Improvement of Training Method for Dynamic Neural Network

Kunihiko Nakazono
University of the Ryukyus
Senbaru 1, Nishihara, Okinawa. 903–0213
nakazono@mibai.tec.u-ryukyu.ac.jp

Kouhei Ohnishi
Keio University
Hiyoshi 3–14–1, Yokohama. 222–8522
ohnishi@sd.keio.ac.jp

Hiroshi Kinjo
University of the Ryukyus
Senbaru 1, Nishihara, Okinawa. 903–0213
kinjo@tec.u-ryukyu.ac.jp

Tetsuhiko Yamamoto
University of the Ryukyus
Senbaru 1, Nishihara, Okinawa. 903–0213
yamamoto@tec.u-ryukyu.ac.jp

Abstract

In this research, we propose a dynamic neural network (DNN) with the characteristics of stiffness, viscosity, and inertia and a training algorithm based on the back-propagation (BP) method to include a momentum term. In a previous research, we proposed a training algorithm for the DNN based on the BP method or GA-based training method. However, in the previous method it was necessary to determine the values of the DNN parameters by trial and error. So, the modified BP method and GA-based training method were designed to train not only the connecting weights but also the property parameters of the DNN.

We develop the BP method to include a momentum term in order to increase the convergence of the training effect. Simulation results show that the DNN with characteristics of stiffness, viscosity, and inertia trained by the modified BP method to include the momentum term obtains good training performances for time series signals generated from periodic function. In this paper, we compare the DNN with a conventional training method in order to verify the effectiveness of the DNN.

1 Introduction

In recent years, recurrent neural networks and spiking neural networks have attracted more research interest than layered neural networks having simple structure [1, 2, 3, 4]. The recurrent neural network is a possible candidate for improving the system dynamics because it takes time delayed inputs into consideration and incorporates a feedback structure in the neuron unit. Research on spiking neural networks is also ongoing. Spiking neural networks treat spike trains and process the signals based on spike pulses. However, the network structure in recurrent neural networks and spiking neural networks is complex compared to that in layered neural networks with a training method.

Here, we propose a dynamic neural network (DNN) that realizes a dynamic property and has a network structure with the properties of stiffness, viscosity, and inertia without time delayed input elements. In a previous research, the DNN was constructed with a training algorithm that used error back-propagation (BP) method [5]. However, the BP method updated only the connecting weights for the DNN, and the property parameters for the DNN had to be decided by trial and error. Therefore, we designed a GA-based training method [6] to train both the connecting weights and the parameters of the DNN. But the GA-based training method took the executing time of computer program that evolved in GA simulation [7, 8]. We developed the modified BP method to train both the connecting weights and the parameters [9].

In the present paper, we design the BP method to include a momentum term in order to increase the convergence of the training effect. The effectiveness of the proposed DNN is verified by identifying time series signals [5, 7, 8, 9]. Simulation results show that the proposed DNN provides higher performance than the conventional method.

2 Structure of DNN

In this research, we propose a DNN using a neuron unit having the properties of stiffness, viscosity, and inertia without time delayed input elements. In the neuron unit, we assume that the output from the neuron possesses the properties of stiffness, viscosity, and inertia, and that the output is propagated in the next neuron unit. The proposed DNN is composed of three hierarchy layers, and the proposed neuron adopts only in a hidden layer. The structure of the DNN is shown in Figure 1.

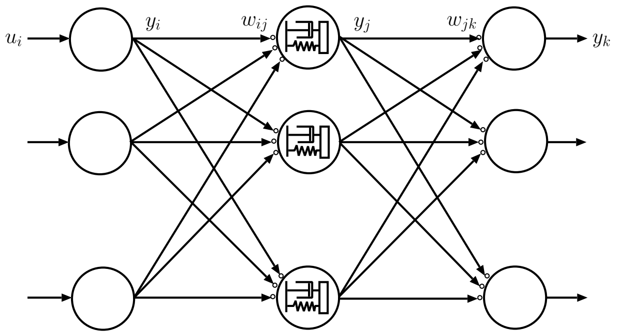

Fig. 1 Structure of DNN

The equations for the DNN are expressed as follows:

$$y_i = u_i, \quad (i = 1, 2, \cdots, N_I) \tag{1}$$

$$y_j = K_j f_j(\mathrm{net}_j) + D_j \dot{f}_j(\mathrm{net}_j) + M_j \ddot{f}_j(\mathrm{net}_j) \tag{2}$$

$$\mathrm{net}_j = \sum_{i=1}^{N_I} w_{ij} y_i, \quad (j = 1, 2, \cdots, N_J) \tag{3}$$

$$y_k = f_k(\mathrm{net}_k) \tag{4}$$

$$\mathrm{net}_k = \sum_{j=1}^{N_J} w_{jk} y_j, \quad (k = 1, 2, \cdots, N_K) \tag{5}$$

where u_i is the input value to the DNN, and y_i, y_j, and y_k are the output values in the input, hidden, and output layers, respectively. The connecting weight from unit i in the input layer to unit j in the hidden layer is denoted by w_{ij}. Similarly, w_{jk} is a connecting weight from unit j in the hidden layer to unit k in the output layer. The total sum of the products of the connecting weight w_{ij} and w_{jk} and the output value is denoted by net_j and net_k, respectively. M_j, D_j, and K_j are the property parameters of inertia, viscosity, and stiffness, respectively, and N_I, N_J, and N_K are the number of neurons in the input, hidden, and output layers, respectively. The activation function $f_j(x)$ in the hidden layer uses a sigmoid function in range of $[-1, 1]$. Also the activation function $f_k(x)$ in the output layer is a linear function.

3 BP-based training method

First, we derive a minimizing sequence of the measurement of error function E:

$$E = \frac{1}{2}\sum_{k=1}^{N_K} e_k^2 = \frac{1}{2}\sum_{k=1}^{N_K}(d_k - y_k)^2 \tag{6}$$

where d_k is the desired signal. In order to minimize the measurement of the error function E of equation (6), both of the connecting weights and the property parameters of the DNN are modified.

The BP-based training method is shown in section 3. The connecting weights and the property parameters of the DNN are updated sequentially based on the steepest descent method.

$$\Delta w_{ij} = w_{ij}^{(\mathrm{new})} - w_{ij}^{(\mathrm{old})} = -\varepsilon \frac{\partial E}{\partial w_{ij}} \tag{7}$$

$$\Delta w_{jk} = w_{jk}^{(\mathrm{new})} - w_{jk}^{(\mathrm{old})} = -\varepsilon \frac{\partial E}{\partial w_{jk}} \tag{8}$$

$$\Delta K_j = K_j^{(\mathrm{new})} - K_j^{(\mathrm{old})} = -\varepsilon \frac{\partial E}{\partial K_j} \tag{9}$$

$$\Delta D_j = D_j^{(\mathrm{new})} - D_j^{(\mathrm{old})} = -\varepsilon \frac{\partial E}{\partial D_j} \tag{10}$$

$$\Delta M_j = M_j^{(\mathrm{new})} - M_j^{(\mathrm{old})} = -\varepsilon \frac{\partial E}{\partial M_j} \tag{11}$$

ε is the training rate ($\varepsilon > 0$).

$\partial E/\partial w_{jk}$ and $\partial E/\partial w_{ij}$ are described as follows.

$$\frac{\partial E}{\partial w_{jk}} = \frac{\partial E}{\partial \mathrm{net}_k} \cdot \frac{\partial \mathrm{net}_k}{\partial w_{jk}} = \frac{\partial E}{\partial \mathrm{net}_k} \cdot y_j \tag{12}$$

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial \mathrm{net}_j} \cdot \frac{\partial \mathrm{net}_j}{\partial w_{ij}} = \frac{\partial E}{\partial \mathrm{net}_j} \cdot y_i \tag{13}$$

In the upper expression, the derivations are defined as

$$\delta_k = \frac{\partial E}{\partial \mathrm{net}_k} \tag{14}$$

$$\delta_j = \frac{\partial E}{\partial \mathrm{net}_j} \tag{15}$$

and δ_k and δ_j are calculated, respectively. First, δ_k is expanded as

$$\begin{aligned}\delta_k &= \frac{\partial E}{\partial \mathrm{net}_k} = \frac{\partial E}{\partial e_k} \cdot \frac{\partial e_k}{\partial y_k} \cdot \frac{\partial y_k}{\partial \mathrm{net}_k} \\ &= -e_k f_k'(\mathrm{net}_k)\end{aligned} \tag{16}$$

δ_k finally becomes equation (16).

When δ_j is calculated in the same way, it becomes Equation (17).

$$\begin{aligned}\delta_j &= \frac{\partial E}{\partial \text{net}_j} \\ &= \sum_{k=1}^{N_K}\left(\frac{\partial E}{\partial e_k}\cdot\frac{\partial e_k}{\partial y_k}\cdot\frac{\partial y_k}{\partial \text{net}_k}\right)\cdot\frac{\partial \text{net}_k}{\partial y_j}\cdot\frac{\partial y_j}{\partial \text{net}_j} \\ &= \sum_{k=1}^{N_K}\delta_k\cdot w_{jk}\cdot\frac{\partial y_j}{\partial \text{net}_j} \qquad (17)\end{aligned}$$

$\partial y_j/\partial \text{net}_j$ is expressed as equations (18)–(21).

$$\begin{aligned}&\frac{\partial y_j}{\partial \text{net}_j} \\ &= K_j\frac{\partial f_j(net_j)}{\partial \text{net}_j}+D_j\frac{\partial \dot{f}_j(\text{net}_j)}{\partial \text{net}_j}+M_j\frac{\partial \ddot{f}_j(\text{net}_j)}{\partial \text{net}_j} \\ &\qquad (18)\end{aligned}$$

$$\begin{aligned}\frac{\partial f_j(\text{net}_j)}{\partial \text{net}_j} &= f_j'(\text{net}_j) \qquad (19)\\ \frac{\partial \dot{f}_j(\text{net}_j)}{\partial \text{net}_j} &= f_j''(\text{net}_j)\cdot\dot{\text{net}}_j \qquad (20)\\ \frac{\partial \ddot{f}_j(\text{net}_j)}{\partial \text{net}_j} &= f_j'''(\text{net}_j)\cdot\dot{\text{net}}_j^2+f_j''(\text{net}_j)\cdot\ddot{\text{net}}_j \\ &\qquad (21)\end{aligned}$$

Next, the derivation of $\partial E/\partial K_j$ is described as follows.

$$\begin{aligned}\frac{\partial E}{\partial K_j} &= \sum_{k=1}^{N_K}\frac{\partial E}{\partial \text{net}_k}\cdot\frac{\partial \text{net}_k}{\partial y_j}\cdot\frac{\partial y_j}{\partial K_j} \\ &= f_j(net_j)\cdot\sum_{k=1}^{N_K}w_{jk}\delta_k \\ &= f_j(net_j)\gamma_j \qquad (22)\end{aligned}$$

where $\gamma_j=\sum_{k=1}^{N_K}w_{jk}\delta_k$.

When the property parameters D_j and M_j for the error function E are calculated in the same way, it becomes Equations (23) and (24).

$$\begin{aligned}\frac{\partial E}{\partial D_j} &= \sum_{k=1}^{N_K}\frac{\partial E}{\partial \text{net}_k}\cdot\frac{\partial \text{net}_k}{\partial y_j}\cdot\frac{\partial y_j}{\partial D_j} \\ &= \dot{f}_j(net_j)\gamma_j \qquad (23)\\ \frac{\partial E}{\partial M_j} &= \sum_{k=1}^{N_K}\frac{\partial E}{\partial \text{net}_k}\cdot\frac{\partial \text{net}_k}{\partial y_j}\cdot\frac{\partial y_j}{\partial M_j} \\ &= \ddot{f}_j(\text{net}_j)\gamma_j \qquad (24)\end{aligned}$$

In order to increase the convergence of the training effect, we modify the equations (7)–(11) to include a momentum term. These equations are described as follows.

$$\begin{aligned}\Delta w_{ij}(t+1) &= -\varepsilon\delta_j y_i+\eta\Delta w_{ij}(t) \qquad (25)\\ \Delta w_{jk}(t+1) &= -\varepsilon\delta_k y_j+\eta\Delta w_{jk}(t) \qquad (26)\\ \Delta K_j(t+1) &= -\varepsilon f_j(\text{net}_j)\gamma_j+\eta_k\Delta K_j(t) \qquad (27)\\ \Delta D_j(t+1) &= -\varepsilon \dot{f}_j(\text{net}_j)\gamma_j+\eta_d\Delta D_j(t) \qquad (28)\\ \Delta M_j(t+1) &= -\varepsilon \ddot{f}_j(\text{net}_j)\gamma_j+\eta_m\Delta M_j(t) \qquad (29)\end{aligned}$$

where t indexes the present time, and η, η_k, η_d, and η_m are small constant values.

In this way, the connecting weights and property parameters of the DNN are updated based on the concept of the modified BP algorithm to include the momentum term.

4 Numerical simulation

The effectiveness of the DNN proposed in the paper is verified by numerical simulation in order to identify a periodic function. The DNN is structured to have a single input and single output (SISO). The method by which a time series signal can be identified is shown in Figure 2. The desired signal, namely the training data $d(t)$, is the signal that has passed one sampling period prior to the input signal $u(t)$.

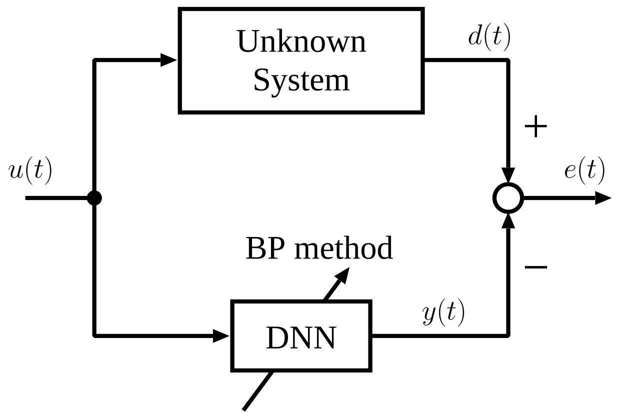

Fig. 2 Identification of signals

In order to facilitate analysis, simulation shows that the DNN identified the time series signal of the single sine periodic function with cycle T as

$$u(t)=\sin\left(\frac{2\pi t}{T}\right). \qquad (30)$$

In numerical simulation, the number of neurons N_J in the hidden layer is 5 units, and the cycle T equals 16. The initial range of the connecting weights of the DNN is set to $[-0.3, 0.3]$ at random, and the initial range of the property parameters is set to 1.0. The training rate is set to $\varepsilon = 0.1$, and the momentum rates are set to $\eta = 0.0005$, $\eta_k = \eta_d = \eta_m = 0.0$, respectively.

The training involved 1,000 iterations. The result of the error function E is shown in Figure 3.

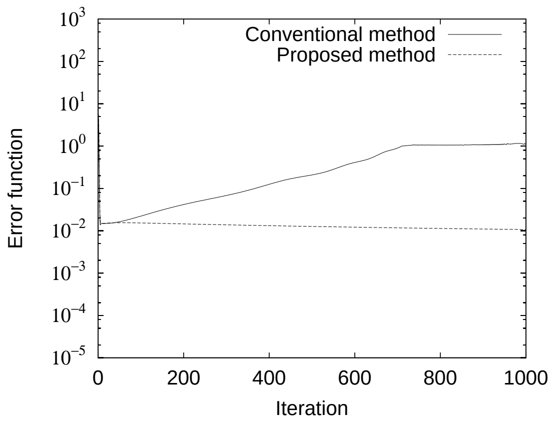

Fig. 3 Iteration result

The result of the simulation shows that the proposed DNN provides good performance compared with the conventional method without momentum term ($\eta = 0.0$, $\eta_k = \eta_d = \eta_m = 0.0$). The error function E decreases gradually and E almost converge about 0.01. The output of the DNN deviated negligibly from the desired signal.

5 Conclusion

In the present paper, the proposed DNN, showing the validity of dynamic neuron with properties of stiffness, viscosity, and inertia, was structured. The training algorithm adopt the modified BP method. We designed a modified BP method to include a momentum term in order to train both the connecting weights and the parameters of the DNN. Simulation results showed that the DNN trained by the BP method realized good training performance compared with the conventional method for time series signals generated from a periodic function.

References

1 D. E. Rumelhart, J. L. McClelland, and PDP Research Group (1989), Parallel distributed processing, MIT Press.

2 R. J. Williams and D. Zipser (1989), A Learning algorithm for continually running fully recurrent neural networks, Neural Computation, 1, No.2, pp.270–280.

3 H. Kinjo, K. Nakazono, and T. Yamamoto (1997), Pattern Recognition for time series signals using recurrent neural networks by genetic algorithms (in Japanese), Trans. of ISCIE, Vol. 10, No. 6, pp.304–314.

4 W. Mass and C. M. Bishop (1999), Pulsed neural networks, pp. 16–53, MIT Press.

5 K. Nakazono, K. Ohnishi, H. Kinjo, and T. Yamamoto (2003), Identification of periodic function using dynamical neural network, AROB 8th '03, Vol.2, pp. 633–636.

6 Edited by L. Davis (1991), Handbook of genetic algorithms, Van Nostrand Reinhold, New York.

7 K. Nakazono, K. Ohnishi, and H. Kinjo (2004), System identification using dynamical neural network with GA-based training, AROB 9th '04, Vol.1, pp. 75–78.

8 K. Nakazono, K. Ohnishi, and H. Kinjo (2005), Identification of time series signals using dynamical neural network with GA-based training, AROB 10th '05, CD-ROM, pp. 25–28.

9 K. Nakazono, K. Ohnishi, and H. Kinjo (2006), Identification using dynamical neural network with modified BP method, AROB 11th '06, CD-ROM, pp. 126–129.