A supervised learning rule adjusting input-output pulse timing for pulsed neural network

Makoto Motoki
motoki@graduate.chiba-u.jp

Seiichi Koakutsu
koakutsu@faculty.chiba-u.jp

Hironori Hirata
hiro@faculty.chiba-u.jp

Graduate School of Science and Technology
Chiba University
1-33 Yayoi-cho, Inage-ku, Chiba-shi, 263-8522, Japan

Abstract

Some supervised learning rules have been already proposed for pulsed neural network (PNN). Those learning rules use information expression with pulse frequency, however, does not use that with pulse timing. Therefore, the conventional learning rules cannot learn pulse timing, although it is important to adjust input/output (I/O) pulse timing of network. The purpose of this paper is to propose a supervised learning rule adjusting I/O pulse timing for PNN. In the proposed method, pulse timing error and pulse frequency error are corrected by adjusting transmission delay and synaptic weight among neurons. Results of computational experiments indicate that the proposed method enables learning of pulse timing in multi-layered feedforward PNN.

1 Introduction

Pulsed Neural Network (PNN) has been getting attention recently as one of neural network models which is proper for temporal data processing. As for learning for PNN, Hebbian learning has been proposed[1][2]. Hebbian learning is one of unsupervised learning rules, and enables to optimize the internal state of network. However, Hebbian learning cannot construct a network which can represent required input/output (I/O) temporal data mapping. In order to construct a network which can represent required I/O mapping, supervised learning rules are necessary. Some supervised learning rules have been already proposed for PNN[3][4]. Those learning rules use information expression with pulse frequency, however, does not use that with pulse timing. Therefore, the conventional learning rules cannot learn pulse timing, although it is important to adjust I/O pulse timing of network.

The purpose of this paper is to propose a supervised learning rule adjusting I/O pulse timing for PNN. In the proposed method, pulse timing error and pulse frequency error are corrected by adjusting transmission delay and synaptic weight among neurons. We verify the effectiveness of the proposed method by computer simulation. Results of computational experiments indicate that the proposed method enables learning of pulse timing in multi-layered feedforward PNN.

2 Pulsed neural network

This study uses a leaky integrate-and-fire neuron model which is the same model used by Gerstner et al[1]. and Eurich et al[2]., so the behavior of this neuron model is explained briefly.

A presynaptic neuron j and a postsynaptic neuron i is connected with a synaptic weight $w_0^{i,j}$ by synapse. A set of firing time T^j of the neuron j and a set of arrival time $T_{\mathrm{d}}^{i,j}$ at which pulses of the neuron j reach the neuron i are defined as follows:

$$T^j = \{t_\nu^j; 1 \le \nu \le n^j\} = \{t \,|\, y^j(t) = 1\}, \tag{1}$$
$$T_{\mathrm{d}}^{i,j} = \{t_{\mathrm{d},\nu}^{i,j} = t_\nu^j + \tau_{\mathrm{d}}^{i,j}; 1 \le \nu \le n^j\}, \tag{2}$$

where n^j is the maximum number of pulses of the neuron j, $y^j(t)$ is the output of the neuron j at time t, and $\tau_{\mathrm{d}}^{i,j}$ is transmission delay between the neuron j and i.

When the pulse of the neuron j arrives, the membrane potential $P_{\mathrm{m}}^i(t)$ of the neuron i changes as follows:
in case of $t > t_\mu^i + \tau_{\mathrm{r}}^i$,

$$\tau_{\mathrm{m}}^i \frac{dP_{\mathrm{m}}^i(t)}{dt} = -P_{\mathrm{m}}^i(t) + \sum_{j \in J} w_0^{i,j} \sum_{t_{\mathrm{d},\nu}^{i,j} \in T_{\mathrm{d}}^{i,j}} \delta\left(t - t_{\mathrm{d},\nu}^{i,j}\right), \tag{3}$$

and in case of $t \le t_\mu^i + \tau_{\mathrm{r}}^i$,

$$P_{\mathrm{m}}^i(t) = 0, \tag{4}$$

where τ_{r}^i is absolute refractory period, τ_{m}^i is depression time constant, J is a set of presynaptic neurons of the neuron i, $w_0^{i,j}$ is synaptic weight, and $\delta(t)$ is Dirac's delta function. In this paper, synaptic weight $w_0^{i,j}$ is $-1 \le w_0^{i,j} \le 1$.

When $P_{\mathrm{m}}^{i}(t)$ exceeds threshold θ^i, the neuron i fires, and the neuron i outputs 1 (eq.(5)). In addition, a set of firing time T^i of the neuron i is expressed as follows:

$$y^i(t) = \mathcal{H}\left(P_{\mathrm{m}}^{i}(t) - \theta^i\right), \quad (5)$$
$$T^i = \{t_{\mu}^{i}; 1 \le \mu \le n^i\} = \{t \,|\, y^i(t) = 1\}, \quad (6)$$

where $\mathcal{H}(t)$ is Heaviside unit function.

3 Proposed method

In this section, definitions of training set and error function are given, and features and procedures of the proposed method are explained.

3.1 Training set and error function

Definitions of training set and error function are given before explaining the proposed method. Input-time-series data L_{I} and output-time-series data L_{O} of training set are defined as follows:

$$L_{\mathrm{I}} = \{l_{\zeta,p}^{i}; i \in I, 1 \le p \le P, 1 \le \zeta \le N_p^i\}, \quad (7)$$
$$L_{\mathrm{O}} = \{l_{\xi,p}^{o}; o \in O, 1 \le p \le P, 1 \le \xi \le N_p^o\}, \quad (8)$$

where I is a set of input layer's neurons, P is the number of learning patterns, and N_p^i is the number of pulses which input to a input layer's neuron i for a pattern p. And O is a set of output layer's neurons, N_p^o is the number of pulses which should output from a output layer's neuron o for the pattern p.

The error function which is used in this paper is defined as follows:

$$E_p = \sum_{o \in O} \int_0^{T_p} \left| \sum_{\xi=1}^{N_p^o} \mathcal{H}(t - l_{\xi,p}^{o}) - \sum_{\nu=1}^{n_p^o} \mathcal{H}(t - t_{\nu,p}^{o}) \right| dt, \quad (9)$$

$$E = \sum_{p=1}^{P} E_p, \quad (10)$$

where T_p is the length of pattern p, n_p^o is n^o on pattern p, $t_{\nu,p}^{o}$ is t_{ν}^{o} on pattern p, E_p is the errors on pattern p, and E is the total of E_p on all patterns.

3.2 Features and procedure

Main features of the proposed method are the following.

a. Pulse timing error between actual output of network and corresponding training signal is back-propagated from output layer to input layer. And pulse timing error is corrected by adjusting transmission delay between a presynaptic neuron and a postsynaptic neuron.

b. Pulse frequency error between actual output of network and corresponding training signal is back-propagated from output layer to input layer. And pulse frequency error is corrected by adjusting synaptic weight between a presynaptic neuron and a postsynaptic neuron.

c. The proposed method is able to apply to multi-layered feedforward PNN.

Procedure of the proposed method is described below.

1. Initialize parameters of PNN.
2. Input all pattern of input-time-series data L_{I} of training set to PNN. And record firing time T^j of all neurons on each pattern. T^j on each pattern p is expressed by $T_p^j = \{t_{\nu,p}^{j}; 1 \le \nu \le n_p^j\}$.
3. Calculate E_p on each pattern p.
4. Adjust transmission delay τ_{d} between each pair of neurons.
5. Adjust synaptic weight w_0 between each pair of neurons.
6. Repeat from 2. to 5. until satisfying given termination condition.

Step 4. and 5. are explained in section 3.3 and 3.4 in detail.

3.3 Transmission delay adjustment

Procedures of adjusting transmission delay which is mentioned at step 4 in section 3.2 are explained in detail. Neuron o, j and k denote a output layer's neuron, a hidden layer's neuron and a hidden or input layer's neuron, respectively. Besides, the neuron k is a presynaptic neuron of the neuron j, and the neuron j is a presynaptic neuron of the neuron o.

First of all, $\tau_{\mathrm{d}}^{o,j}$ which is transmission delay between the hidden layer's neuron j and the output layer's neuron o is updated as follows:

$$\tau_{\mathrm{d}}^{o,j}(c+1) = \tau_{\mathrm{d}}^{o,j}(c) + \alpha_{\tau_{\mathrm{d}}} \sum_{p=1}^{P} E_p \sum_{t_{\nu,p}^{j} \in T_p^j} \Delta\tau_{\mathrm{d},p}^{o,j}(t_{\nu,p}^{j}), \quad (11)$$

$$\Delta\tau_{\mathrm{d},p}^{o,j}(t_{\nu,p}^{j}) = \sum_{l_{\xi,p}^{o} \in F_p^o(t_{\mathrm{d},\nu,p}^{o,j})} W_{\tau_{\mathrm{d}}}^{j}(l_{\xi,p}^{o} - t_{\mathrm{d},\nu,p}^{o,j}), \quad (12)$$

$$F_p^o(t_{\mathrm{d},\nu,p}^{o,j}) = \{\, l_{\xi,p}^{o} \,|\, t_{\nu,p}^{j} < l_{\xi,p}^{o} \le t_{\nu+1,p}^{j}\}, \quad (13)$$

$$W_{\tau_{\mathrm{d}}}^{\nu}(\varphi) = e^{-\varphi^2} \sin\left(\frac{2\pi}{\tau_{\mathrm{r}}^{\nu}}\varphi\right), \quad (14)$$

where c is learning cycle, $\alpha_{\tau_{\mathrm{d}}}$ is learning rate regarding transmission delay, and $\Delta\tau_{\mathrm{d},p}^{o,j}(t_{\nu,p}^{j})$ is a function of $t_{\nu,p}^{j}$. In addition, $F_p^o(t_{\mathrm{d},\nu,p}^{o,j})$ is a set of elements which are included in L_{O} on pattern p between $t_{\nu,p}^{j}$ and $t_{\nu+1,p}^{j}$, and $W_{\tau_{\mathrm{d}}}^{j}(\cdot)$ is a window function regarding transmission delay.

$\gamma_p^j(t_{\nu,p}^j)$ which is the total of $\Delta\tau_{\mathrm{d},p}^{o,j}(t_{\nu,p}^j)$ of all postsynaptic neurons is calculated for each pulse of the neuron j as follows:

$$\gamma_p^j(t_{\nu,p}^j) \;=\; \sum_{o\in O}\Delta\tau_{\mathrm{d},p}^{o,j}(t_{\nu,p}^{o,j}). \qquad (15)$$

Secondly, $\tau_{\mathrm{d}}^{j,k}$ which is transmission delay between the hidden layer's neuron j and the other hidden layer's or input layer's neuron k is updated as following:

$$\tau_{\mathrm{d}}^{j,k}(c+1) \;=\; \tau_{\mathrm{d}}^{j,k}(c)+\alpha_{\tau_{\mathrm{d}}}\sum_{p=1}^{P}E_p\sum_{t_{\nu,p}^k\in T_p}\Delta\tau_{\mathrm{d},p}^{j,k}(t_{\nu,p}^k), \quad (16)$$

$$\Delta\tau_{\mathrm{d},p}^{j,k}(t_{\nu,p}^k) = \sum_{t_{\mu,p}^j\in F_p^j(t_{\mathrm{d},\nu,p}^{j,k})} W_{\tau_{\mathrm{d}}}^k\left(t_{\nu,p}^j+\beta_{\tau_{\mathrm{d}}}\gamma_p^j(t_{\mu,p}^j)-t_{\mathrm{d},\nu,p}^{j,k}\right), (17)$$

$$F_p^j(t_{\mathrm{d},\nu,p}^{j,k}) = \{\, t_{\mu,p}^j \,|\, t_{\nu,p}^k < t_{\mu,p}^j \le t_{\mathrm{d},\nu+1,p}^k\}, \qquad (18)$$

where $F_p^j(t_{\mathrm{d},\nu,p}^{j,k})$ is a set of elements which are included in T_p^j on pattern p between $t_{\nu,p}^k$ and $t_{\nu+1,p}^k$, and $\beta_{\tau_{\mathrm{d}}}$ is reduction rate regarding transmission delay.

$\beta_{\tau_{\mathrm{d}}}$ should be set appropriate value in accordance with time step of the finite difference Δt and transmission delay τ_d.

In order to update transmission delay of presynaptic neurons of the neuron k, $\gamma_p^k(t_{\nu,p}^k)$ which is the total of $\Delta\tau_{\mathrm{d},p}^{j,k}(t_{\nu,p}^k)$ of all postsynaptic neurons is calculated for each pulse of the neuron k as follows:

$$\gamma_p^k(t_{\nu,p}^k) = \sum_{j\in J}\Delta\tau_{\mathrm{d},p}^{j,k}(t_{\nu,p}^k). \qquad (19)$$

Eq.(16)$\sim$(19) is repeated until the presynaptic neuron k becomes the input layer's neuron.

3.4 Synaptic weight adjustment

Procedure of adjusting synaptic weight which is mentioned at step 5 in section 3.2 is explained in detail. Neuron o, j and k is the same as the neurons described in section 3.4.

First of all, $w_0^{o,j}$ which is synaptic weight between the hidden layer's neuron j and the output layer's neuron o is updated as follows:

$$w_0^{o,j}(c+1) \;=\; w_0^{o,j}(c)+\alpha_{w_0}\sum_{p=1}^{P}E_p\Delta w_{0,p}^{o,j}, \qquad (20)$$

$$\Delta w_{0,p}^{o,j} \;=\; \sum_{\nu=1}^{n^j}\sum_{g_{\mu,p}^o\in H_p^o(t_{\mathrm{d},\nu,p}^{o,j})} W_{w_0}^j(g_{\mu,p}^o-t_{\mathrm{d},\nu,p}^{o,j})\Psi_p^o(g_{\mu,p}^o), \quad (21)$$

$$W_{w_0}^{\mu}(\varphi) \;=\; \frac{e^{-2\varphi/\tau_{\mathrm{r}}^{\mu}}}{(1+e^{-2\varphi/\tau_{\mathrm{r}}^{\mu}})^2}, \qquad (22)$$

$$H_p^o(t_{\mathrm{d},\nu,p}^{o,j}) \;=\; \{g_{\mu,p}^o \,|\, t_{\nu,p}^j < g_{\mu,p}^o \le t_{\nu+1,p}^j, \;\; g_{\mu,p}^o \in G_p^o\}, \quad (23)$$

$$G_p^o \;=\; \{g_{\mu,p}^o; 1\le\mu\le N_{\mathrm{G},p}^o\} = \{t \,|\, \Psi_p^o(t)\neq 0\}, \qquad (24)$$

$$\Psi_p^o(t) \;=\; \sum_{\xi=1}^{n^{\xi}}\int_{t-\Delta t/2}^{t+\Delta t/2}\delta(u-l_{\xi,p}^o)du - \sum_{\nu=1}^{n^o}\int_{t-\Delta t/2}^{t+\Delta t/2}\delta(u-t_{\nu,p}^o)du, \qquad (25)$$

where c is learning cycle, α_{w_0} is learning rate regarding synaptic weight, and $\Delta w_{0,p}^{o,j}$ is update value of weight between the neuron j and o on pattern p. In addition, $W_{w_0}^j(\cdot)$ is a window function regarding synaptic weight, $H_p^o(t_{\mathrm{d},\nu}^{o,j})$ is a set of elements which are included in G_p^o on pattern p between $t_{\nu,p}^j$ and $t_{\nu+1,p}^j$, G_p^o is a set of time when $\Psi_p^o(t)$ is plus or minus value, and $\Psi_p^o(t)$ is a function which denotes the difference between the output of the neuron o and corresponding training signal.

$\Psi_p^j(t)$ which denotes the difference between the output of the neuron j and $\Psi_p^o(t)$ of the postsynaptic neuron o is calculated by eq.(26). G_p^j which denotes a set of time when $\Psi_p^j(t)$ is plus or minus value is defined by eq.(27).

$$\begin{aligned}\Psi_p^j(t) = \mathrm{Sign}\Bigg(&\mathrm{Sign}\Bigg(\sum_{o\in O}w_0^{o,j}\Big\{\beta_{w_0}\Psi_p^o(t+\tau_{\mathrm{d}}^{o,j}) \\ &+\sum_{t_{\nu,p}^o\in(T_p^o\cap\bar{G}_p^o)}\int_{t-\Delta t/2}^{t+\Delta t/2}\delta(u-t_{\nu,p}^o+\tau_{\mathrm{d}}^{o,j})du\Big\}\Bigg) \\ &-\sum_{\nu=1}^{n^j}\int_{t-\Delta t/2}^{t+\Delta t/2}\delta(u-t_{\nu,p}^j)du\Bigg),\end{aligned} \qquad (26)$$

$$G_p^j \;=\; \{g_{\mu,p}^j; 1\le\mu\le N_{\mathrm{G},p}^j\} = \{t \,|\, \Psi_p^j(t)\neq 0\}, \qquad (27)$$

$$\mathrm{Sign}(\varphi) = \begin{cases} -1 & \text{for} \quad \varphi<0, \\ \;\;0 & \text{for} \quad \varphi=0, \\ \;\;1 & \text{for} \quad \varphi>0, \end{cases} \qquad (28)$$

where β_{w_0} is reduction rate for synaptic weight, and $\bar{G}_p^o$ is a complementary set of G_p^o. In other words, $\bar{G}_p^o$ is a set of elements which are not included in G_p^o, but are included in a set of time $\{0, \Delta t, 2\Delta t, \cdots, T_p - \Delta t, T_p\}$, where the number of elements of this set is $T_p/\Delta t + 1$. Consequently, $(T_p^o \cap \bar{G}_p^o)$ denotes a set of required firing-time of the neuron o on pattern p.

Secondly, $w_0^{j,k}$ which is a synaptic weight between the hidden layer's neuron j and the other hidden layer's or input layer's neuron k is updated as following:

$$w_0^{j,k}(c+1) \;=\; w_0^{j,k}(c)+\alpha_{w_0}\sum_{p=1}^{P}E_p\Delta w_{0,p}^{j,k}, \qquad (29)$$

$$\Delta w_{0,p}^{j,k} = \sum_{\nu=1}^{n^k}\sum_{g_{\mu,p}^j\in H_p^j(t_{\mathrm{d},\nu,p}^{j,k})} W_{w_0}^k(g_{\mu,p}^j-t_{\mathrm{d},\nu,p}^{j,k})\Psi_p^j(g_{\mu,p}^j), \quad (30)$$

$$H_p^j(t_{\mathrm{d},\nu,p}^{j,k}) \;=\; \{g_{\mu,p}^j \;|\; t_{\nu,p}^k < g_{\mu,p}^j \le t_{\nu+1,p}^k, \;\; g_{\mu,p}^j \in G_p^j\}. \quad (31)$$

In order to update synaptic weight of presynaptic neurons of the neuron k, $\Psi_p^k(t)$ which denotes the difference between the output of neuron k and $\Psi_p^j(t)$ of the postsynaptic neuron j is calculated by eq.(32). G_p^k which denotes a set of time when $\Psi_p^k(t)$ is plus or minus value is defined by eq.(33).

$$\Psi_p^k(t) = \mathrm{Sign}\left(\mathrm{Sign}\left(\sum_{j \in J} w_0^{j,k} \left\{ \beta_{w_0} \Psi_p^j(t + \tau_\mathrm{d}^{j,k}) + \sum_{t_{\nu,p}^j \in (T_p^j \cap \bar{G}_p^j)} \int_{t-\Delta t/2}^{t+\Delta t/2} \delta(u - t_{\nu,p}^j + \tau_\mathrm{d}^{j,k}) \right\} \right) du - \sum_{\nu=1}^{n^k} \int_{t-\Delta t/2}^{t+\Delta t/2} \delta(u - t_{\nu,p}^k) du \right), \tag{32}$$

$$G_p^k = \{ g_{\mu,p}^k ; 1 \le \mu \le N_{\mathrm{G},p}^k \} = \{ t \,|\, \Psi_p^k(t) \ne 0 \}. \tag{33}$$

Eq.(29)-(33) is repeated until the presynaptic neuron k becomes the input layer's neuron.

4 Experiments

In this section, results of computational experiments are given in order to demonstrate the advantages of the proposed method.

The network structure of PNN used in the experiments is a multi-layered feedforward network which is one input layer, one output layer, and two hidden layers. The number of the input layer's neurons is two, that of the output layer's neurons is one, and that of the hidden layer's neurons per layer is 3∼15.

The training set used in the experiments is the following. Input-time-series data and output-time-series data are both periodic data.

The termination condition of learning used in the experiments is the following. The learning is stopped as success when error E becomes less than 5.0×10^{-3}, and as failure when the learning cycle c reaches 4,000 epochs without E being less than 5.0×10^{-3}.

In the experiments, convergence rate of learning is observed when the number of hidden layer's neurons is changed from 3 to 15. In addition, difference of convergence rate between synchronized I/O pulse timing case and non-synchronized I/O pulse timing case is compared. The learning is executed 100 times from different initial snaptic weights and transmission delay.

The convergence rate which is the average of 100 trials is showed in Figure 1. In Figure 1, x-axis indicates the number of hidden layer's neurons par layer and y-axis indicates the convergence rate. Figure 1 indicates that the convergence rate becomes large when the number of hidden layer's neurons increases. However, the rate saturates when the number of neurons becomes 15. In addition, the convergence rate of synchronized I/O pulse timing case is lower than that of non-synchronized one. This result indicates that the proposed method enables PNN to learn pulse timing with only small reduction of the convergence rate.

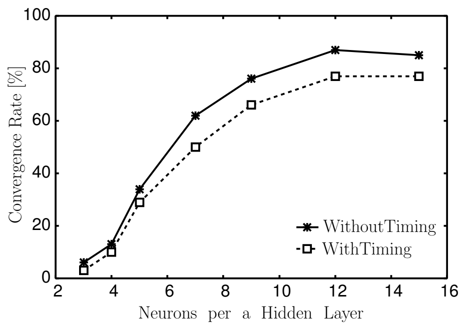

Figure 1: Convergence rate of learning.

5 Conclusion

The purpose of this paper was to propose a supervised learning rule adjusting I/O pulse timing for PNN. We verified the effectiveness of the proposed method by computer simulation. Results of computational experiments indicated that the proposed method enables learning of pulse timing in multi-layered feedforward PNN. However, the searching ability of solutions of the proposed method is not enough, since the convergence rate decreases when the number of hidden layer's neurons becomes small. To improve the searching ability of solutions is one of future works.

References

[1] Gerstner W, Kempter R, van Helmmen JL and Wagner H (1996), A neuronal learning rule for sub-millisecond temporal coding. *Nature*, vol.383, pp.76-78

[2] Eurich CW, Cowan JD and Milton JG (1997), Hebbian delay adaptation in a network of integrate-and-fire neurons. *Artificial neural networks* - ICANN'97, pp.157-162

[3] Kuroyanagi S, Iwata A (1998), An Supervised Learning Rule of Puled Neuron Model (in Japanese). *Technical report of IEICE*, NC97-151, pp.95-102

[4] Fatamata N, Kuroyanagi S and Iwata A (2002), An Implementation Method of Pulsed Neuron Model for FPGA Devices (in Japanese). *Technical report of IEICE*, NC2001-211, pp.121-128

[5] Maass W and Bishop CM (2001), Pulsed Neural Networks," *The MIT Press* (2001)