A Multilayer Neural Network with Nonlinear Inputs and Trainable Activation Functions: Structure and Simultaneous Learning Algorithm

Kenji Nakayama Akihiro Hirano Issei Ido
Dept. of Elec and Comp Eng., Kanazawa Univ., 2-40-20, Kodatsuno, Kanazawa, 920-8667, Japan
E-mail:nakayama@t.kanazawa-u.ac.jp

Abstract

Network size of neural networks is highly dependent on activation functions. A trainable activation function has been proposed, which consists of a linear combination of some basic functions. The activation functions and the connection weights are simultaneously trained. An 8 bit parity problem can be solved by using a single output unit and no hidden unit. In this paper, we expand this model to multilayer neural networks. Furthermore, nonlinear functions are used at the unit inputs in order to realize more flexible transfer functions. The previous activation functions and the new nonlinear functions are also simultaneously trained. More complex pattern classification problems can be solved with a small number of units and fast convergence.

1. Introduction

In designing neural networks, fast learning, high probability of convergence and small network size are very important. They are highly dependent on network models, learning algorithms and problems to be solved. The transfer function from the inputs to the unit output also plays a very important role in this issue.

Many pruning methods for hidden units have been proposed [1],[2]. A method, which combine processes of selecting activation functions and pruning hidden units, was proposed for multilayer neural networks [3]. Effective activation functions can be selected, with which the number of the hidden units is well reduced. This method, however, belongs to the pruning methods. Thus, a relatively large number of hidden units are required at the beginning of learning. 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, trainable properties of the functions are rather limited.

A trainable activation function has been proposed, which consists of a linear combination of some basic functions [9]. This activation function is trained together with the connection weights by the gradient descent algorithm. An 8-bit parity check problem can be effectively solved. Since this approach employs linear connection weights, a transfer function from the inputs and the output of the unit is rather limited.

In this paper, we further develop the previous method to have a nonlinear function at the unit input. Its coefficients are trained together with the previous activation functions. Computer simulation using several pattern classification problems will be demonstrated to confirm usefulness.

2. Network Architecture

2.1 Network Equations

Although the proposed nonlinear input-functions and trainable activation functions can be applied to any network structures, multilayer neural networks are taken into account in this paper. The two-layer neural network is considered.

Input patterns:

$$\mathbf{x} = [x_0, x_1, ..., x_I]^T, \ x_0 = 1 \tag{1}$$

Hidden layer:

$$\mathbf{w}_{hj} = [w_{hj0}, w_{hj1}, ..., w_{hjI}]^T \tag{2}$$
$$u_{hj} = g_{hj}(\mathbf{x}, \mathbf{w}_{hj}) \tag{3}$$
$$y_{hj} = f_{hj}(u_{hj}) \tag{4}$$

Output layer:

$$\mathbf{y}_h = [y_{h0}, y_{h1}, ..., y_{hJ}]^T, \ y_{h0} = 1 \tag{5}$$
$$\mathbf{w}_k = [w_{k0}, w_{k1}, ..., w_{kJ}]^T \tag{6}$$
$$u_k = g_k(y_h, \mathbf{w}_k) \tag{7}$$
$$y_k = f_k(u_k) \tag{8}$$

$g_{hj}()$ and $g_k()$ are nonlinear functions, and $f_{hj}()$ and $f_k()$ are activation functions.

2.2 Activation Functions

The trainable activation function proposed in [9] is described here for convenience. A composite form activation functions are used, which combine several basic functions. A sigmoidal function is used as the basic function. The reason why the sigmoidal function is selected is the following. The Gaussian and sinusoidal functions can be composed of several sigmoidal functions. However, the reverse approximation requires a large number of the Gaussian and sinusoidal functions.

The proposed activation function is expressed in Eq.(9), which is a linear combination of several sigmoidal functions.

$$y = f(u) = \sum_{l=1}^{L}\{\frac{a_l}{1+e^{-b_l u - c_l}} + d_l\} \tag{9}$$

This activation function includes four parameters a_l, b_l, c_l and d_l in each basic function. Thus, $4L$ free parameters are used in one activation function. They will be optimized together with the nonlinear functions.

3. Nonlinear Input Function

3.1 Transfer Function

In the case of the hidden units, the transfer function from the inputs $\mathbf{x} = [x_0, x_1, ..., x_I]^T$ to the unit output y_{hj} is given by Eqs.(3) and (4). Now, letting u_0 be some constant of u_{hj}, it has the same value on the hyper-plane given by

$$u_0 = g_{hj}(\mathbf{x}, \mathbf{w}_{hj}) \tag{10}$$

On this hyper plane, y_{hj} also has the same value given by

$$y_0 = f_{hj}(u_0) \tag{11}$$

Thus, the shape of the hyper-plane in the I-dimensional x-space and the curve of the activation functions determine the transfer function from $\mathbf{x}$ to y_{hj}.

The input potential u_{hj} in the previous work [9] is the linear combination of the inputs. This causes some limitation on the relation between the inputs and the output of the units.

3.2 Polynomial Function

What kinds of nonlinear functions are useful is highly dependent on learning algorithm. It is very important to optimize the nonlinear functions together with the activation functions in a simple way. For this purpose, we introduce a high-order polynomial function. One example for Eq.(3) is shown here.

$$u_{hj} = w_{hj0} + w_{hj1}x_1^2 + w_{hj2}x_1x_2 + w_{hj3}x_2^2 \tag{12}$$

This nonlinear function is a linear combination of the coefficients to be optimized. This will derive an efficient learning algorithm.

Generally, we express the polynomial function as follows:

$$u_{hj} = \sum_{i=1}^{\hat{I}} w_{hji}\hat{x}_i \tag{13}$$

$$u_k = \sum_{j=1}^{\hat{J}} w_{kj}\hat{y}_{hj} \tag{14}$$

$\hat{x}_i$ and $\hat{y}_{hj}$ include high-order terms of x_i and y_{hj}. $\hat{I}$ and $\hat{J}$ are the number of the terms in the polynomial functions.
Examples Examples for the linear combination and the 2nd-order polynomial are shown here.
Linear Combination:

$$u_{hj} = w_{h0} + w_{h1}x_1 + w_{h2}x_2 \tag{15}$$

Second-order Polynomial:

$$u_{hj} = w_{h0} + w_{h1}x_1 + w_{h2}x_1^2 + w_{h3}x_2 + w_{h4}x_2^2 \tag{16}$$

Contour lines are shown in Figs.1 and 2. As previously described, u_{hj} and then y_{hj} takes the same value on the same contour. In Fig.1, the u_{hj}-axis is shown with a dashed line. The activation function $f_{hj}(u_{hj})$ has its own shape on this axis.

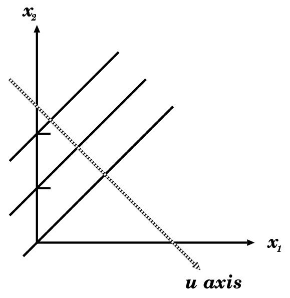

Fig. 1. Contour lines for linear combination.

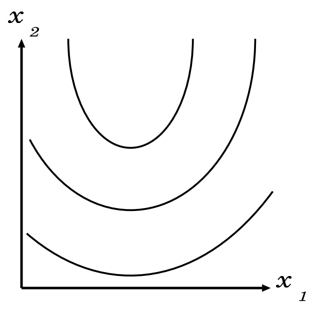

Fig. 2. Contour lines for 2nd-order polynomial.

3.3 Network Structure

Based on the previous discussion, we propose a multilayer neural network, which includes the polynomial input functions and the trainable activation functions, as shown in Fig.3. The nonlinear blocks generate the high-order terms of $\mathbf{x}$. Furthermore, by passing the high-order terms through the linear combination part, the polynomial is composed.

4. Simultaneous Learning Algorithm

4.1 Learning Algorithm

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 y_k, the mean squared error is given by,

$$E = \frac{1}{K}\sum_{k=1}^{K}(d_k - y_k)^2 \tag{17}$$

Furthermore, let $p(n)$ be parameters of the activation functions and the connection weights, where n is the iteration

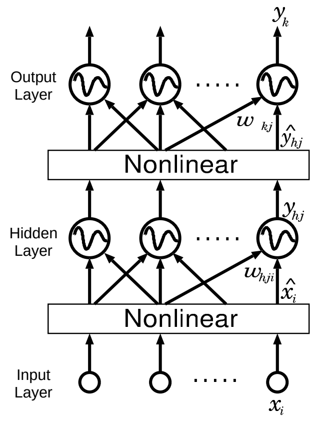

Fig. 3. Proposed network structure with polynomial inputs and trainable activation functions.

number. $p(n)$ is updated as follows:

$$p(n+1) = p(n) - \eta \frac{\partial E}{\partial p(n)} \tag{18}$$

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

$$p(n+1) = p(n) + \Delta p(n) \tag{19}$$

Due to the page limitation, the correction terms $\Delta p(n)$ are summarized in the following.

Activation functions in output layer:

$$\begin{aligned}
\Delta a_{kl}(n) &= \eta \delta_k \phi_{kl} + \alpha \Delta a_{kl}(n-1) && (20)\\
\Delta b_{kl}(n) &= \eta \delta_k a_{kl}(n) u_k \phi_{kl}(1-\phi_{kl}) \\
&\quad + \alpha \Delta b_{kl}(n-1) && (21)\\
\Delta c_{kl}(n) &= \eta \delta_k a_{kl}(n) \phi_{kl}(1-\phi_{kl}) \\
&\quad + \alpha \Delta c_{kl}(n-1) && (22)\\
\Delta d_{kl}(n) &= \eta \delta_k + \alpha \Delta d_{kl}(n-1) && (23)\\
\delta_k &= d_k(n) - y_k(n) && (24)\\
\phi_{kl} &= \phi(b_{kl}(n) u_k + c_{kl}(n)) && (25)\\
\phi(x) &= \frac{1}{1+e^{-x}} && (26)
\end{aligned}$$

a_{kl}, b_{kl}, c_{kl} and d_{kl} are the parameters in $f_k()$. α is a learning rate of the momentum term.

Connection weights from hidden layer to output layer:

$$\begin{aligned}
\Delta w_{kj}(n) &= \eta \delta_k \hat{y}_{hj} \sum_{l=1}^{L} [a_{kl(n)} b_{kl}(n) \phi_{kl}(1-\phi_{kl})] \\
&\quad + \alpha \Delta w_{kj}(n-1) && (27)
\end{aligned}$$

Activation function in hidden layer:

$$\begin{aligned}
\Delta a_{jl}(n) &= \eta \phi_{jl} \\
&\quad \cdot \sum_{k=1}^{K} [\delta_k w_{kj} \sum_{l=1}^{L} (a_{kl}(n) b_{kl}(n) \phi_{kl}(1-\phi_{kl})] \\
&\quad + \alpha \Delta a_{jl}(n-1) && (28)
\end{aligned}$$

$$\phi_{jl} = \phi(b_{jl}(n) u_{hj} + c_{jl}(n)) \tag{29}$$

$$\begin{aligned}
\Delta b_{jl}(n) &= \eta a_{jl} u_{hj} \phi_{jl}(1-\phi_{jl}) \sum_{k=1}^{K} [\delta_k w_{kj} \\
&\quad \cdot \sum_{l=1}^{L} (a_{kl} b_{kl} \phi_{kl}(1-\phi_{kl})] \\
&\quad + \alpha \Delta b_{jl}(n-1) && (30)
\end{aligned}$$

$$\begin{aligned}
\Delta c_{jl}(n) &= \eta a_{jl}(n) \phi_{jl}(1-\phi_{jl}) \sum_{k=1}^{K} [\delta_k w_{kj} \\
&\quad \cdot \sum_{l=1}^{L} a_{kl}(n) b_{kl}(n) \phi_{kl}(1-\phi_{kl})] \\
&\quad + \alpha \Delta c_{jl}(n-1) && (31)
\end{aligned}$$

$$\begin{aligned}
\Delta d_{jl}(n) &= \eta \sum_{k=1}^{K} [\delta_k w_{kj} \\
&\quad \cdot \sum_{l=1}^{L} a_{kl}(n) b_{kl}(n) \phi_{kl}(1-\phi_{kl})] \\
&\quad + \alpha \Delta d_{jl}(n-1) && (32)
\end{aligned}$$

a_{jl}, b_{jl}, c_{jl} and d_{jl} are the parameters in $f_{hj}()$.

Connection weights from input layer to hidden layer:

$$\begin{aligned}
\Delta w_{hji}(n) &= \eta \hat{x}_i \sum_{l=1}^{L} a_{jl}(n) b_{jl}(n) \phi_{jl}(1-\phi_{jl}) \\
&\quad \cdot \sum_{k=1}^{K} [\delta_k w_{kj} \sum_{l=1}^{L} a_{kl}(n) b_{kl}(n) \phi_{kl}(1-\phi_{kl})] \\
&\quad + \alpha \Delta w_{hji}(n-1) && (33)
\end{aligned}$$

4.2 Acceleration of Learning Process

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, the control methods have been proposed [9].

5. Simulation Results

5.1 Example 1: Ring in Square

Figure 4 shows a ring in square problem. The data in the ring are classified into a class C_1, and the others into a class C_2. This figure shows the targets for the outputs.

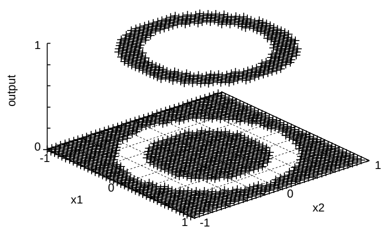

Fig. 4. Ring in square problem.

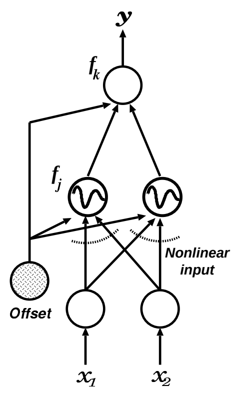

Fig. 5. Proposed network applied to ring in square problem.

The network used for this problem is shown in Fig.5. The nonlinear input function is a 3rd-order function. The trainable activation function, used in the hidden units, includes 2 basic functions. The output unit has a fixed sigmoidal function. 1200 training data are randomly selected from two classes shown in Fig.4. The training was stopped if the mean square error (MSE) for all training data is less than 0.001. The learning curves are shown in Figs.6 and 7.

The multilayer neural network with fixed sigmoidal units

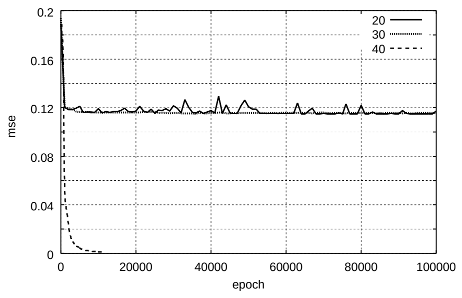

Fig. 6. Learning curves by using multilayer network with 20, 30 and 40 hidden units.

requires 40 hidden units and 10,000 epochs for convergence. The new method requires 2 hidden units and 300 epochs.

Figures 8 and 9 show the nonlinear input functions of the 1st and 2nd hidden units. The activation functions for the 1st and 2nd hidden units are shown in Figs.10 and 11. Combining them, the outputs of the 1st and 2nd hidden units are obtained as shown in Figs.12 and 13. Finally, they are combined at the output unit resulting in the final output, which is almost the same as the target shown in Fig.4.

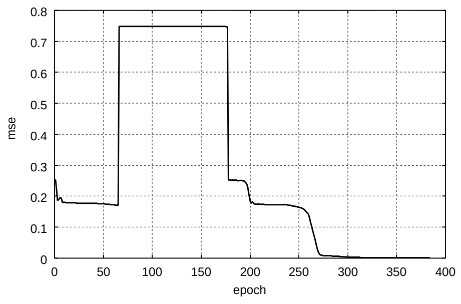

Fig. 7. Learning curve by using proposed method.

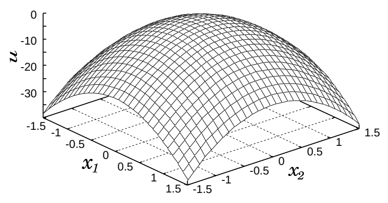

Fig. 8. Nonlinear input-function of 1st hidden unit.

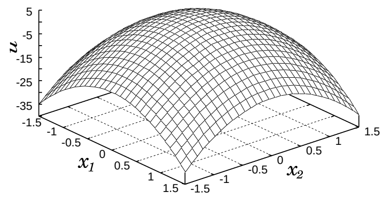

Fig. 9. Nonlinear input-function of 2nd hidden unit.

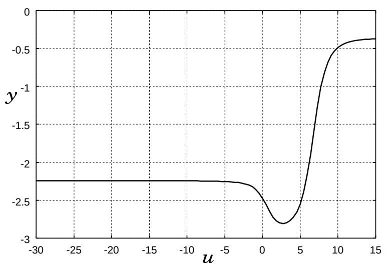

Fig. 10. Activation function of 1st hidden unit.

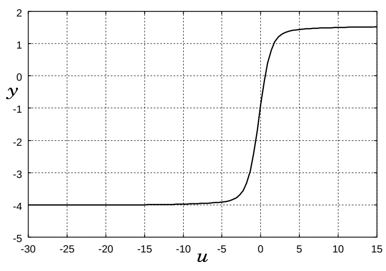

Fig. 11. Activation function of 2nd hidden unit.

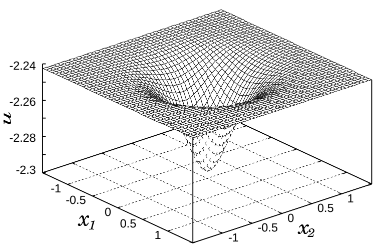

Fig. 12. Output of 1st hidden unit.

5.2 Example 2: Double rings in square

A double rings in square problem is shown in Fig.14. The multilayer neural network with 40 fixed sigmoidal hidden units and the proposed method with the same network as in Example 1 are applied to this problem. The learning curves are shown in Fig.15. The proposed method requires 2,500 epochs. In the case of the MLNN, 25 percent of the trials converged. 25,000 epochs are required.

From these simulation results, the proposed method is very useful to reduce the network size and to achieve fast convergence.

6. Conclusion

A multilayer neural network with the nonlinear input functions and the trainable activation functions has been proposed. They are simultaneously trained. Simulation results using complicated pattern classification demonstrate small size network and fast convergence are achieved by the proposed method.

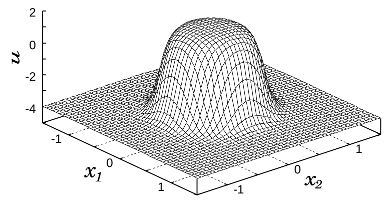

Fig. 13. Output of 2nd hidden unit.

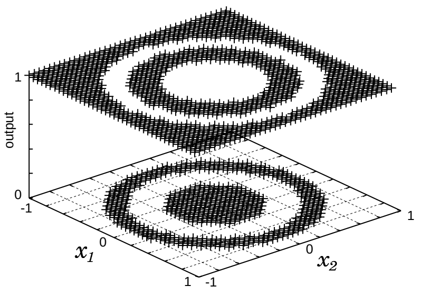

Fig. 14. Double rings in square problem.

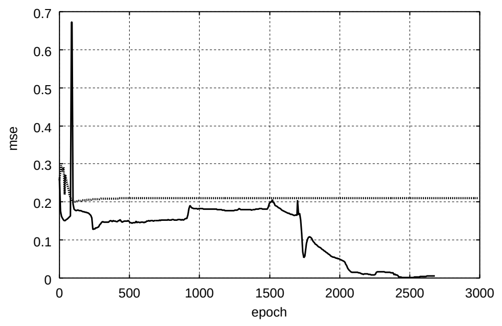

Fig. 15. Learning curves for MLNN (dotted line) and proposed method (solid line) in double rings in square problem.

References

[1] J.Sietsma and R.J.F.Dow,"Neural net pruning-Why and how," Proc. IEEE ICNN'88, pp.325-333, 1988.

[2] J.Sietsma and R.J.F.Dow,"Creating artificial neural networks that generalize," INNS 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.431-436, 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," INNS Neural Networks, vol.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-Gurman,"An extended neuron model for efficient time-series generation and prediction," Proc ICANN'97, Lausanne, pp.1005-1010, Oct. 1997.

[9] K.Nakayama and M.Ohsugi,"A simultaneous learning method for both activation functions and connection weights of multilayer neural networks", Proc. IEEE INNS IJCNN'98, Anchorage, pp.2253-2257, May 1998.