A Simultaneous Learning Method for Both Activation Functions and Connection Weights of Multilayer Neural Networks

Kenji Nakayama Moritomo Ohsugi
Dept. of Electrical and Computer Eng., Kanazawa Univ., Japan
nakayama@t.kanazawa-u.ac.jp

Abstract This paper proposes a simultaneous learning algorithm for both activation functions and connection weights. The activation function is composed of several basic functions, such as sigmoidal function, Gaussian function and so on. In order to avoid local minima, the activation functions are controlled and randomly disturbed every some epochs. The activation functions are automatically optimized for each application. Probability and speed of learning are higher than the conventionals.

Keywords — Activation function, Multilayer neural networks, Learning algorithm, Pattern classification

1. Introduction

In designing neural networks, fast learning, high possibility of convergence and small network size are very important. They are highly dependent on network models, learning algorithms and problems to be solved. They are also highly related to activation functions.

Many pruning methods for hidden units have been proposed [1],[2]. A method, which combine processes of selecting activation functions and of pruning hidden units, was proposed for multilayer neural networks trained by back-propagation algorithm [3]. Effective activation functions can be selected, with which the number of the hidden units can be minimized. This method is, however, one of the pruning methods, thus, a relatively large number of the hidden units should be prepared. Several kinds of activation functions have been used, and their comparison has been discussed [4],[5]. Furthermore, some learning methods for activation functions have been proposed [6]-[8]. However, types of the functions and efficiency are rather limited.

In this paper, we propose a simultaneous learning method, which can optimize both activation functions and connection weights through the gradient descent algorithm. Furthermore, in order to avoid local minima, a control method for the activation functions with random disturbance is also proposed. This is applied to the activation functions every some epochs. The parity check problem with 8 bits and pattern classification problems are used in computer simulation to verify efficiency of the proposed method.

2. Network Model

2.1 Activation functions

We employ a composite form activation functions, which combine several basic functions such as sigmoidal function, Gaussian function, sinusoidal function and so on. Among them, we employ the sigmoidal function in this paper. Of course our method is not restricted to this function.

The reason why we select the sigmoidal function is as follows: The Gaussian and sinusoidal functions can be composed of several sigmoidal functions. However, . the reverse approximation is rather difficult. This means it requires a large number of the Gaussian and sinusoidal functions.

The proposed activation function is expressed in Eq. (1), which is a sum of several sigmoidal functions.

$$u=\sum_{i=1}^{I} w_i x_i \tag{1a}$$

$$y=f(u)=\sum_{l=1}^{L} \{\frac{a_l}{1+exp(-b_l u - c_l)} + d_l\} \tag{1b}$$

u is the input potential, $f(\cdot)$ is the activation function and y is the output of the nonlinear unit. This activation function includes four parameters a_l, b_r, c_l and d_l in each basic function and L basic functions. So, 4L free parameters are used in one activation function. They will be optimized through the learning process together with the connection weights. The gradient descent algorithm can be applied for this purpose.

2.2 Network structure

The network structures, to which the proposed activation function can be applied, are not restricted. It can be applied any network structures. In this paper, the multilayer neural networks are taken into account.

The following two layer neural network is considered.

Input patterns:

$$x=\{x_i \mid i=0 \sim I),\ x_0=1 \tag{2}$$

*Output **of** hidden layer:*

$$y=\{y_j \mid j=0 \sim J\},\quad y_j=f_j(u_j),\quad y_o=l \tag{3a}$$

$$u_j=w_j^T x \tag{3b}$$

*Output **of** output layer:*

$$z=\{z_k \mid k=1 \sim K\},\ z_k=f_k(v_k) \tag{4a}$$

$$v_k= w_k^T y \tag{4b}$$

Connection weights:

Input- to-Hidden $w_j=\{w_{ji} \mid i=0 \sim I\},\ j=0 \sim J$
Hidden-to-Output $w_k=\{w_{kj} \mid j=0 \sim J\}, k=1 \sim K$

3. Learning Algorithm

3.1 Learning of parameters

The proposed learning algorithm is based on the gradient descent algorithm, which minimize the mean squared error. Letting d_k be the target for the output z_k, the mean squared error is given by,

$$E=\frac{1}{K}\sum_{k=1}^{K}(d_k - z_k)^2 \tag{5}$$

Furthermore, let $p(n)$ be parameters on behalf on the activation functions and the connection weights, where n is the iteration number. $p(n)$ is updated as follows:

$$p(n+1)= p(n) - \eta\frac{\partial E}{\partial p} \tag{6}$$

η is a learning rate. Furthermore, the correction is denoted as follows:

$$p(n+1)= p(n) + \Delta p(n) \tag{7}$$

Due to the page limitation, the correction term $\Delta p(n)$ for activation functions and the connection weights in the proposed algorithm are only summarized in the following.

. *Activation functions* in *the output layer:*

$$\mathrm{A}\, a_{kl}(n) = \eta\,\delta_k\,\phi_{kl} + a\,\mathbf{A}\, a_{kl}(n-1) \tag{8}$$

$$\mathrm{A}\, b_{kl}(n) = \eta\, 6\,_k a_{kl} v_k\,\phi_{kl}(1-\phi_{kl})+\alpha\,\mathrm{A}\, b_{kl}(n-1) \tag{9}$$

$$\mathbf{A}\, c_{kl}(n) = \eta\, 6\,_k a_{kl}\,\phi_{kl}(1-\phi_{kl})+\alpha\,\mathrm{A}\, c_{kl}(n-1) \tag{10}$$

$$\mathrm{A}\, d_{kl}(n) = \eta\, 6\,_k + a\,\mathrm{A}\, d_{kl}(n-1) \tag{11}$$

$$\delta_k=d_k - z_k \tag{12}$$

$$\phi(x)=1/(1 + e^{-x}) \tag{13}$$

$$\phi_{kl}= \phi(b_{kl}(n)v_k + c_{kl}(n)) \tag{14}$$

a is a learning rate of the momentum term.

. *Connection weights from hidden layer to output layer:*

$$\mathrm{A}\, w_{kj}(n)=\eta\, 6\,_k y_j \sum_{l=1}^{L}\{a_{kl} b_{kl}\,\phi_{kl}(1-\phi_{kl})\} + a\,\mathrm{A}\, w_{kj}(n-1) \tag{15}$$

. *Activation functions in hidden layer:*

$$\mathrm{A}\, a_{jl}(n)=\eta\,\phi_{jl}\sum_{k=l}^{K}[\,6\,_k w_{kj}\sum_{l=1}^{L}\{a_{kl} b_{kl}\,\phi_{kl}(1-\phi_{kl})\}] + a\,\Delta a_{jl}(n-1) \tag{16}$$

$$\phi_{jl}= \phi(b_{jl} u_j+c_{jl}) \tag{17}$$

$$\mathrm{A}\, b_{jl}(n)=\eta\, a_{jl} u_j\,\phi_{jl}(1-\phi_{jl})\sum_{k=l}^{K}[\,\delta_k w_{kj} \cdot \sum_{l=1}^{L}\{a_{kl} b_{kl}\,\phi_{kl}(1-\phi_{kl})\}] + \alpha\,\mathrm{A}\, b_{jl}(n-1) \tag{18}$$

$$\mathrm{A}\, c_{jl}(n)=\eta\, a_{jl}\,\phi_{jl}(1-\phi_{jl})\sum_{k=l}^{K}[\,6\,_k w_{kj} \cdot \sum_{l=1}^{L}\{a_{kl} b_{kl}\,\phi_{kl}(1-\phi_{kl})\}] + \alpha\,\mathrm{A}\, c_{jl}(n-1) \tag{19}$$

$$\Delta d_{jl}(n)= \eta \sum_{k=1}^{K}[\,\delta_k w_{kj} \sum_{l=1}^{L}\{a_{kl} b_{kl}\,\phi_{kl}(1-\phi_{kl})\}]$$

$$+ \alpha \Delta d_{jl}(n-1) \tag{20}$$

· *Connection weights from input layer to hidden layer:*

$$\Delta w_{ji}(n) = \eta\, x_i \sum_{l=1}^{L} a_{jl} b_{jl} \phi_{jl}(1-\phi_{jl})$$
$$\cdot \sum_{k=1}^{K} \{ \delta_k w_{kj} \sum_{l=1}^{L} a_{kl} b_{kl} \phi_{kl}(1-\phi_{kl}) \}$$
$$+ \alpha \Delta w_{ji}(n-1) \tag{21}$$

α is a learning rate of the momentum term.

3.2 Control of activation functions

When the linear part of the sigmoidal functions locate out of the interest regions, derivative of the sigmoidal functions becomes very small, resulting in very slow convergence. Furthermore, if the linear part of several basic functions locate at the same place, they cannot be effectively used to approximate some functions to be realized.

In order to use all the basic functions effectively, we introduce the following control methods.

.Shifting Basic Functions

The sigmoidal functions are controlled so that their linear part locate in the interesting regions. This kind of method was proposed for accelerating the BP algorithm,

The input potential distribution is evaluated. Let u_{max} and u_{min} be the maximum and minimum of u in Eq. (1a) . If the center of the sigmoidal function, that is $t_l=-c_l/b_l$ locates the outside of $[u_{min}, u_{max}]$, then t_l is modified as,

$$t_l = \frac{u_{max}+u_{min}}{2} + \frac{u_{max}-u_{min}}{2} | \tag{22}$$

θ is a random number distributed within [-1, 1].

.Disturbing Basic Functions

In order to remove the redundancy, caused by the basic functions locate at the same place, the centers, even though they locate within $[u_{min}, u_{max}]$ are also controlled as follows: If the distance between the center of several basic functions is less than $(u_{max} - u_{min})/2P$, where P is the number of the data used in the learning,

$$\| t_l - t_k \| < (u_{max} - u_{min})/2P \tag{23}$$

then, they are controlled by

$$t_l(new) = t_l(old) + \frac{u_{max} - u_{min}}{2P} \theta \tag{24}$$

3.3 Equivalent neural networks

The proposed activation function with L basic functions can be decomposed into L hidden units with the basic function. However, the number of the connection weights is increased by L times as large as that of the proposed. Furthermore, possibility of convergence and convergence speed can be drastically improved by optimizing the activation function. Fixed activation functions mean that shape of the hyper planes composed of the connection weights to one neuron is fixed, only their location in the data space is optimized through a learning process. On the other hand, the proposed method can optimize both the shape of the hyper planes and their location.

If there are some limitations, under which only the sigmoidal function is available, for instance, then after the training, the proposed network can be decomposed into the model, which satisfies the above requirement. We want to claim once again the following. Even though the equivalent networks, having the basic function as an activation function, exist, learning performance of the proposed model is very high. This point can be confirmed through the simulation in Sec.5

4. Simulation and Discussions

4.1 Network structure

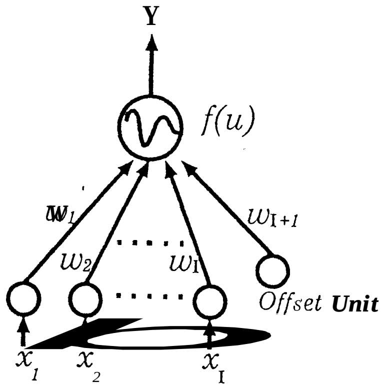

Fig.1 Network with proposed activation function.

In this paper, we employ the single output unit network without the hidden layer as shown in Fig.1. The activation function in the output. unit consists of L sigmoidal functions. Therefore, for comparison, the ordinary multilayer neural network（MLNN）, having a hidden layer. with L hidden units and one output unit as shown in Fig.2, is used.

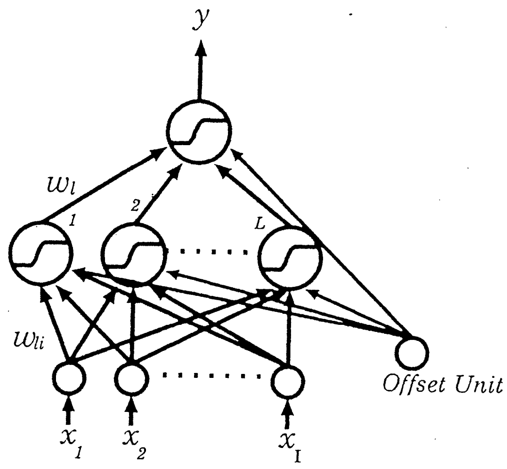

Fig.2 Multilayer neural network with sigmoidal hidden and output unit.

The proposed model has $I+1$ connection weights and one output unit with the composite function. The conventional model has $L(I+1)$ connection weights and $L+1$ units with the simple sigmoidal function.

4.2 Simulation results and discussions

.8bit Parity Problem:

This problem is rather difficult problem for multilayer neural networks. Because slightly different input data should be mapped onto entirely different outputs. The number of the basic functions in the proposed activation function L is set to be 40. The same number of hidden units and one output unit are used in the ordinary MLNN.

Ten trials are performed using different initial parameters. Figures 3 and 4 show the learning curves for the proposed and the conventional, respectively. In the former, 7 trials out of ten successfully converge. However, in the conventional, ten trials cannot converge or take a very long time. Thus, the proposed method holds a strong likelihood of fast convergence.

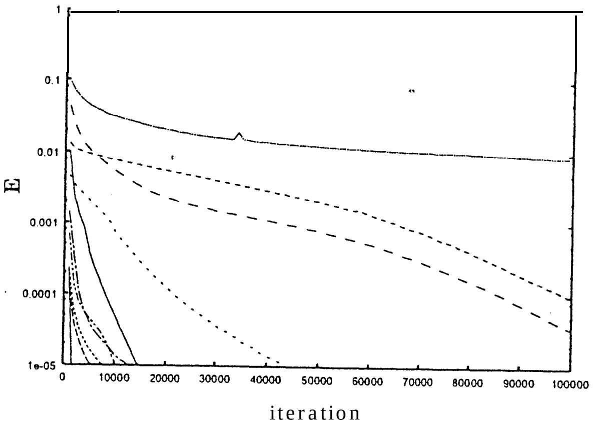

Fig.3 Learning curves of networks shown in Fig.1 with proposed activation function.

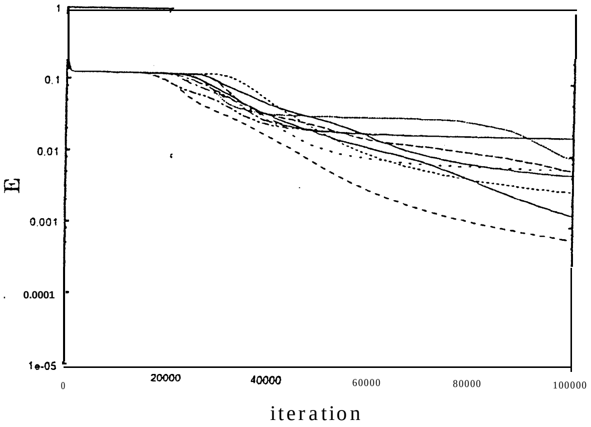

Fig.4 Learning curves of ordinary MLNN shown in Fig.2.

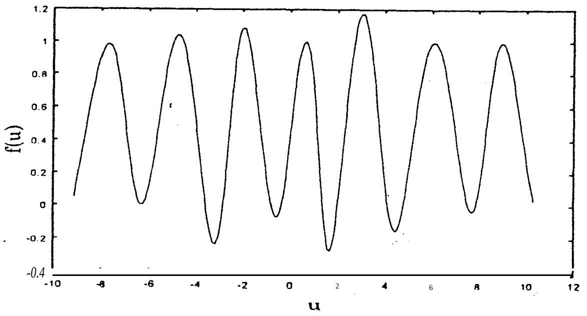

Fig.5 Activation function of output unit. Horizontal axis is input and vertical axis output.

Figure 5 shows the activation function. The horizontal axis indicates the input and the vertical the output. Since, n-bit parity mapping is a periodic function. So, one sinusoidal function, for instance, can realize this mapping [3].

Figure 5 shows the obtained activation function near the periodic function. From this result, we can confirm efficiency of the proposed method.

Of course, we can use some acceleration method for the ordinary MLNN, they are, however, applied to the proposed activation function in the same way. So, we compare the proposed learning method with the standard BP algorithm

.Pattern Classification in 2D Space

Figure 6 shows a two-class pattern classification problem. In the white slits, data are not distributed. In this case, two output units with the proposed activation function are used in the network shown in Fig.1. The targets for Class 1 and 2 are (1,0) and (0,1), respectively.

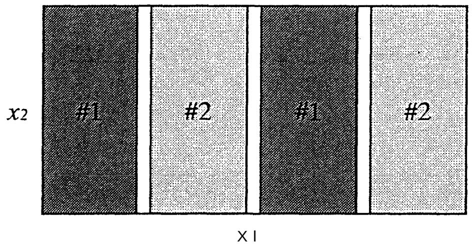

Fig.6 Two-class pattern classification.

Figure 7 shows the activation functions for Class 1 (dotted line) and Class 2 (solid line) . Here, x_2 is fixed. However, the activation functions have the same figure for any x_2 value. From this result, the proposed activation function can approximate the desired function.

5. Conclusions

In this paper, we have proposed the simultaneous learning method for both the activation functions and the connection weights. The disturbance method, in order to avoid falling into local minima is also introduce. The optimum functions for the given problems can be approximated. If an arbitrary function can be realized in a single unit, the number of hidden units can be drastically reduced. Furthermore, difficult problems for MLNNs with fixed activation functions can be solved with high probabi lity and fast convergence speed.

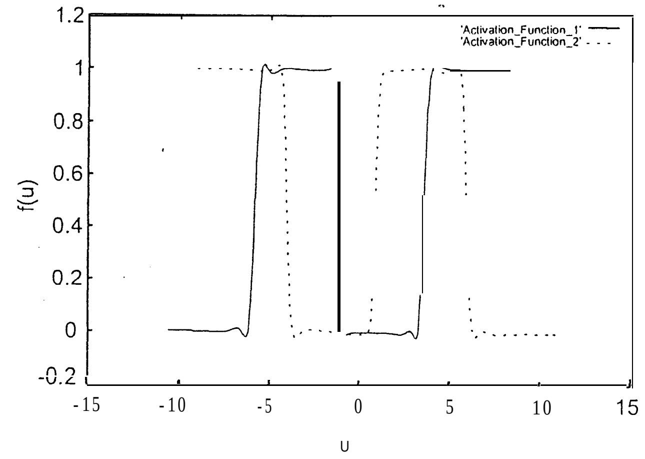

Fig.7 Activation functions of two-class pattern classification. Horizontai axis is input and vertical axis output.

References

[1] J.Sietsma and R.J.F.Dow, *Neural net pruning-Why and how,* Proc. IEEE ICNN'88, pp.325333, 1988.

[2] J.Sietsma and R.J.F.Dow, *Creating artificial neural networks that generalize,* Neural Networks, vol.4, pp.67-79, 1991.

[3] K.Nakayama and Y.Kimura, *Optimization of activation functions in multilayer neural network,* Proc. IEEE ICNN'94, Orlando, pp.43 1-436, June 1994.

[4] K.Hara and K.Nakayama, *Comparison of activation functions in multilayer neural network for pattern classification,* Proc. ICANN'94, Sorrento, pp.819-822, May 1994.

[5] J.C.Zhang, M.Zhang and J.Fulcher, *Financial prediction using higher order trigonometric polynomial neural network group model,* Proc. IEEE ICNN'97, Houston, pp.2231-2234, June 1997.

[6] C.T.Chen and W.D.Chabg, *A feedforward neural network with function shape autotuning,* Neural Networks, vo1.9, no.4, pp.627-641, 1996.

[7] Y.Wu, M.Zhao and X.Ding, *Beyond weights adaptation: A new neural model with trainable activation function and its supervised learning,* Proc. IEEE ICNN'97, 'Houston, pp.1152-1157, June 1997.

[8] T.Burg and N.T-Gut-man, *An extended neuron model for efficient time-series generation and prediction,* Proc ICANN'97, Lausanne, pp.1005-1010, Oct. 1997.