

Theorem 1 In HNR algorithm, under assumption C0, if $b_i, i = 1, 2, \cdots, n$ is not a solution of equation

$$\begin{bmatrix} \dfrac{\partial L_{10}}{\partial a_{11}} & \dfrac{\partial L_{20}}{\partial a_{21}} & \cdots & \dfrac{\partial L_{n0}}{\partial a_{n1}} \\ \dfrac{\partial L_{10}}{\partial a_{12}} & \dfrac{\partial L_{20}}{\partial a_{22}} & \cdots & \dfrac{\partial L_{n0}}{\partial a_{n2}} \\ \vdots & \vdots & \ddots & \vdots \\ \dfrac{\partial L_{10}}{\partial a_{1(p+1)}} & \dfrac{\partial L_{20}}{\partial a_{2(p+1)}} & \cdots & \dfrac{\partial L_{n0}}{\partial a_{n(p+1)}} \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix} = 0, \tag{5}$$

then the necessary and sufficient condition of step 2 of HNR algorithm converging to $\rho = 1$ is $\Delta a_{ij} = 0, i = 1,2,\cdots,n, j = 1,2\cdots,p+1$.

Remark Obviously, we may complete the whole algorithm by assuring the convergence of step 2. If we may obtain $b_i, i = 1,2,\cdots,n$ satisfying Theorem 1, the HNR algorithm may converge to $\rho = 1$ while $\Delta a_{ij} = 0$. We may see from (5) that this is related to the selection of sample set.

Theorem 2 In HNR algorithm, under assumption C0 and C1, if there exist sample inputs $\underline{x}_l, l = 1,2,\cdots,m$, that make each element of the left side of (5)>0, then the HNR algorithm 1) converges. 2) $\rho(k)$ converges to 1 at an exponent rate, i.e. $\exists\ r \in (0,1]$, such that

$$\| \rho(k) - 1 \| \leqslant r^k \| \rho(0) - 1 \|, \quad k = 0,1,\cdots.$$

Remark Theorem 2 together with theorem 1 offers a sufficient condition for convergence of HNR algorithm. But their conditions seem to be too strict. However, from our simulations in part 5 and the application case in part 6, we find these conditions can be fulfilled in most practical cases.

Theorem 3 In HNR algorithm, under assumption C2, if step 2 converges to $\rho = 1$, then the solution of step 2 $W_0^* \in W^*$, while LMS error of HNR solution (after step 3) $\varepsilon' \leqslant \varepsilon$.

Remark From this theorem we know the convergence of HNR algorithm assure the convergence of HNR network on sample set. This is critical, because to memorize given samples is fundamental. We may notice that $\varepsilon \to 0$ leads to $\varepsilon' \to 0$. And we may see from latter discussion that step 3 is necessary for assuring the generalization capacity.

Theorem 4 In HNR algorithm, under assumption C1 and C2, if 1) step 2 converges to $\rho = 1$. 2) the minimum in step 2 is unique, then $\tilde{A} = A$, and $E(\tilde{b}) = b$.

Remark Theorem 4 tells us that under some circumstance, the HNR network will really converge to the function it learns. Of course, these conditions are related to the sample set fed to HNR network and can be treated as guidances of sample selection.

Using above theorems, we may explain some widely used experience for network structure decision. 1) With a given set of samples, the more hidden nodes, the weaker the generalization capacity. We know with a given set of samples, more hidden nodes means more minima in HNR step 2. So the second premise of theorem 4 becomes difficult to meet, which leads to weaker generalization capacity. We also know, this rule is not always correct. For example, if the sample number is too small, no matter how we adjust the network structure, we cannot get high performance generalization. Theorem 4 can also explain this, for assumption C0 and its first premise are related to the sample set. Only when the network together with its samples fulfills all the assumptions of theorem 4 can it be generalized. 2) With a given function, the more the hidden nodes (with adequate samples), the stronger the generalization capacity. This can be obviously deducted from Theroem 3. More hidden nodes and samples means smaller ε, which leads to smaller ε'.

5 Simulation Results

In order to test the performance of HNR algorithm, we did various simulation experiments. For easy demonstration with curves, all test examples hereby use one-input-one-out-

put functions. With these simulations, we show some interesting features of HNR algrithm.

5.1 High Learning Speed

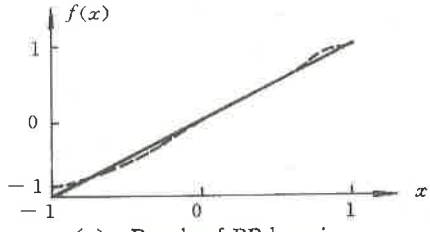
The error back propagation algorithm consumes a great deal of computing time[6]. In HNR algorithm, we use direct, not recursive, LS method to get parameter b_i at step 3. Therefore, we only search optimal a_{ij} in the first layer, not through all the layers like BP algorithm. Thus, the learning speed is greatly improved. Table 1 shows the convergence iterations of BP and HNR algorithm in learning $f(x) = x, x \in [0,1], f(x) = x^2, x \in [0,1], f(x) = \sin(2\pi x), x \in [0,1]$ and $f(x) = 1/2.5x + \ln(2.5x), x \in [0.2,1]$. It must be pointed out that in learning $f(x) = \sin(2\pi x)$, we have to set hidden node number to 10, instead of 9 with HNR, for the convergence of BP algorithm. We may notice from Table 1 that the convergence iterations are enormously reduced by HNR algorithm.

Table 1 Convergence speed comparison of BP and HNR

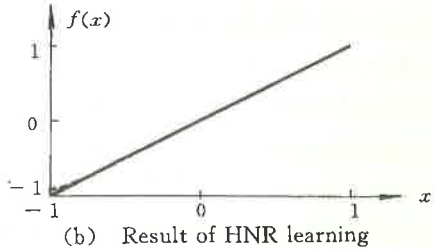
$f(x)$	x	x^2	$\sin(2\pi x)$	$1/2.5x + \ln(2.5x)$
BP	30960(1,4,1)	709(1,7,1)	5693(1,10,1)	9114(1,9,1)
HNR	14(1,4,1)	13(1,7,1)	16(1,9,1)	18(1,9,1)
Error	0.001	0.01	0.01	0.03

5.2 Strong Generalization Capacity

We know the generalization capacity of neural network trained by BP algorithm is often poor. For example, if we want to train a BP network to implement $f(x) = x, x \in [-1,1]$, and just offer samples on the interval $[0,1]$, the trained network can never have good performance on the interval $[-1,0]$. This is shown in Fig. 2(a). (solid line for $f(x)$ and dashed line for $\hat{f}(x)$). From Fig. 2(b), we may observe HNR network's strong generalization capacity with the identical sample set. In learning function $f(x) = x^2$, and many other functions, we get simliar results. And we also find, with hidden node number increasing, the generalization capacity of the trained HNR network becomes stronger. The results of learning $f(x) = x, x \in [0,0.9]$ with different hidden node number and generating it to the interval $[-1,1]$ are shown in Fig. 3 and Fig. 4.



(a) Result of BP learning



(b) Result of HNR learning

Fig. 2 Generalization ability of BP and HNR network

6 Application Case

After composing the HNR algorithm, we use it in the prediction of aromatic hydrocarbon content in a catalytic reform process. The input variables here are temperature, pressure, velocity, ratio of H_2 over hydrocarbon, density of oil, and quality of oil. In an oil refining plant, operators should constantly adjust the operating parameters for resisting the fluctuation of input oil quality to assure stable aromatic hydrocarbon content of output oil. Therefore, it is

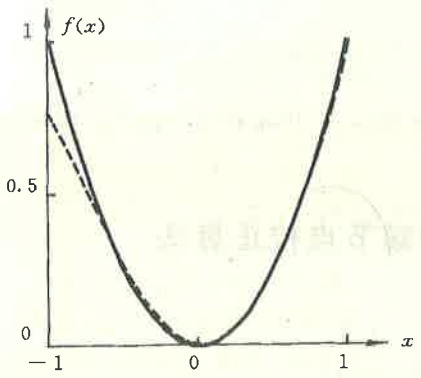


Fig. 3 HNR generalization: 8 hidden nodes with 9 samples

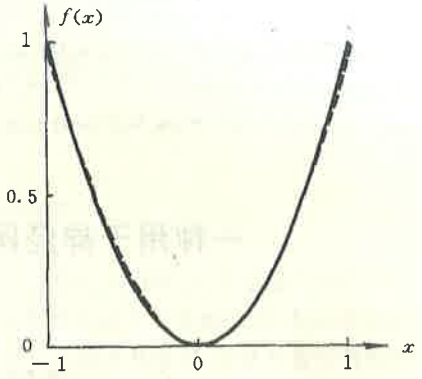


Fig. 4 HNR generalization: 14 hidden nodes with 17 samples

very important to predict aromatic hydrocarbon content under a certain condition. One of our task is to use a HNR network to accomplish this task. We use 20 samples selecting from the operation data of September and October 1993 to train HNR network, it converges after 384 iterations to the precision of 0.01. While using BP algorithm to train with the same sample set, we get convergence after 76995 iterations to the same precision. The predicting and testing values of aromatic hydrocarbon content of 18 practical operation days in December 1993 and January 1994 are listed in Table 2. We may notice they are very similar to one another.

Table 2 Prediction of content of aromatic hydrocarbon

Testing	52.88	51.49	55.48	53.78	55.64	52.00	53.44	53.76	55.40
Predicting	52.97	49.07	52.23	49.77	51.73	52.41	52.97	53.27	54.60
Testing	52.00	51.93	52.21	52.38	46.91	46.71	52.44	49.46	53.31
Predicting	51.85	51.74	52.04	52.61	48.67	46.61	51.56	49.48	52.92

7 Conclusion

We have discussed the structure, algorithm and convergence property of HNR algorithm, and also observed many advantages of this algorithm over old ones, especially over the widely used BP algorithm. However, there are still many problems left to be probed further. Although we have observed some interesting features about the generalization capacity of the algorithm, the discussion of this generalization capacity is not enough at present time. Therefore, the properties of this algorithm should be given more study and be tested by more practical applications.

References

1 Hecht-Nielsen, R.. Theory of the backpropagation neural network. Neural Networks for Perception. Vol. 2, Ed. Harry Wechsler, San Diego, CA: Academic Press, 1991

2 Hecht-Nielsen, R.. Mapping networks: multi-layer data transformation structures. Neurocomputing. San Diego, CA: Addison-Wesley Publishing Company, 1990

3 Rumelhart, D. E.. Learning internal representation by error propagation. Parallel Distributed Processing. Combridge, MA: MIT Press, 1986

4 Yan, Ping-fan. Mapping capacity of feedforward neural network. Artificial Neural Network. Hefei: Anhui Education

Press,1991

5 Hornik,K. ,Stinchcombe,M. and White,H. . Multi layer feedforward networks are universal approximators. Neural Networks,1989,2(3):359—366

6 Davalo,Eric and Patrick Naim. Multilayer neural network. Neural Networks,Hong Kong:Macmillan Education LTD,1991

一种用于神经网络训练的隐节点校正算法

卢 进

(清华大学自动化系·北京,100084)

摘要:误差反传算法被广泛用于多层前馈神经网络的训练.但该算法的收敛性问题并没有解决,这导致训练后的网络泛化能力一般很差.本文研究了这一问题,并基于神经网络映射定理提出一种用于训练网络逼近单输出函数的隐节点校正(HNR)算法.这在神经辨识领域是有用的,因为大多数工业对象都是多输入单输出的.我们对HNR算法的收敛性和泛化能力作了理论上的研究.仿真实验和在催化重整过程建模中的应用实例表明该算法在一定条件下具有很高的学习速度和较强的泛化能力.

关键词:神经网络;学习算法;泛化能力

本文作者简介

卢 进 1967年生.现为清华大学自动化系研究生.研究兴趣包括:非线性系统辨识与控制,自适应控制,神络网络理论,智能控制等.