A NOVEL SPIKING PERCEPTRON THAT CAN SOLVE XOR PROBLEM

Jie Yang, Wenyu Yang, Wei Wu[*]

Abstract: In this short note, we introduce a new architecture for spiking perceptron: The actual output is a linear combination of the firing time of the perceptron and the spiking intensity (the gradient of the state function) at the firing time. It is shown by numerical experiments that this novel spiking perceptron can solve the XOR problem, while a classical spiking neuron usually needs a hidden layer to solve the XOR problem.

Key words: *Spiking perceptron, firing time, spiking intensity, XOR problem*

Received: October 15, 2009
Revised and accepted: February 7, 2011

1. Introduction

Spiking neural networks have been used as a *brain-like* tool by neuroscientists for practical applications [1, 2], such as auto-associator, pattern-association, classification and clustering. Formally, a spiking perceptron receives a set of spikes with firing times and fires if the state variable crosses a threshold [3]. In the existing literature, the output of the spiking perceptron is usually taken as the firing time, and the target of training is to learn a set of target firing times for a set of training samples. The XOR classification problem is a well-known benchmark problem for testing the capacity of a certain kind of neural network. It has been shown that a hidden layer is usually included in the spiking perceptron for solving the XOR problem [1, 4, 5]. We recall that a classical feedforward perceptron without a hidden layer, of which the output is the value of a Sigmoid function at the inner product of the weight vector and the input vector, cannot solve the XOR problem either. However, we note that a spiking perceptron obtains its output with much more effort than a classical feedforward perceptron, in that for a given input (a set of spikes), the spiking perceptron has to go through a sequence of temporary outputs by increasing the time parameter until the state variable crosses a given threshold. Therefore, it seems reasonable to expect the spiking perceptron to behave better than the classical feedforward perceptron. For instance, can we modify the traditional spiking neuron a little bit such that it can solve the XOR problem?

[*]Jie Yang, Wenyu Yang, Wei Wu – corresponding author
School of Mathematical Sciences, Dalian University of Technology, Dalian, China, E-mail: `wuweiw@dlut.edu.cn`

A novel spiking neuron is proposed in this paper to give a positive answer to the above question. We introduce a new parameter for the output, i.e., the spiking intensity (the gradient of the state function) at the firing time. Now, the output of the neuron is not the firing time alone, but a linear combination of the firing time and the spiking intensity. We show by numerical experiments that this new spiking perceptron is capable of solving the XOR problem, and we believe that this novel spiking perceptron can be used as a building block to build up more efficient spiking neural networks.

2. Spiking Perceptron

First, let us describe the traditional spiking perceptron defined in [4]. A spiking perceptron has a set of presynaptic neurons that receives and processes the input vector $\mathbf{t} = (t_1, \cdots, t_n)$, consisting of a set of spiking times t_i, as follows:

$$x(t) = \sum_{i=1}^{n} \sum_{k=1}^{K} w_{ik} \varepsilon(t - t_i - d_i^k) \tag{1}$$

where $x(t)$ is called state function, w_{ik} is the k-th synaptic weight between the presynaptic neuron i and the postsynaptic neuron, $\varepsilon(t)$ is an activation function modeling the actual postsynaptic potential to a single spike, t_i is the time of the presynaptic spike of neuron i, and d_i^k is the given synaptic delay of the k-th synaptic sub-connection between i and the postsynaptic neuron. As in [4], the activation function $\varepsilon(t)$ is chosen as:

$$\varepsilon(t) = \begin{cases} \frac{t}{\tau} e^{1-\frac{t}{\tau}}, & \text{if} \quad t > 0 \\ 0, & \text{if} \quad t \leq 0 \end{cases} \tag{2}$$

where τ is the time decay constant that describes how quickly a neuron will respond to an input spike. This function is asymmetrical and has a maximum value of 1 at $t = \tau$ [5]. The firing time t^a of the postsynaptic neuron is defined as the first time when the state variable $x(t)$ exceeds a given threshold v:

$$t^a = \min\{t \mid x(t) \geq v, t \geq 0\}. \tag{3}$$

The target of training a spiking perceptron is to choose the weights such that the perceptron gives a set of desired target firing times of the output neurons for a given set of input patterns.

The idea of our novel spiking perceptron is that the response to the input should be measured not only by the firing time t^a, but also by the firing intensity $x'(t^a)$. Therefore, the output of our novel spiking perceptron is defined as

$$y = f(u_1 t^a + u_2 x'(t^a) - u_3) \tag{4}$$

where t^a is computed by Eq. (3); and the weights u_1 and u_2, the threshold u_3, and the weights w_{ik} are determined through learning as described in the next section. In particular, the activation function for the output neuron is selected as the sigmoid function

$$f(x) = \frac{1}{1 + e^{-x}}. \tag{5}$$

The structure of the new perceptron is shown in Fig. 1. We observe that the lower part of the network described by Eq. (1) and Eq. (3) is the original spiking perceptron, and the upper part described by Eq. (4) is a classical feedforward perceptron.

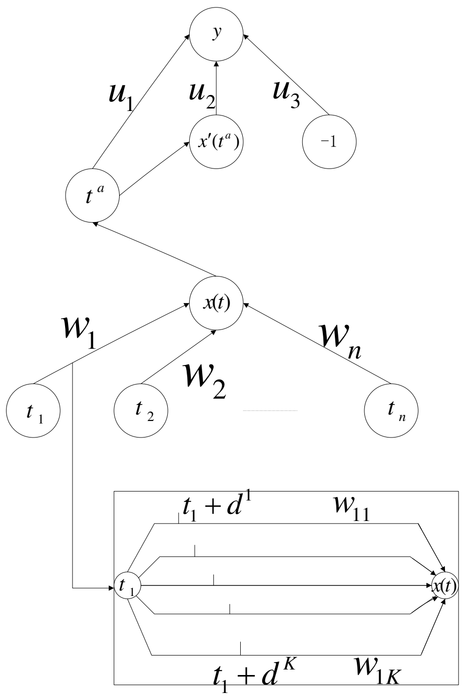

Fig. 1 *Structure of the novel spiking perceptron Eq.* (4).

3. Error-Backpropagation Training Algorithm

We use online gradient descent method for supervised learning of the perceptron. Given is a training pattern set $\{\mathbf{t}^{(s)}, o^{(s)}\}_{s=1}^{S} \in R^n \times R$, where $o^{(s)}$ is the desired output for the input $\mathbf{t}^{(s)}$. A training sequence $\{\mathbf{t}^T, o^T\}_{T=1}^{\infty}$ is generated by randomly choosing each pair $(\mathbf{t}^T, o^T)$ from the training set $\{\mathbf{t}^{(s)}, o^{(s)}\}_{s=1}^{S}$. At the T-th step of the training, we use the gradient descent method to minimize the instantaneous error function

$$E = E(w^T, u^T) = \frac{1}{2}(y^T - O^T)^2 \tag{6}$$

where y^T is the actual output of the neuron for the input $\mathbf{t}^T$ with the present weights $w^T = \{w_{ik}^T\}$ and $u^T = \{u_i^T\}$. Then, the present weights are updated by

$$u_i^{T+1} = u_i^T + \triangle u_i^T \tag{7}$$

and

$$w_{ik}^{T+1} = w_{ik}^T + \triangle w_{ik}^T \tag{8}$$

where

$$\triangle u_i^T = -\eta_1 \frac{\partial E}{\partial u_i^T} \tag{9}$$

$$\triangle w_{ik}^T = -\eta_2 \frac{\partial E}{\partial w_{ik}^T} \tag{10}$$

and η_1 and η_2 are the learning rates.

The computation of $\frac{\partial E}{\partial u_i^T}$ in Eq. (9) is an easy job by using Eq. (4):

$$\frac{\partial E}{\partial u_i^T} = \frac{\partial E}{\partial y^T}\frac{\partial y^T}{\partial u_i^T} \tag{11}$$

Note that the derivative of the sigmoid function Eq. (5) is

$$f'(x) = (1 - f(x))f(x).$$

Hence

$$\frac{\partial E}{\partial u_1^T} = (y^T - O^T)(1 - y^T)y^T t^a \tag{12}$$

$$\frac{\partial E}{\partial u_2^T} = (y^T - O^T)(1 - y^T)y^T x'(t^a) \tag{13}$$

and

$$\frac{\partial E}{\partial u_3^T} = -(y^T - O^T)(1 - y^T)y^T. \tag{14}$$

Next, we compute $\frac{\partial E}{\partial w_{ik}^T}$. Using the chain rule, we have

$$\frac{\partial E}{\partial w_{ik}} = \frac{\partial E}{\partial y}\frac{\partial y}{\partial t^a}\frac{\partial t^a}{\partial x(t^a)}\frac{\partial x(t^a)}{\partial w_{ik}} + \frac{\partial E}{\partial y}\frac{\partial y}{\partial x'(t^a)}\frac{\partial x'(t^a)}{\partial w_{ik}} \tag{15}$$

Here the superscript T has been removed for sake of clear representation. The only trouble here is the computation of the term $\frac{\partial t^a}{\partial x(t^a)}$, because t^a is a functional of the function $x(\cdot)$, and this functional cannot be expressed explicitly. To overcome this difficulty, we follow Bohte et al. [4] to compute the term $\frac{\partial t^a}{\partial x(t^a)}$ as

$$\frac{\partial t^a}{\partial x(t^a)} = -1/x'(t^a). \tag{16}$$

From Eq. (1) the derivative of the state variable is computed as

$$x'(t^a) = \sum_{i=1}^{n}\sum_{k=1}^{K} w_{ik}\varepsilon'(t^a - t_i - d_i^k) \tag{17}$$

where

$$\begin{aligned} &\varepsilon'(t^a - t_i - d_i^k) \\ = &\begin{cases} \frac{1}{\tau}e^{1-\frac{t^a-t_i-d_i^k}{\tau}}(1 - \frac{t^a-t_i-d_i^k}{\tau}), & \text{if } t^a - t_i - d_i^k > 0 \\ 0, & \text{else} \end{cases} \end{aligned} \tag{18}$$

Then, Eq. (15) is rewritten as

$$\begin{aligned} \frac{\partial E}{\partial w_{ik}} = \; & (y^T - O^T)(1 - y^T)y^T[-u_1\frac{1}{x'(t^a)}\varepsilon(t^a - t_i - d_i^k) \\ & +u_2\varepsilon'(t^a - t_i - d_i^k)] \end{aligned} \tag{19}$$

4. Numerical Experiment

In this section, we test the performance of the proposed new spiking perceptron Eq. (4) with two inputs and one output for XOR problem. As a matter of fact, the terminal output of network Eq. (4) is finally computed by a classical feedforward perceptron, so there is no need to encode the output, i.e., the desired output is either 0 or 1. For the input, the coordinate values 0 and 1 are usually encoded as firing times 6 and 0, respectively [1, 4, 5]. To sum up, the time coded inputs and the corresponding desired outputs for the XOR problem are as follows:

Input		Output
0	0	1
0	6	0
6	0	0
6	6	1

In this paper, however, we use analog input values to test the spiking perceptron. First, a set of vectors $\{z_1^s, z_2^s\}_{s=1}^{160}$ are generated by adding small random noises within $[-0.1, 0.1]$ to the four basic input vectors $\{0, 0\}$, $\{0, 1\}$, $\{1, 0\}$, and $\{1, 1\}$. Then, each z_i^s is encoded in the following way:

$$t_i^s = \frac{Max - z_i^s}{Max - Min} \cdot 6 \tag{20}$$

where the values Max and Min are extremal values of z_i^s. Now we get a set of training patterns $\{\mathbf{t}^s, O^s\}_{s=1}^{160}$, among which 80 patterns are used for training and the rest for testing.

The perceptron has $K = 6$ synapses, and the delay $d^k = k$ for $k = 1, 2, \cdots, 6$. The initial synaptic weights are restricted to be positive. But some synaptic weights may become negative later on in the training procedure. Set $\tau = 7$ in Eq. (2). The output weights u_i are initialized as random numbers close to zero. Fig. 2 displays

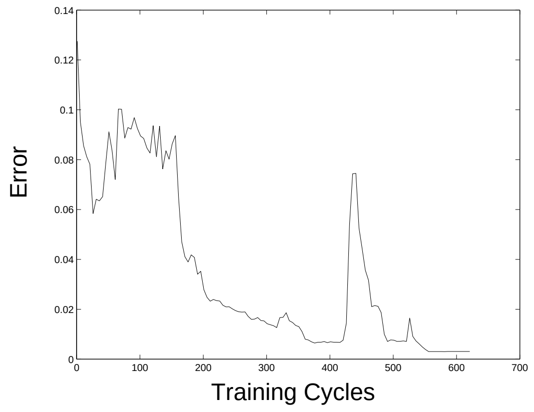

Fig. 2 *The mean square error for XOR problem.*

the result with the learning rate $\eta_1 = 0.04$ and $\eta_2 = 0.01$, and the firing threshold $v = 1$. As shown in the figure, the value of the instantaneous error function Eq. (6) becomes consistently less than 0.003 after about 620 training cycles, and the error-rate (the number of wrongly classified samples divided by the total sample number 80) is zero. In the testing process, each test input vector is supplied to the neuron. The average mean square error is 0.0012 and the classification accuracy is 95%. We can see that our novel spiking perceptron successfully solves the XOR problem.

Acknowledgement

This work was supported by the National Natural Science Foundation of China (10871220) and the Fundamental Research Funds for the Central Universities of China (DUT10JS08).

References

[1] Ghosh-Dastidar S., Adeli H.: A new supervised learning algorithm for multiple spiking neural networks with application in epilepsy and seizure detection, Neural Networks, **22**, 10, 2009, pp. 1419-1431.

[2] Wysoski S. G., Benuskova L., Kasabov N.: Evolving spiking neural networks for audiovisual information processing, Neural Networks, **23**, 7, 2010, pp. 819-835.

[3] Maass W.: Networks of spiking neurons: the third generation of neural network models, Neural Networks, **10**, 9, 1997, pp. 1659-1671.

[4] Bohte S. M., Kok J. N., La Poutr J. A.: Error-backpropagation in temporally encoded networks of spiking neurons, Neurocomputing, **48**, 1-4, 2002, pp. 17-37.

[5] Ghosh-Dastidar S., Adeli H.: Improved spiking neural networks for EEG classification and epilepsy and seizure detection, Integrated Computer-Aided Engineering, **14**, 3, 2007, pp. 187-212.