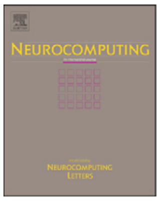
Neurocomputing 71 (2008) 3640–3643

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom



Letters

Solving local minima problem with large number of hidden nodes on two-layered feed-forward artificial neural networks

Bumghi Choi [a,*], Ju-Hong Lee [a], Deok-Hwan Kim [b]

[a] Department of Computer Science and Information Engineering, Inha University, Incheon, Republic of Korea
[b] Department of Electronic Engineering, Inha University, Incheon, Republic of Korea

ARTICLE INFO

Article history:
Received 13 March 2007
Received in revised form
17 January 2008
Accepted 4 April 2008
Communicated by T. Heskes
Available online 3 May 2008

Keywords:
Backpropagation
Local minima
Hidden nodes
Target values
Separate learning

ABSTRACT

The gradient descent algorithms like backpropagation (BP) or its variations on multi-layered feed-forward networks are widely used in many applications. However, the most serious problem associated with the BP is local minima problem. Especially, an exceeding number of hidden nodes make the corresponding network deepen the local minima problem. We propose an algorithm which shows stable performance on training despite of the large number of hidden nodes. This algorithm is called separate learning algorithm in which hidden-to-output and input-to-hidden separately trained. Simulations on some benchmark problems have been performed to demonstrate the validity of the proposed method.
© 2008 Elsevier B.V. All rights reserved.

1. Introduction

Backpropagation (BP) is a well-known representative of all gradient descent algorithms, which is widely used technique for supervised neural network learning in many areas of application. BP has a great virtue of simplicity on implementation and calculation. Despite that there are many problems on BP. The most serious problem of BP is that the learning process can not guarantee to a global minimum, trapping into local minima or saturation points. Techniques from global optimization (GO) have been proposed to solve the local minima problem of BP. The GO can be classified broadly into two major groups, deterministic and stochastic. Deterministic methods [2,3,5,8,11] guarantee convergence to a global optimum only for certain classes of functions [6], and require more auxiliary calculations. Stochastic algorithms [1,4,9], guarantee convergence to global optimum only when the number of searches can be allowed infinitely. Among them, simulated annealing and genetic algorithms are well known.

A local minimum is a suboptimal equilibrium point at which system error is non-zero and the hidden output matrix is singular [12]. The complex problem which has a large number of patterns needs as many hidden nodes as patterns in order not to cause a singular hidden output matrix. The exceeding number of hidden nodes can possibly make al lot of local minima.

Our goal is to invent a novel idea to weaken the local minima problem in the networks with the exceeding number of hidden nodes. The new algorithm will be called separate learning algorithm, in which the hidden-to-output connections and the input-to-hidden connections separately trained as Fig. 1 with a modified gradient descent method.

2. Separate learning algorithm

We concentrate our attention just on a two-layered network. We expect to easily extend to multi-layer environment. As described in Fig. 1, the separate learning algorithm is to separate a whole network into two networks, hidden-to-output network, input-to-hidden network. First, the hidden-to-output connection is trained as usual BP with the following cost function:

$$E_{\text{hidden-to-output}}[w] = \frac{1}{2}\sum_{\mu j}\left[\zeta_j^\mu - g\left(\sum_i w_{ij}^2 y_i\right)\right]^2 \tag{1}$$

where g is a sigmoid or tangent hyperbolic function, $\zeta_j{}^\mu$ is the ideal or target value of output unit j for input pattern μ where y_i is the output value of hidden layer unit i, $w_{ij}{}^2$ is the weight from hidden unit i to output unit j.

* Corresponding author.
E-mail addresses: bumghichoi@yahoo.co.kr (B. Choi), juhong@inha.ac.kr (J.-H. Lee), deokhwan@inha.ac.kr (D.-H. Kim).

0925-2312/$ - see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.neucom.2008.04.004

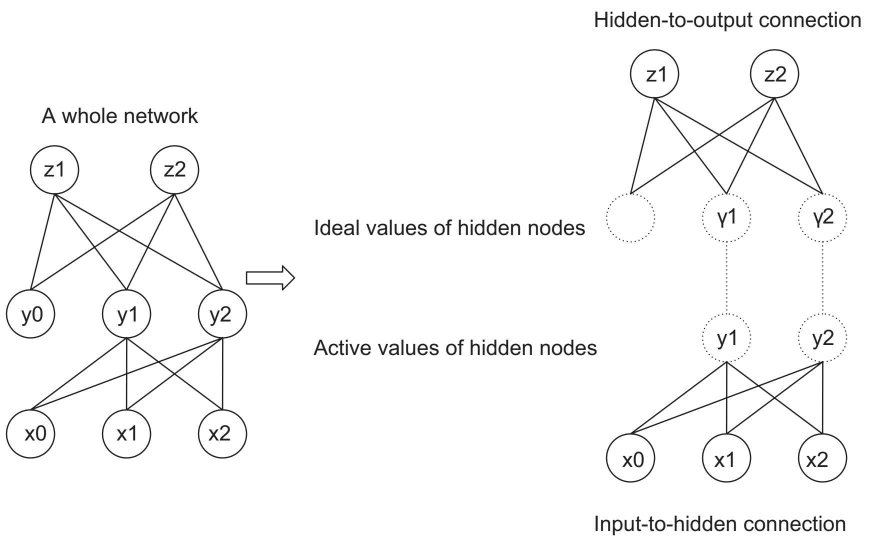

A whole network
z1
z2
y0
y1
y2
x0
x1
x2
Hidden-to-output connection
Ideal values of hidden nodes
Active values of hidden nodes
Input-to-hidden connection

Fig. 1. The separation of a whole network.

For the hidden-to-output connections, the weight updating rule gives

$$\Delta w_{ij}^2 = -\eta \frac{\partial E}{\partial w_{ij}^2} = \eta \sum_{\mu} [\zeta_j^\mu - g(h(2)_j^\mu)] g'(h(2)_j^\mu) y_i^\mu \tag{2}$$

where η is a learning rate ranged from 0 to 1, and $h(2)_j{}^\mu$ is the input value to output unit j for input pattern μ. As described in Fig. 1, for training the hidden layer, ideal values of hidden units are defined as the correct value to eliminate the error of output layer, and derived by Newton's method approximately.

The error function is approximated by Newton's method using first derivatives only

$$E(w) \approx E(Y) + (\Gamma - Y)\nabla E(Y) \tag{3}$$

where Y is a current vector in hidden layer for a pattern μ, and Γ is the ideal vector of Y such that $E(w) = 0$. Γ is the most desirable when $E(w) = 0$. Thus,

$$\Gamma - Y \approx \frac{-E(Y)\nabla E(Y)}{|\nabla E(Y)|^2} \tag{4}$$

The ideal value of hidden unit i, γ_i is the ith component of the vector Γ and y_i is the ith component of Y

$$\gamma_i \approx \frac{E(Y)\sum_j (\varsigma_j - z_j) g'(h(2)_j) w_{ij}}{\sum_i (\sum_j (\varsigma_j - z_j) g'(h(2)_j) w_{ij})^2} + y_i \tag{5}$$

Now the evaluation function for the hidden-to-connections is derived from the ideal value in (5)

$$E_{\text{input-to-hidden}}[W] = \frac{1}{2} \sum_{\mu i} \left[\gamma_i^\mu - g\left(\sum_k w_{ki}^1 \xi_k^\mu \right) \right]^2 \tag{6}$$

For the input-to-hidden connections, the error function is partially differentiated with respect to the w_{ki}

$$\Delta w_{ki}^1 = -\eta \frac{\partial E_{\text{input-to-hidden}}}{\partial w_{ki}^1} = \eta \sum_{\mu} (\gamma_i^\mu - y_i^\mu) g'(h(1)_i^\mu) \xi_k^\mu \tag{7}$$

where $h(1)_i{}^\mu$ is the input value to hidden unit i for an input pattern μ.

The proposed algorithm can be differentiated from other approaches as mentioned in that it does not use a random perturbation of parameters or the hidden output values. It just adopts a divide-and-conquer strategy in which a whole network is divided into two networks, hidden-to-output and input-to-hidden. Each part is trained differently. Training the upper connections is same as BP. But to evaluate in the input-to-hidden connections, we need the ideal values of hidden units as Eq. (5). The algorithm is summarized as follows:

1. Compute the output values of hidden and output layers using given parameters and input values for each pattern at the first epoch.
2. Train the hidden-to-output connections using BP algorithm with the input-to-hidden connections fixed. The input values are to be the output values of hidden layer. The ideal value of each output unit is same as the original network. The error is evaluated by Eq. (1) and the weights of the hidden-to-output connections are updated by Eq. (2).
3. If the training process of 2 converges within the given time limit, the whole training will stop as a successful one. If the training is over the time limit or go slow (it is when the training error cannot be reduced more than 1% of the previous error), go to 4.
4. Select one hidden unit in order, and compute the ideal value approximately using (5).
5. Train the input-to-hidden connection for the selected hidden unit using Eq. (6) while other connections are fixed. If it converges or the training speed goes slow as described in 3, go to 2.

In the three-layered networks, the process 5 is subdivided in the same way as 2, 3, and 4. In this manner, the proposed algorithm can be easily extended to the general case in which multi-layered networks are provided.

3. Simulations and evaluation

3.1. Environment of experiments

In order to verify the performance of the separate learning algorithm proposed in this paper, experiments have been performed using a computer with an AMD XP 2600+2.0 GB CPU and 512 MB random access memory (RAM). Each experiment was practiced pnr BP, SARPRO [10], and proposed algorithm.

From the results, we compared and evaluated the convergence rates (success rate, SE), average learning time (time, in millisecond) in case of success. A mean square error limit of 0.01 and a training time limit of 30,000 ms are set to all experiments.