Learning Method of Recurrent Spiking Neural Networks to Realize Various Firing Patterns using Particle Swarm Optimization

Yasuaki Kuroe[1,2], Hitoshi Iima[2] and Yutaka Maeda[1]
[1]*Faculty of Engineering Science, Kansai University, Suita-shi, Osaka, Japan*
[2]*Faculty of Information and Human Sciences, Kyoto Institute of Technology, Kyoto, Japan*

Keywords: Spiking Neural Network, Learning Method, Particle Swarm Optimization, Burst Firing, Periodic Firing.

Abstract: Recently it has been reported that artificial spiking neural networks (SNNs) are computationally more powerful than the conventional neural networks. In biological neural networks of living organisms, various firing patterns of nerve cells have been observed, typical examples of which are burst firings and periodic firings. In this paper we propose a learning method which can realize various firing patterns for recurrent SNNs (RSSNs). We have already proposed learning methods of RSNNs in which the learning problem is formulated such that the number of spikes emitted by a neuron and their firing instants coincide with given desired ones. In this paper, in addition to that, we consider several desired properties of a target RSNN and proposes cost functions for realizing them. Since the proposed cost functions are not differentiable with respect to the learning parameters, we propose a learning method based on the particle swarm optimization.

1 INTRODUCTION

Recently there is a surge in the research of artificial spiking neural networks (SNNs) due to the fact that the functions of spiking neurons are closer to the physiological functions of the generic biological neurons than the conventional threshold and sigmoidal neurons (Mass and Bishop C., 1998; Gerstner and van Hemmen, 1993b; Mass, 1997b). In artificial spiking neural networks the information is encoded and processed by the spike trains (sequence of action potentials) similar to the biological neural networks (BNNs), through a discontinuous and nonlinear encoding mechanism (Mass and Bishop C., 1998; Gerstner and van Hemmen, 1993b). The conventional neuron models usually tend to ignore these sophisticated discontinuous encoding mechanisms. In addition to the SNNs' similarity to the BNNs, recently it has been reported that they are computationally more powerful than the conventional artificial neural networks (Mass, 1997b; Mass, 1997a; Mass, 1996). It is however much more difficult to analyze and synthesize the SNNs than the conventional threshold and sigmoidal neural networks. This is due to their associated nonlinear and discontinuous encoding mechanisms, which make the SNNs continuous and discrete hybrid-dynamical systems. In this paper we discuss a learning method, which is one of fundamental problems of NNs, for the recurrent spiking neural networks (RSNNs).

In the case of the sigmoidal neural networks, the backpropagation method was proposed by Rumelhart et al. (Rumelhart and McClelland, 1986) for feedforward types of neural networks, which is one of the pioneering works that trigger the research interests of applications of the neural networks. Following the backpropagation method, learning methods have been developed for recurrent sigmoidal neural networks (Kuroe, 1992).

In the case of the SNNs, their learning methods have not been actively studied due to their associated nonlinear and discontinuous encoding mechanisms and only a few studies have been done. As unsupervised learning W. Gestner et al. proposed a learning method for feedforward SNNs, which is based on the concept of the Hebbian learning (Gerstner and van Hemmen, 1993a). As supervised learning for SNNs the following studies have been done. K. Selvaratnam et al. proposed a gradient based learning method for RSNNs (Selvaratnam and Mori, 2000) based on sensitivity equation approach and Y. Kuroe et al. proposed based on adjoint equation approach (Kuroe and Ueyama, 2010). Backpropagation-like learning method is proposed for feedforward SNNs (Bohte et al., 2002) and for deep SNNs (Lee and Pfeiffer, 2016).

Kuroe, Y., Iima, H. and Maeda, Y.
Learning Method of Recurrent Spiking Neural Networks to Realize Various Firing Patterns using Particle Swarm Optimization.
DOI: 10.5220/0008164704790486
In *Proceedings of the 11th International Joint Conference on Computational Intelligence (IJCCI 2019)*, pages 479-486
ISBN: 978-989-758-384-1

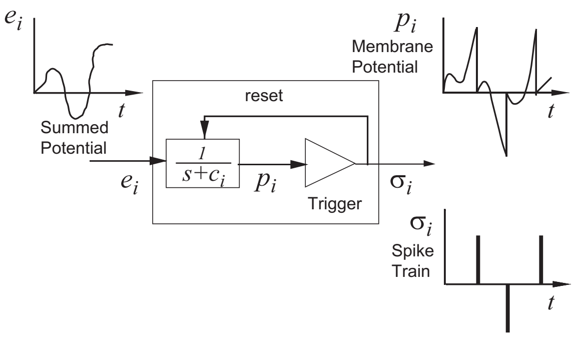

Figure 1: Schematic of the firing mechanism of the integrate and fire type SN.

In biological neural networks of living organisms, various firing patterns of nerve cells have been observed (Izhikevich, 2007), typical examples of which are burst firings and periodic firings. It is expected that artificial SNNs with recurrent architectures realize various complicated firing patterns (Izhikevich, 2004; Izhikevich, 2007) because of their behavior as nonlinear dynamical systems. In this paper we propose a learning method which can realize various firing patterns for RSNNs. As stated above learning methods of RSNNs were proposed in (Selvaratnam and Mori, 2000; Kuroe and Ueyama, 2010), in which the learning problem is formulated such that the number of spikes emitted by a neuron and their firing instants coincide with given desired ones. In addition to that, in this paper we consider several desired properties of a target RSNN and propose cost functions for realizing them. Since the proposed cost functions are not differentiable with respect to the learning parameters and their gradients do not exist, the gradient based methods cannot be used. We propose a learning method based on the particle swarm optimization (PSO)(Kennedy and Eberhart, 2001).

2 SPIKING NEURAL NETWORKS

2.1 Firing Mechanism of SNs

In this paper we consider the integrate-and-fire type spiking neurons (SNs) shown in Fig. 1. The firing mechanism of the i-th integrate-and-fire type spiking neuron is as follows. When an input stimulus $e_i(t)$ is fed into the integrator with transfer function $1/(s+c_i)$, a spike is emitted at the moment when the internal state $p_i(t)$ which corresponds to the membrane potential of a biological neuron reaches the threshold value s_i. At the instant of spike emission the sign of the internal state $p_i(t)$ is observed and assigned to the output spike and the internal state $p_i(t)$is reset to zero. The firing mechanism is mathematically described as follows.

$$\sigma_i(t) = \sum_{k_i=1}^{K_i} \varepsilon_{i,k_i} \times \delta(t - t_{i,k_i}) \tag{1}$$

$$t_{i,k_i} = \min[t : t > t_{i,k_i-1}, |p_i(t)| \geq s_i] \tag{2}$$

$$\varepsilon_{i,k_i} = \mathrm{sgn}[p_i(t_{i,k_i}^-)] \tag{3}$$

$$\frac{dp_i(t)}{dt} = -c_i p_i(t) + e_i(t), \ \ t_{i,k_i-1} < t < t_{i,k_i} \tag{4}$$

$$p_i(0) = p_i^0, \tag{5}$$

$$p_i(t_{i,k_i}^+) = 0, \ \ k_i = 1, \dots K_i, \tag{6}$$

where, $\sigma_i(t)$: output sequence of spikes of the i-th spiking neuron, K_i: total number of spikes fired before time t, t_{i,k_i}: time at which the k_i^{th} spike is emitted, $p_i(t)$: internal state, p_i^0: initial condition of $p_i(t)$, s_i: threshold value, $e_i(t)$: input to the spiking neuron, and $p_i(t_{i,k_i}^-) = \lim_{\varepsilon\to 0} p_i(t_{i,k_i} - \varepsilon)$, $p_i(t_{i,k_i}^+) = \lim_{\varepsilon\to 0} p_i(t_{i,k_i} + \varepsilon)$, $\varepsilon > 0$. Equation (6) denotes the resetting mechanism of the SN at spike emission times.

2.2 Model of Spiking Neural Networks

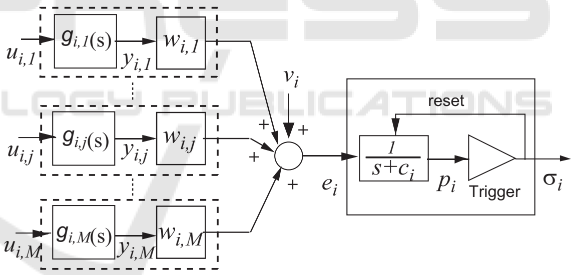

Figure 2: Model of the connections of i-th spiking neuron in the SNN.

In this paper we consider recurrent spiking neural networks (RSNNs) in which integrate-and-fire type spiking neurons shown in Fig.1 are fully connected through synaptic weights $w_{i,j}$ and time delay elements $g_{i,j}(s)$. Figure 2 shows a schematic diagram of the connection of the ith spiking neuron in the RSNN. We let here the number of neurons composed in the RSNN be M. The elements $g_{i,j}(s)$ determine shape of post synaptic potentials, so called spike-response function as shown in Fig. 3, or delay due to the spike transmission between spiking neurons. Synaptic weights $w_{i,j}$ are multiplied by the time delay elements ($w_{i,j} \times g_{i,j}(s)$). The input stimulus $e_i(t)$ of the i-th SN is generated by the weighted sum of each output $y_{i,j}(t)$ of the element $g_{i,j}(s)$ and the external input $v_i(t)$. The input $u_{i,j}(t)$ of $g_{i,j}(s)$ is connected with

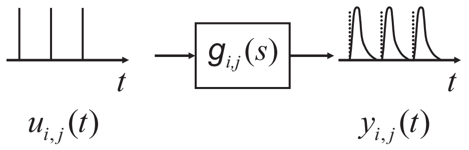

Figure 3: Spikes and spike-response function.

the output of the j-th neuron $\sigma_j(t)$. The connection topology of the entire RSNN is given by

$$e_i(t) = \sum_{j=1}^{M} w_{i,j} y_{i,j}(t) + v_i(t), \tag{7}$$

$$u_{i,j}(t) = \sigma_j(t). \tag{8}$$

For the convenience of the derivation of the learning algorithms, the elements $g_{i,j}(s)$ are expressed in the state space form:

$$\frac{dx_{i,j}(t)}{dt} = A_{i,j}x_{i,j}(t) + b_{i,j}u_{i,j}(t) \tag{9}$$

$$y_{i,j}(t) = c_{i,j}x_{i,j}(t) \tag{10}$$

$$x_{i,j}(0) = x^0_{i,j}, \quad i,j = 1,\cdots,M \tag{11}$$

$$g_{i,j}(s) = c_{i,j}(sI - A_{i,j})^{-1}b_{i,j},$$

where $x_{i,j}(t)$: N dimensional state vector, $x^0_{i,j}$: N dimensional initial state vector and the dimensions of the system matrices and vectors $A_{i,j}$, $b_{i,j}$ and $c_{i,j}$ are $N \times N$, $N \times 1$ and $1 \times N$, respectively. Equations (1)$\sim$(11) give the whole description of the RSNN considered in this paper.

3 PROPOSED METHOD

3.1 Examples of Firing Patterns

In biological neural networks of living organisms, various firing patterns of nerve cells are observed. The purpose of this paper is to propose a learning method of neural network that produces such firing patterns. Here we introduce two examples of such firing patterns observed in biological neural networks of living organisms like human brains. One typical example is a burst firing pattern. It is a firing pattern as shown in Fig. 4, in which the number of firings suddenly increases and decreases and the firing time interval and the non-firing time interval are clearly divided. In the figure an example of the time evolution of the membrane potential of a neuron which generates burst firing is shown. Another typical example is a periodic firing pattern. It is a firing pattern in which a same firing pattern is repeated with a constant period. In both firing patterns, burst and periodic firing patterns, there could exist various firing patterns depending on firing time, firing frequency and so on.

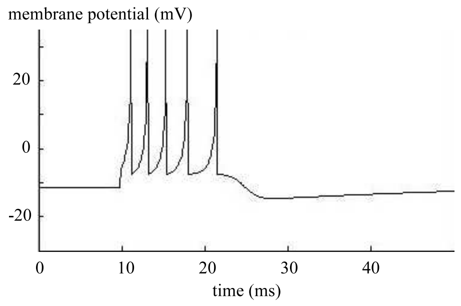

Figure 4: An example of burst firing.

3.2 Formulation of Learning Problems

3.2.1 Learning Problem for Generating Various Firing Patterns

We now formulate the learning problem which realizes various firing patterns. As stated above, in burst firing patterns the firing time interval and the non-firing time interval are clearly divided. In order to realize this, it is necessary to divide total time interval into some sub time intervals and specify a desired firing pattern to each sub time interval.

Suppose that the RSNN operates starting at time $t = 0$ and finishing at time $t = t_f$ and the total time interval is $0 \le t < t_f$. For each neuron, say the i-th SN denoted by SN$_i$ in the RSNN, we divide the total time interval into sub time intervals and the number of sub time intervals is denoted by S_i. For the i-th SN, let tv_i-th sub time interval be $t^s_{i,tv_i} \le t < t^e_{i,tv_i}, (tv_i = 1,2,\cdots,S_i)$, where the start time t^s_{i,tv_i} and the finish time t^e_{i,tv_i} satisfy the following equations.

$$t^e_{i,tv_i} = t^s_{i,tv_i+1}(i = 1,2,\cdots,M; tv_i = 1,\cdots,S_i - 1)$$
$$t^s_{i,1} = 0, \ \ t^e_{i,S_i} = t_f(i = 1,2,\cdots,M) \tag{12}$$

We now formulate the learning problem which realizes various firing patterns. We already proposed a learning method of the RSNNs described by (1)$\sim$(11), in which the learning problem is formulated such that the number of spikes emitted by SN and their firing instants coincide with given desired numbers of spikes and their desired firing instants for the interval $[0,t_f]$ (Selvaratnam and Mori, 2000; Kuroe and Ueyama, 2010). In this paper, in order to realize various firing patterns, in addition to the learning problem in (Selvaratnam and Mori, 2000; Kuroe and Ueyama, 2010) we consider the learning problems in which desired number of spikes without specifying their firing instant is given, and desired values of upper and/or lower limit of the number of spikes emitted by SN are given. The learning problem is formulated as follows.

[Learning Problem]
Determine the values of the synaptic weights $w_{i,j}$ such that the RSNN generates the desired spike sequence if one of the values of the following items is specified for any given neuron SN$_i$ in the RSNN and any given sub time interval $t^s_{i,tv_i} \leq t < t^e_{i,tv_i}$:

a) the number of spikes and their firing instants,

b) the number of spikes,

c) the value of upper limit of the number of spikes,

d) the value of lower limit of the number of spikes or

e) the values of lower and upper limits of the number of spikes.

Let the set of the neurons SN$_i$ which one of the items a)–e) is specified be O and the set of the indexes tv_i of the sub time intervals which one of the items a)–e) is specified be U_i. For the sub time interval $t^s_{i,tv_i} \leq t < t^e_{i,tv_i} (tv_i \in U_i)$, let the number of spikes which SN$_i$ emits be K_{i,tv_i}, and the time instant of k_{i,tv_i}-th spike be $t_{i,k_{i,tv_i}}$ ($k_{i,tv_i} = 1, 2, \cdots, K_{i,tv_i}$). We introduce the measures which represent discrepancies between K_{i,tv_i}, $t_{i,k_{i,tv_i}}$ and their specified values in a)–e) as cost functions, denoted by $J_{i,tv_i}(t_{i,k_{i,tv_i}}, K_{i,tv_i})$, and define the total cost function as follows.

$$J_1 = \sum_{i \in O} \sum_{tv_i \in U_i} \alpha_{i,tv_i} J_{i,tv_i}(t_{i,k_{i,tv_i}}, K_{i,tv_i}) \qquad (13)$$

where α_{i,tv_i} are weight coefficients. The definition of J_{i,tv_i} will be given in the next section. Letting the learning parameters be $X_1 = (w_{1,1}, w_{1,2}, \cdots, w_{i,j}, \cdots, w_{M,M})$, the learning problem here is reduced to the following optimization problem:

$$\begin{aligned} &\text{Minimize } J_1 \\ &\text{w.r.t. } X_1 \end{aligned} \qquad (14)$$

3.2.2 Learning Problem for Generating Periodic Firing Patterns

In order to make the network generate persistent periodic phenomena it should be a nonlinear dynamical system like the RSNNs described by (1)∼(11). Here we formulate the learning problem which generates various patterns in the previous subsection and makes them periodic with a given desired period T. In order to make them periodic, the state variables p_i and $x_{i,j}(t)$ of the RSNN must satisfy the conditions:

$$p_i(t) = p_i(t+T), \qquad x_{i,j}(t) = x_{i,j}(t+T). \quad (15)$$

The problem here is to determine the values of the network parameters which minimize the cost function J_1 and make the conditions (15) being satisfied simultaneously. We introduce another cost function to realize the conditions. Note that the initial states of the RSNN which realize the conditions (15) are unknown. As the learning parameters we choose not only the synaptic weights $w_{i,j}$ but also the initial states of the RSNN, p^0_i and $x^{0,n}_{i,j}$ ($i = 1, 2, \cdots, M$; $n = 1, 2, \cdots, N$). Defining $X_2 = (w_{1,1}, \cdots, w_{i,j}, \cdots, w_{M,M}, p^0_1, \cdots, p^0_i, \cdots, p^0_M, x^{0,1}_{1,1}, \cdots, x^{0,n}_{i,j}, \cdots, x^{0,N}_{M,M})$, the learning problem here is reduced to the following optimization problem:

$$\begin{aligned} &\text{Minimize } J_1 + \beta J_2 \\ &\text{w.r.t. } X_2 \\ &\text{subject to } |p^0_i| < s_i \end{aligned} \qquad (16)$$

J_2 is the cost function realizing the periodicity condition (15), the definition of which is given in the next section, and β is a weight coefficient. Note that the constraint $|p^0_i| < s_i$ is set to prevent the fact that SN$_i$ always fires at the initial time $t = 0$ if $|p^0_i| \geq s_i$.

3.3 Defining the Cost Functions

It is known that, since PSO is an optimization method that uses only values of a cost function and does not require its continuity or the existence of its gradient, various cost functions can be set according to optimization problems. It is, therefore, important what kind of cost functions are set. We now propose cost functions which realize the items a)–e) and the periodicity condition (15).

3.3.1 Cost Function for Firing Instants

Let the desired number of spikes emitted by SN$_i$ for the time interval $t^s_{i,tv_i} \leq t < t^e_{i,tv_i}$ be K^d_{i,tv_i} and their desired firing instants be $t^d_{i,k_{i,tv_i}}$ (k_{i,tv_i}=1,2,$\cdots$,K^d_{i,tv_i}). In order to realize that the number of spikes emitted by each SN$_i$ and their firing instants coincide with the given desired ones, we define the following cost function J_{11}.

$$J_{11} = \sum_{k_{i,tv_i}=1}^{K_{i,tv_i}} |t^d_{i,k_{i,tv_i}} - t_{i,k_{i,tv_i}}|^{ex} + J_{111} \qquad (17)$$

where

$$J_{111} = \begin{cases} \displaystyle\sum_{k_{i,tv_i}=K^d_{i,tv_i}+1}^{K_{i,tv_i}} |t^p_{i,k_{i,tv_i}} - t_{i,k_{i,tv_i}}|^{ex}, (K_{i,tv_i} > K^d_{i,tv_i}) \\ \displaystyle\sum_{k_{i,tv_i}=K_{i,tv_i}+1}^{K^d_{i,tv_i}} |t^d_{i,k_{i,tv_i}} - t_{i,K_{i,tv_i}}|^{ex}, (K_{i,tv_i} < K^d_{i,tv_i}) \\ 0, \qquad (K_{i,tv_i} = K^d_{i,tv_i}) \end{cases}$$

and *ex* is the exponentiation parameter which is usually chosen as $ex = 1$ or 2. J_{111} is a penalty function which is applied when the number of spikes emitted by SN$_i$ does not coincident with the desired one. When $K_{i,tv_i} > K^d_{i,tv_i}$, in order to avoid the excess number of spikes being fired we set provisional teaching signals for the excess spikes. Let $t^p_{i,k_{i,tv_i}}$ be the timings of the provisional teaching signals to the firing instants of the excess spikes $t_{i,k_{i,tv_i}}, k_{i,tv_i} = K^d_{i,tv_i}+1,\cdots,K_{i,tv_i}$. We choose values of $t^p_{i,k_{i,tv_i}}$ to be larger enough than t_f ($t^p_{i,k_{i,tv_i}} >> t_f$). When $K_{i,tv_i} < K^d_{i,tv_i}$, in order to make each SN$_i$ generate additional spikes we set the penalty as follows. Considering that at the firing instant $t_{i,K_{i,tv_i}}$ of the last actual spike emitted by SN$_i$ the missing $K^d_{i,tv_i} - K_{i,tv_i}$ number of spikes fire simultaneously, we set the error between the last actual firing instant and the desired firing instants as in the second case of J_{111}.

3.3.2 Cost Function for the Upper and Lower Limits of the Number of Spikes

Let the desired upper and lower limit values of the number of spikes emitted by SN$_i$ for the time interval $t^s_{i,tv_i} \le t < t^e_{i,tv_i}$ be K^{du}_{i,tv_i} and K^{dl}_{i,tv_i}, respectively. The cost function for the upper limit is set so that its value increases as the number of spikes emitted by SN$_i$ exceeds the upper limit value K^{du}_{i,tv_i} as follows.

$$J_{12} = \max(K_{i,tv_i} - K^{du}_{i,tv_i}, 0)^{ex} \tag{18}$$

The cost function for the lower limit is set so that its value increases as the number of spikes emitted by SN$_i$ lowers the lower limit value K^{dl}_{i,tv_i} as follows.

$$J_{13} = \max(K^{dl}_{i,tv_i} - K_{i,tv_i}, 0)^{ex} \tag{19}$$

3.3.3 Cost Function for Generating Periodic Firing Patterns

In order to make firing pattern be periodic with period T, the state variables $p_i(t)$ and $x_{i,j}(t)$ of the RSNN must satisfy the conditions (15). The cost function J_2 for the problem (16) is set so as to realize the conditions (15) as follows.

$$\begin{aligned} J_2 &= \sum_{n_T=1}^{N_T}\sum_{i=1}^{M}\Bigg(|p_i^0 - p_i(n_T T)|^{ex} \\ &+ \sum_{j=1}^{M}\sum_{n=1}^{N}|x^{0,n}_{i,j} - x^n_{i,j}(n_T T)|^{ex}\Bigg) \end{aligned} \tag{20}$$

It is theoretically enough to let $N_T = 1$, however, we introduce the parameter N_T for numerical accuracy.

The items a)–e) in Subsection 3.2.1 can be realized by appropriately choosing from the cost functions J_{11}, J_{12}, J_{13} defined in the above section and combining them as follows.

a) J_{11} b) $J_{12}+J_{13}$ (Let $K^{du} = K^{dl}$.)
c) J_{12} d) J_{13} e) $J_{12}+J_{13}$

3.4 Learning Method based on the PSO

The cost function J_{11} defined by (17) is differentiable with respect to the network parameters $X_1 = (w_{1,1}, w_{1,2}, \cdots, w_{i,j}, \cdots, w_{M,M})$ if $ex = 2$, and gradient based learning methods were proposed in (Selvaratnam and Mori, 2000; Kuroe and Ueyama, 2010) for RSNNs. Since the other cost functions are not differentiable, we use the particle swarm optimization method (PSO) (Kennedy and Eberhart, 2001) in order to solve the optimization problems (14) and (16). In the following we explain the outline of the PSO.

Consider an optimization problem of determining values of N_x decision variables $X = (x_1, \cdots, x_n, \cdots, x_{N_x})$ which minimize a cost function $J(X)$. In the PSO, a swarm is prepared and its all member particles are used for solving the problem. Let P be the number of the particles (the swarm size), and $X^{p,k} = (x_1^{p,k}, \cdots, x_n^{p,k}, \cdots, x_{N_x}^{p,k})$ be the candidate solution of p-th particle ($p = 1, 2, \cdots, P$) at the k-th iteration. The candidate solution of p-th particle at the next ($(k+1)$-th) iteration is determined based on the update equation of the PSO as follows.

$$x_n^{p,k+1} = x_n^{p,k} + \triangle x_n^{p,k+1} \tag{21}$$

where

$$\begin{aligned} \triangle x_n^{p,k+1} &= W\triangle x_n^{p,k} \\ &+ C_1 rand_1()_n^{p,k}(pbest_n^{p,k} - x_n^{p,k}) \\ &+ C_2 rand_2()_n^{p,k}(gbest_n^{k} - x_n^{p,k}) \end{aligned} \tag{22}$$

and the update value $\triangle X^p = (\triangle x_1^p, \cdots, \triangle x_n^p, \cdots, \triangle x_{N_x}^p)$ is called the velocity vector of the p-th particle, $rand_1()_n^{p,k}, rand_2()_n^{p,k}$ are uniform random numbers in the range from 0 to 1, W is a weight parameter called the inertia weight, and C_1 and C_2 are weight parameters called the acceleration coefficients. $pbest^{p,k} = (pbest_1^{p,k}, \cdots, pbest_n^{p,k}, \cdots, pbest_{N_x}^{p,k})$ is called the personal best which is the best candidate solution found by the p-th particle until the k-th iteration and $gbest^k = (gbest_1^k, \cdots, gbest_n^k, \cdots, gbest_{N_x}^k)$ is called the global best which is the best candidate solution found by all the members of the swarm, and they are given as: $pbest_n^{p,k} = x_n^{p,k^*}$ and $gbest_n^k = pbest_n^{p^*,k}$ where $k^* = \arg\min_{1\le k'\le k} J(X^{p,k'})$ and $p^* = \arg\min_{1\le p\le P} J(pbest^{p,k})$.

4 NUMERICAL EXPERIMENTS

4.1 Problems and Experiment

We prepare the following three learning problems for experiments and apply the proposed method to them.

Experiment 1

We consider the RSNN consists of fully connected five neurons $\text{SN}_i (i = 1, 2, \dots, 5)$ and one input neuron SN_{input} which generates a triggering input $v_i(t)$ as shown in Fig. 5. We let the triggering input be $v_i(t) = \delta(0)$. In this problem the total time interval $[0.0, 5.0)$ is divided into two sub time intervals $[0.0, 2.5)$ and $[2.5, 5.0)$. The desired firing sequences shown in Table 1 are given to the RSNN, where it is trained so as that SN_1 fires the desired numbers of spikes for each sub time interval and without assigning their firing instants, and SN_2, SN_3 and SN_4 fire the desired lower and/or upper limit values of the number of spikes. SN_5 is not given desired firing sequences. This experiment is a learning problem in which the items b), c), d) and e) in 3.2.1 is specified.

Experiment 2

Experiment 2 is a learning problem in which the item b) in 3.2.1 is specified and burst firings with the desired number of spikes are realized. We consider the same RSNN as in Experiments 1 shown in Fig. 5. The time interval $[0.0, 5.0)$ is divided into three sub time intervals $[0.0, 1.0), [1.0, 1.5)$ and $[1.5, 5.0)$. The desired firing sequences shown in Table 2 are given to SN_1 of the RSNN so that it emits a specified desired number of spikes for the time interval $[1.0, 1.5)$ and it does not fire for the other time intervals $[0.0, 1.0)$ and $[1.5, 5.0)$. The experiments are carried out for two conditions I ($K^d_{1,2} = 10$) and II ($K^d_{1,2} = 20$).

Experiment 3

Experiment 4 is a learning problem in which the item b) in 3.2.1 is specified and a periodic firing is realized. We consider the RSNN consists of two neurons $\text{SN}_i (i = 1, 2)$ which are mutually connected as shown in Fig. 6 because neural oscillators are usually realized by using such fully connected two neuron networks. The total time interval is $[0.0, 20.0)$ and it is divided into five sub time intervals $[0.0, 4.0), [4.0, 8.0), [8.0, 12.0), [12.0, 16.0)$ and $[16.0, 20.0)$. The desired firing sequences shown in Table 3 are given to SN_1 of the RSNN so that it fires a specified desired number of spikes for each sub time interval. In addition to that the periodic conditions (15) are given so that the RSNN generates a periodic firing pattern with the period $T = 20$.

In order to solve the problems of the above experiments it is necessary to choose appropriate cost functions in the optimization problems (14) and (16). We choose the following cost functions for each problem:

$$\text{Ex. 1:}\ J = \sum_{i \in O} \sum_{tv_i \in U_i} (J_{12} + J_{13})$$
$$\text{Ex. 2-I, II:}\ J = \sum_{i \in O} \sum_{tv_i \in U_i} (J_{12} + J_{13})$$
$$\text{Ex. 3:}\ J = \sum_{i \in O} \sum_{tv_i \in U_i} (J_{12} + J_{13}) + J_2$$

In the experiments the parameters of the RSNN are set as follows. For all experiments we choose $c_i = 0.2$, $s_i = 1.0$ and the parameters in the state space model of the elements $g_{i,j}(s)$ are: $N = 2$ and $A_{i,j} = \begin{pmatrix} -3.0 & 0.0 \\ 0.0 & -6.0 \end{pmatrix}$, $b_{i,j} = \begin{pmatrix} 1.0 \\ 1.0 \end{pmatrix}$, $c_{i,j} = (1.0, -1.0)$. In Experiment 1 and 2 the initial states are chosen as $p^0_i = 0.0$, $x^0_{i,j} = \begin{pmatrix} 0.0 \\ 0.0 \end{pmatrix}$.

For all the experiments the exponentiation parameter ex of the cost functions is chosen as $ex = 2$. For Experiment 3 the parameter N_T of the cost function J_2 is chosen as $N_T = 1$ and the weight coefficient γ is chosen as $\gamma = 25$.

For all experiments the parameters of the particle swarm optimization are chosen as follows. The parameters of the update equations (21) and (22) (W, C_1, C_2) are chosen:$(W, C_1, C_2) = (0.7, 1.4, 1.4)$, the number of the particles is $P = 100$ or $P = 200$, the maximum number T_{max} of learning iterations is $T_{max} = 5000$. In each experiment the learning iteration finishes when the number of the learning iteration reaches T_{max}. For each particle the initial values of the candidate solution and its velocity, $w^{p,0}_{i,j}$ and $\triangle w^{p,0}_{i,j}$, are determined randomly within the range $[-10, 10]$. In Experiment 3 for each particle the initial values of the candidate solution and its velocity, p^0_i, $\triangle p^0_i$, $x^{0,n}_{i,j}$ and $\triangle x^{0,n}_{i,j}$ are determined randomly within the range $[-1, 1]$ by considering the fact that $s_i = 1.0$.

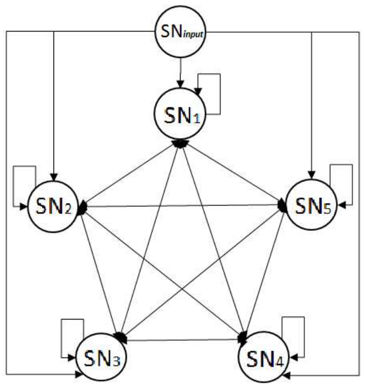

Figure 5: Fully connected five neurons $\text{SN}_i (i = 1, \dots, 5)$ and one input neuron SN_{input}.

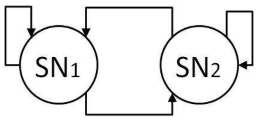

Figure 6: RSNN consists of mutually connected two neurons SN$_i (i = 1,2)$.

Table 1: Desired Firing Sequences (Ex. 1).

Ex. 1 $(S_1, S_2 = 2, t_f = 5.0)$

$[t^s_{i,tv_i}, t^e_{i,tv_i})$	$[0.0, 2.5)$	$[2.5, 5.0)$
SN$_1$	$K^d_{1,1} = 8$	$K^d_{1,2} = 3$
SN$_2$	$K^{dl}_{2,1} = 8$	$K^{dl}_{2,2} = 3$
SN$_3$	$K^{du}_{3,1} = 10$	$K^{du}_{3,2} = 10$
SN$_4$	$K^{dl}_{4,1} = 10, K^{du}_{4,1} = 5$	$K^{dl}_{4,2} = 20, K^{du}_{4,2} = 15$

Table 2: Desired Firing Sequences (Ex. 2).

Ex. 2 $(S_1 = 3, S_2, \cdots, S_5 = 1, t_f = 5.0)$

	$[t^s_{i,tv_i}, t^e_{i,tv_i})$	$[0.0, 1.0)$	$[1.0, 1.5)$	$[1.5, 5.0)$
I	SN$_1$	$K^d_{1,1} = 0$	$K^d_{1,2} = 10$	$K^d_{1,3} = 0$
II	SN$_1$	$K^d_{1,1} = 0$	$K^d_{1,2} = 20$	$K^d_{1,3} = 0$

Table 3: Desired Periodic Firing Sequences (Ex. 3).

Ex. 3 $(S_1 = 5, t_f = 20.0)$

$[t^s_{i,tv_i}, t^e_{i,tv_i})$	$[0.0, 4.0)$	$[4.0, 8.0)$	$[8.0, 12.0)$
SN$_1$	$K^d_{1,1} = 4$	$K^d_{1,2} = 4$	$K^d_{1,3} = 4$
$[t^s_{i,tv_i}, t^e_{i,tv_i})$	$[12.0, 16.0)$	$[16.0, 20.0)$	
SN$_1$	$K^d_{1,4} = 4$	$K^d_{1,5} = 4$	

4.2 Results and Discussions

We applied the proposed learning method to each experiment. The results are shown here.

Result of Experiment 1

After training the RSNN shown in Fig. 5 by the proposed method we run it by the same triggering signal $v_i(t) = \delta(0)$ from the input neuron SN$_{input}$ and observe its behavior. Figures 7, 8, 9 and 10 show examples of time evolutions of the internal state $p_1(t)$ of SN$_1$, $p_2(t)$ of SN$_2$, $p_3(t)$ of SN$_3$ and $p_4(t)$ of SN$_4$ obtained from the result, respectively. It is observed from the figures that SN$_1$, SN$_2$, SN$_3$ and SN$_4$ emit the desired firing sequences shown in Table 1, which implies that the learning is successfully done by the proposed method.

Result of Experiment 2

Figures 11 and 12 show examples of time evolutions of the internal state $p_1(t)$ of SN$_1$ obtained from the results Ex. 3-I and Ex. 3-II, respectively. It is observed from the figures the burst firings with the desired number of spikes are realized and the learning is successfully done by the proposed method.

Result of Experiment 3

After training the RSNN shown in Fig. 6 by the proposed method, we run it with the obtained initial condition and observe its behavior. Note that the RSNN shown in Fig. 6 has no input neuron and it is triggered by the initial conditions. Figures 13 shows an example of the time evolution of the internal state $p_1(t)$ of SN$_1$ obtained from the result. It is observed that the periodic firings with the desired number of spikes are realized.

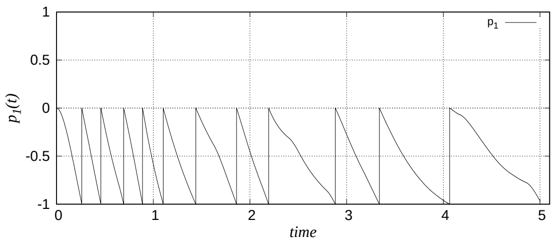

Figure 7: Time evolution of $p_1(t)$ after learning in Ex. 1.

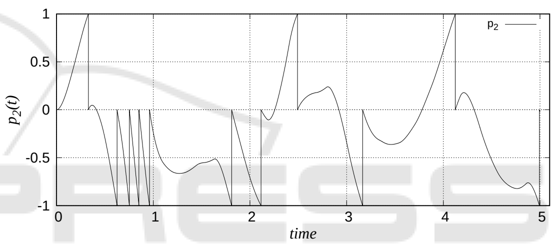

Figure 8: Time evolution of $p_2(t)$ after learning in Ex. 1.

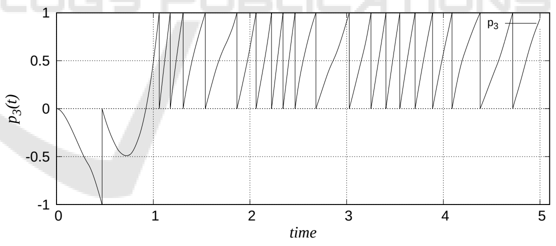

Figure 9: Time evolution of $p_3(t)$ after learning in Ex. 1.

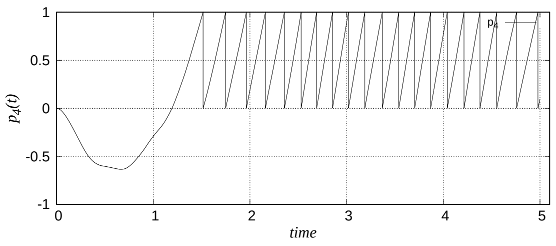

Figure 10: Time evolution of $p_4(t)$ after learning in Ex. 1.

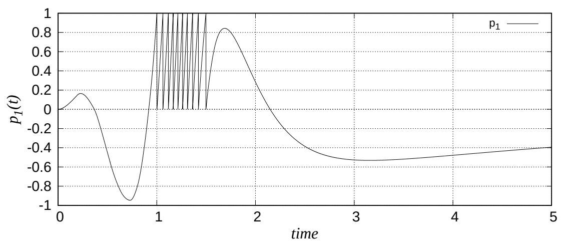

Figure 11: Time evolution of $p_1(t)$ after learning in Ex. 2-I.

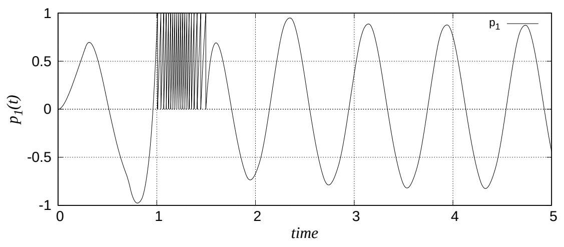

Figure 12: Time evolution of $p_1(t)$ after learning in Ex. 2-II.

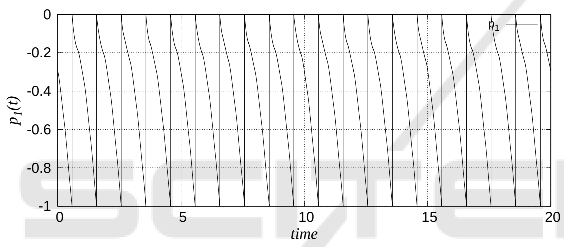

Figure 13: Time evolution of $p_1(t)$ after learning in Ex. 3.

5 CONCLUSIONS

In biological neural networks of living organisms, various firing patterns of nerve cells have been observed, typical example of which are burst firings and periodic firings. The RSNNs are expected to realize various complicated firing patterns because of their behavior as nonlinear dynamical systems. In this paper we proposed a learning method which can realize various firing patterns for RSNNs. We considered several desired properties of a target RSNN and proposed cost functions for realizing them. Since the proposed cost functions are not differentiable with respect to the learning parameters and their gradients do not exist, we propose larning methods based on the particle swarm optimization (PSO). Some experimental examples were provided to show the validity of the proposed method.

This work was financially supported in part by the Kansai University Fund for the Promotion and Enhancement of Education and Research, 2018 and JSPS KAKENHI (18K11483).

REFERENCES

Bohte, S. M., K., N., J., and Poutre, H. L. (2002). Error-backpropagation in temporally encoded networks of spiking neurons. *Neurocomputing*, Vol.48(Issues 1-4):pp.17–37.

Gerstner, W., R. R. and van Hemmen, J. L. (1993a). Why spikes? hebbian learning and retrieval of time-resolved excitation patterns. *Biological. Cybernetics*, Vol.69:pp.503–515.

Gerstner, W. and van Hemmen, J. L. (1993b). How to describe neural activities: Spikes, rates or assemblies? *Advances in Neural Information Processing Systems, San Mateo*, Vol.6:pp.363–374.

Izhikevich, E. (2007). *Dynamical systems in Neuroscience -The Geometry of Excitability and Bursting*. MIT Press.

Izhikevich, E. M. (2004). Which model to use for cortical spiking neurons? *IEEE Transactions on Neural Networks*, Vol.15(No.5):pp.1063–1070.

Kennedy, J. and Eberhart, R. (2001). *Swarm Intelligence*. Morgan Kaufmann Publishers.

Kuroe, Y. (1992). Learning method of recurrent neural networks. *ISCIE Control and Information*, Vol.36(No.10):pp.634–643.

Kuroe, Y. and Ueyama, T. (2010). Learning methods of recurrent spiking neural networks based on adjoint equations approach. *Proc. of WCCI 2010 IEEE World Congress on Computational Intelligence*, pages pp.2561–2568.

Lee, J.H., D. T. and Pfeiffer, M. (2016). Training deep spiking neural networks using backpropagation. *Frontiers in Neuroscience*, Vol.10(Article 508):pp.1–13.

Mass, W. (1996). *On the Computational Complexity of Network of Spiking Neurons*, volume Vol.7. MIT Press.

Mass, W. (1997a). Fast sigmoidal networks via spiking neurons. *Neural Computation*, Vol.9:pp.279–304.

Mass, W. (1997b). Networks of spiking neurons: The third generation of neural network models. *Neural Networks*, Vol.10(No.9):pp.1659–1671.

Mass, W. and Bishop C., E. (1998). *Pulsed Neural Nets*. MIT Press.

Rumelhart, D. E. and McClelland, J. L. (1986). *Parallel Distributed Processing*, volume Vols. 1 and 2. MIT Press.

Selvaratnam, K., K. Y. and Mori, T. (2000). Learning methods of recurrent spiking neural networks - transient and oscillatory spike trains. *Trans. of the Institute of Systems, Control and Information Engineers*, Vol.44(No.3):pp.95–104.