Neural Networks 96 (2017) 33–46

Contents lists available at ScienceDirect

Neural Networks

journal homepage: www.elsevier.com/locate/neunet

Robust spike-train learning in spike-event based weight update

Sumit Bam Shrestha [a], Qing Song [b,*]

[a] Temasek Laboratories, 5A Engineering Drive 1, #09-02, Singapore 117411, Singapore
[b] School of Electrical and Electronic Engineering, Nanyang Technological University, 50 Nanyang Avenue, Singapore 639798, Singapore

ARTICLE INFO

Article history:
Received 22 February 2017
Received in revised form 17 August 2017
Accepted 22 August 2017
Available online 12 September 2017

Keywords:
Spiking neural network
Multilayer spike-train learning
Supervised learning
Weight convergence
Robust stability
Adaptive learning rate

ABSTRACT

Supervised learning algorithms in a spiking neural network either learn a spike-train pattern for a single neuron receiving input spike-train from multiple input synapses or learn to output the first spike time in a feedforward network setting. In this paper, we build upon spike-event based weight update strategy to learn continuous spike-train in a spiking neural network with a hidden layer using a dead zone on–off based adaptive learning rate rule which ensures convergence of the learning process in the sense of weight convergence and robustness of the learning process to external disturbances. Based on different benchmark problems, we compare this new method with other relevant spike-train learning algorithms. The results show that the speed of learning is much improved and the rate of successful learning is also greatly improved.

1. Introduction

In Artificial Neural Networks (ANNs), we try to mimic the functionality of natural brain with highly abstracted neuron models and a high degree of interconnections between them to perform useful tasks. Usually, the neuron model is abstracted using a non-linear activation function. Spiking Neural Networks (SNNs) are less abstract than these ANNs with non-linear activation functions in the sense that they use spiking neurons as their computational units. Spiking neurons process a series of spikes in time, termed spike-train, as its inputs and produce a spike-train as its output response. In nature, the biological neuron also uses similar spike sequences to receive and transmit information between each other. It is well-known that the currency of information exchange in biological neurons is spike-train. However, how the spike train encapsulates information is a subject of debate. There are two schools of thought on it: *rate encoding* and *pulse encoding*. Traditionally it was understood that the instantaneous rate of spike occurrence holds the totality of the information being transmitted by a spike-train (Butts, Weng, Jin, Yeh, Lesica, Alonso, et al. (2007); Maass (1997); Mainen and Sejnowski (1995); Rullen and Thorpe (2001); Shadlen and Newsome (1994)). It is called rate encoding. The numeric inputs and outputs in most of the ANNs used nowadays are interpreted as instantaneous firing rate. Numerous observations have been recorded in neuroscience, especially in the visual cortex and auditory neurons, where the response of neurons along multiple layers of neurons is so swift that the firing rate interpretation is implausible, thereby suggesting that the precise time of spike carries a substantial amount of information (Butts et al., 2007; Gollisch & Meister, 2008; Mainen & Sejnowski, 1995; Rullen & Thorpe, 2001). This principle is called pulse encoding. In fact, the current thinking in neuroscience is that rate coding and pulse coding are two extremes. The brain network uses a combination of both rate coding and pulse coding, depending upon the situation (Kumar, Rotter, & Aertsen, 2010). The key point is the use of spikes for information transfer, either at precise time or firing at a certain rate or a mix of both. It suggests that SNNs have the potential to process information in an efficient and rapid manner.

SNNs find their applications in the field of neuroscience research (Dethier, Gilja, Nuyujukian, Elassaad, Shenoy, & Boahen, 2011; Markram, 2006; Rom, Erel, Glikson, Lieberman, Rosenblum, Binah, et al., 2007; Webb & Scutt, 2000). They have a promising prospect as a general learning machine as well: theoretical analysis shows that they are computationally at least as capable as a Multi-Layer Perceptron (MLP) (Maass, 1996a,b, 1997; Paugam-Moisy & Bohte, 2011). However, using an SNN as a learning machine poses its set of challenges arising from the complex dynamics of a spiking neuron and an additional time dimension to consider. The supervised learning efforts can be broadly categorized into two groups. First is SpikeProp (Bohte, Kok, & La Poutre, 2002) and its derivatives *viz.* Adaptive SpikeProp (Shrestha & Song, 2013, 2015) Robust SpikeProp (Shrestha & Song, 2017a), Extended SpikeProp (Schrauwen & Van Campenhout, 2004), SpikeProp for multiple spiking neuron (Booij & tat Nguyen, 2005; Ghosh-Dastidar & Adeli, 2009; Xu, Zeng, Han, & Yang, 2013), Resilient

* Corresponding author.
E-mail addresses: tslsbs@nus.edu.sg (S.B. Shrestha), eqsong@ntu.edu.sg (Q. Song).

http://dx.doi.org/10.1016/j.neunet.2017.08.010
0893-6080/

Propagation (RProp) (McKennoch, Liu, & Bushnell, 2006), etc. These methods allow learning in a multilayer architecture. However, they are mostly limited to a single spike. The second group consists of methods that train a single neuron to learn spike-train patterns: Remote Supervised Method (ReSuMe) (Ponulak, 2005; Ponulak & Kasinski, 2009), Chronotron (Florian, 2007), Spike Pattern Association Neuron (SPAN) (Mohemmed, Schliebs, Matsuda, & Kasabov, 2012, 2013), Tempotron (Gütig & Sompolinsky, 2006), Multi-spike Tempotron (Gütig, 2016), precise spike driven synaptic plasticity rule (Yu, Tang, Tan, & Li, 2013), Linear Algebraic Method (Carnell & Richardson, 2005), etc. Note that most of the algorithms mentioned above learn to spike a specific spike train pattern. Tempotron (Gütig & Sompolinsky, 2006), however, learns to spike at least one spike or not spike at all. The exact time of spike is not controlled in any way. Similar is the case for Multi-spike Tempotron (Gütig, 2016), which learns to spike fixed number of spikes corresponding to an input pattern, but not the timing of the spikes. ReSume has also been extended to learn in multilayer architecture (Sporea & Grüning, 2013) using backpropagation rule on the instantaneous firing rate of neurons. In the remainder of this paper, we will focus our attention towards supervised learning methods that can learn spike-train patterns in a multilayer architecture.

For the learning spike-trains in a multilayer spiking neural network, there are multilayer extension of ReSuMe (Multi-ReSuMe) (Sporea & Grüning, 2013) and multi-spike extensions of SpikeProp *viz.* SpikeProp-Multi (Xu et al., 2013) and EvSpikeProp (Shrestha & Song, 2016). Multi-ReSume is independent of the spiking neuron model which makes it versatile. However, it is based on the assumption that the input spike-trains are linearly transformed into output spike-train through linear interaction with weights. In reality, there is a complex non-linear dynamic relation between input spike-trains and the output spike-train it is emitting. On the other hand, the multi-spike extensions of SpikeProp consider the internal dynamics of a spiking neuron. SpikeProp-Multi, however, assumes that the number of desired spikes and the number of actual spikes that occurred in a simulation interval must be same. This condition cannot always be guaranteed in practice. EvSpikeProp circumvents this issue using event based weight update strategy. We will elaborate on the event based weight update strategy in Section 3. Nevertheless, EvSpikeProp faces the convergence and stability issues typical in SpikeProp. In this paper, we will present a dead zone on–off based adaptive learning rate for EvSpikeProp that will ensure convergence and provide robustness against learning disturbances, be it in the form of disturbance in desired spike sample or internal disturbance due to modeling imperfection, and hence stability of the learning process in L_2 and L_∞ space. Overall this approach to stability and robustness is similar to our work on stability and robustness of SpikeProp algorithm (Shrestha & Song, 2017a, b). The main difference with SpikePropR (Shrestha & Song, 2017a) is in the approach of error system analysis. In SpikePropR, we used individual error system in whereas, we use overall total error of the learning system in this paper similar to SpikePropRT (Shrestha & Song, 2017b).

The remainder of the paper is organized as follows. We will start with the preliminaries of spiking neuron model and the SNN architecture in Section 2. Next, we will introduce event-based weight update strategy and propose an adaptive learning rate for EvSpikeProp using learning rate normalization which ensures weight convergence and robustness to disturbances in Section 3. Next, we will formulate the error system network for our new learning method; followed by weight convergence analysis in the presence of external disturbance; and extend the weight convergence result to establish stability of normalized error signals in L_2 space and show the boundedness of error signals in L_∞ space in Section 4. We will then compare the performance of our learning rule with existing methods in terms of different benchmark problems in Section 5, followed by conclusion and notes on future research direction in Section 6.

Notation: in this paper, matrices are denoted by upper case boldface letters (e.g. $\boldsymbol{Z}$); vectors are denoted by lower case boldface letters (e.g. $\boldsymbol{z}$); scalars are denoted by lower case letters (e.g. z); and count of items is denoted by upper case letters (e.g. Z). Given a matrix $\boldsymbol{Z}$, we use $\boldsymbol{z}_{i:}$ to denote the ith row, $\boldsymbol{z}_{:i}$ to denote the ith column and z_{ij} to denote the (i,j)th element. Given a vector $\boldsymbol{z}$, we use z_i to denote the ith element. Sets are denoted by calligraphic letters (e.g. $\mathcal{G}$). The cardinal number of a set $\mathcal{G}$ is denoted by $|\mathcal{G}|$. For matrix–vector calculus, we use denominator layout convention.

2. Spiking neuron and SNN model

2.1. Spiking neuron model

In this paper, we will consider short term memory neuron model (Paugam-Moisy & Bohte, 2011). It is a variant of Spike Response Model (SRM) (Gerstner, 1995) in which refractory response due to the most recent spike only. It is formally presented below.

Consider a neuron N_j which receives input spikes from a set of presynaptic neurons, Γ_j. Every input spike will induce a post synaptic potential (PSP) through multiple synaptic connections and the PSP at the kth synapse depends on the synaptic weight $w_{ji}^{(k)}$ and synaptic delay $d_{ji}^{(k)}$. The most recent spike of N_j at $t = t_j^{(f-1)}$ will produce refractory response as well. The cumulative effect of all the PSP from input spikes and the refractory response form the membrane potential of N_j. Denote the firing times of presynaptic neuron by $\mathcal{F}_i = \{t_i^{(f)}\} : i \in \Gamma_j \wedge t_i^{(f)} + d_{ji}^{(k)} > t^{(f-1)}$. Then the membrane potential is given by

$$u_j(t) = \nu(t - t_j^{(f-1)}) + \sum_{i\in\Gamma_j} \sum_{t_i^{(f)}\in\mathcal{F}_i} \sum_k w_{ji}^{(k)} \varepsilon(t - t_i^{(f)} - d_{ji}^{(k)})$$

$$= \nu(t - t_j^{(f-1)}) + \sum_{i\in\Gamma_j} \sum_k w_{ji}^{(k)} y_{ji}^{(k)}(t, \boldsymbol{t}_i) \tag{1}$$

$$\text{where} \qquad y_{ji}^{(k)}(t, \boldsymbol{t}_i) = \sum_{t_i^{(f)}\in\mathcal{F}_i} \varepsilon(t - t_i^{(f)} - d_{ji}^{(k)}) \tag{2}$$

is the normalized PSP due to N_i, $\varepsilon(\cdot)$ is the spike response kernel and $\nu(\cdot)$ is the refractory response kernel. The choice of spike response kernel and refractory response kernel does not influence the analysis presented in this paper. For simulation results in Section 5, we will use the following version of the kernels:

$$\varepsilon(s) = \frac{s}{\tau_s} e^{1-\frac{s}{\tau_s}} \mathcal{H}(s), \tag{3}$$

$$\nu(s) = -\vartheta e^{-\frac{s}{\tau_r}} \mathcal{H}(s) \tag{4}$$

where $\mathcal{H}(s)$ is the heaviside step function, τ_s is synaptic time constant and τ_r is the refractory time constant.

Whenever the membrane potential $u_j(t)$ rises up to a threshold value ϑ, N_j emits a spike at $t = t_j^{(f)}$ i.e.

$$t_j^{(f)} \;:\; u_j(t_j^{(f)}) = \vartheta,\; u_j'(t_j^{(f)}) > 0,\; t_j^{(f)} > t_j^{(f-1)}. \tag{5}$$

For completeness, denote the function that maps membrane potential of neuron to spike time as

$$t_j^{(f)} = f(u_j) \tag{6}$$

$$\text{and} \qquad \boldsymbol{f}_m(\cdot) = [\cdots, f(\cdot), \ldots]^{\mathsf{T}} \in \mathbb{R}^{|\mathcal{F}|} \tag{7}$$

denotes a function that maps membrane potential into a vector consisting of all the multiple spikes.