Neurocomputing 72 (2009) 685–692

Contents lists available at ScienceDirect

Neurocomputing

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

An analysis of the use of Hebbian and Anti-Hebbian spike time dependent plasticity learning functions within the context of recurrent spiking neural networks

Andrew Carnell

Department of Computer Science, University of Bath, Bath BA2 7AY, UK

ARTICLE INFO

Available online 6 November 2008

Keywords:
LSM
Convergence
Hebbian learning
Spike train storage

ABSTRACT

It is shown that the application of a form of spike time dependent plasticity (STDP) within a highly recurrent spiking neural net based upon the LSM leads to an approximate convergence of the synaptic weights. Convergence is a desirable property as it signifies a degree of stability within the network. An *activity link L* is defined which describes the link between the spiking activity on a connection and the weight change of the associated synapse. It is shown that under specific conditions Hebbian and Anti-Hebbian learning can be considered approximately equivalent. Also, it is shown that such a network habituates to a given stimulus and is capable of detecting subtle variations in the *structure* of the stimuli itself.

1. The liquid state machine

The liquid state machine (LSM) concept was introduced by Maass et al. [10]. The LSM in one of its basic form consists of a pool of highly recurrently connected spiking *leaky integrate and fire* (LIF) neurons.

This pool typically receives a spiking input $n_{input}(t)$ from a pool of input neurons that are connected to a selection of neurons within the LSM. The LSM acts as a medium through which the input can be expressed in a higher dimensional form. A readout neuron which receives a connection from all of the neurons within the LSM is then able to be taught, using a learning algorithm on the connections from the pool, to perform some computable function $F(n_{input}(t))$ on the input, with the strong limiting factor being the size of the LSM, see [1].

1.1. A typical implementation

Analyses of the computational qualities of the LSM typically involve the use of a randomly generated layer of LIF neurons, as shown in Fig. 1, which facilitate the projection of an input into higher dimensions. An input spike train $n_{input}(t)$ is presented to the 'liquid' layer. Learning of a particular function $F(n_{input}(t))$ is then typically accomplished by altering the synaptic weights that connect a readout pool of neurons to this liquid layer. For example, using some learning algorithm it is, in some cases, possible to train a pool of readout neurons to extract from the liquid layer, the information about the input stream $n_{input}(t)$ up to some time into the past τ, that is required to compute a given function $F(n_{input}(t-\tau))$ for different values of τ. The difference in implementation in this paper is that unlike the LSM, which typically has static weights in the recurrent part of the network, the effect of implementing a learning regime in which the synaptic weights can be modified, in a way that is influenced by the firing activity within the network, is investigated.

2. Spike time dependent plasticity learning

Spike time dependent plasticity or STDP learning rules are based upon Hebb's postulate, see [6]. Suppose that a neuron n_{post} receives an input connection from neuron n_{pre}. Consider, that if the pre-synaptic neuron n_{pre} fires before the post-synaptic neuron n_{post}, then the synaptic weight on the link between them is strengthened. In this case n_{pre} can be thought of as contributing to the firing of n_{post}, so its influence is encouraged. If the firing sequence is reversed and n_{pre} fires *after* n_{post} then the connection is weakened.

The case when the synaptic weight is strengthened if n_{post} fires before n_{pre} and weakened if the firing is reversed is known as Anti-Hebbian STDP. The delay between the firing time t_{pre} of n_{pre} and the firing time t_{post} of n_{post} is denoted as $t_{delay} = t_{pre} - t_{post}$ and is the determining factor in how large the synaptic weight change should be. A comprehensive analysis of STDP learning can be found in [2].

E-mail address: csparc@bath.ac.uk

0925-2312/$ - see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.neucom.2008.07.012

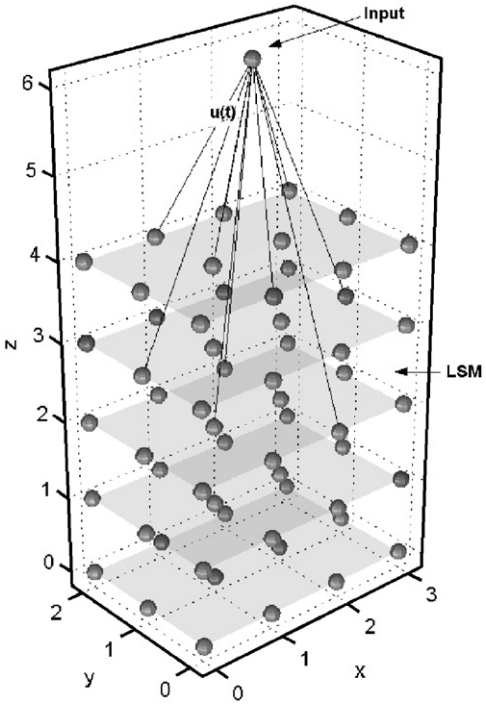

Fig. 1. A basic LSM consisting of LIF neurons with spiking input neuron n_{input}. Each neuron is located on the integer points of a 3-D column.

2.1. The STDP learning window functions

In this paper, an asymmetric Hebbian learning function is used, the form of which can be seen in Fig. 2. This is a typical learning function which increases those synaptic weights whose pre-synaptic neuron fires just before its post-synaptic neuron and weakens the synaptic weight if the post-synaptic neuron fires before the pre-synaptic neuron. The amount of the modification is determined according to the product of the learning window function $S(t_{delay})$ and some learning rate ϕ.

An Anti-Hebbian learning window function is one that operates in reverse to the Hebbian version. In this implementation, for Anti-Hebbian STDP $-\phi$ is used for the learning rate instead of ϕ. It should be noted that there are different forms of STDP learning, see Izhikevich [8] which shows that BCM (Bienenstock, Cooper, Munro) learning can be derived from STDP learning under certain circumstances.

2.2. Weight change and normalisation

Consider a neuron Y within a recurrent neural network, similar to Fig. 1, that receives inputs from neurons $X_1,\ldots,X_k$ with respective synaptic weights $w_1,\ldots,w_k$. W is the weight vector associated with the input of Y, $W=(w_1,\ldots,w_k)$. The individual synaptic weights that comprise W are modified by two processes; a Hebbian weight update that is calculated for $w_1,\ldots,w_k$, and a normalisation procedure which ensures that the input synapses to a neuron have a constant norm, therefore this a competitive Hebbian process, see [4]. The normalisation is applied to W only after all weights $w_1,\ldots,w_k$ have had the Hebbian weight update procedure applied. Network activity is simulated in 0.0002 s time-steps and weights are updated every 0.5 s, using the firing activity of the previous 0.5 s, for the duration of the simulation. These update procedures are performed on the weight vectors of each neuron of the network at each time-step. Suppose R is the norm of the original unmodified weight vector W. For a neuron X_i firing at time t_i and a neuron Y firing at time τ, we get

$$w_i := w_i + \phi S(t_i - \tau) \quad \text{for } i = 1,\ldots,k$$

$$W := R * W/\|W\|$$

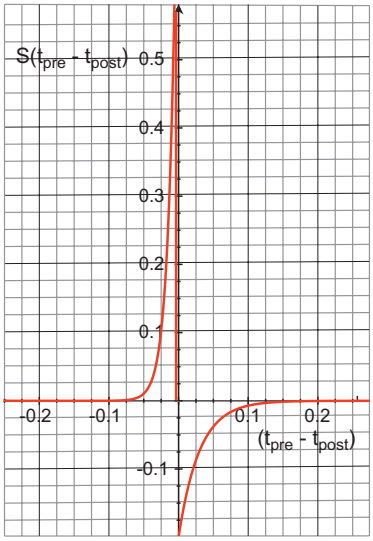

Fig. 2. The form of the learning window function used to calculate the weight change due to STDP. The actual weight update value is given by the product of this learning function $S(t_{delay})$ and the learning rate ϕ. The time scale of the learning window function is such that the peak occurs at a distance of 2.5 ms from the post-synaptic firing time. This value was chosen in accordance with the finding by Gerstner in [4] that the learning window function maximum $S(t_{delay})_{max}$ should be in the range $\delta/2 \leqslant S(t_{delay})_{max} \leqslant \delta$, where δ is the rise time of a post-synaptic spike, typically less than 5 ms. STDP is a biologically derived learning method, but there are many unanswered questions as to what actually happens when STDP is applied, especially in the context of recurrent spiking neural networks. An axonal time-delay is incorporated into a synaptic delay on the transmission of a spike to the receiving neuron. This synaptic time-delay was drawn from a Gaussian distribution with a mean of 0.0015 s for excitatory-to-excitatory neuron connections, and a mean of 0.0008 s for all other connections, and with a standard deviation of 10% of the mean.

2.3. Activity link

In order to describe better the weight change brought about by the Hebbian update, and as a means for describing the link between the spiking activity on a synaptic connection and the weight change of the synapse associated with that connection, the activity link $L(a,b)$ is defined where a and b are spike trains. Suppose that a has spike times $t_1,\ldots,t_m$ and b has spike times $\tau_1,\ldots,\tau_n$, then we can define the following:

$$L(a,b) = \sum_{i:=1}^{m}\sum_{j=1}^{n} S(t_i - \tau_j)$$

Consider neuron Y from the previous section. Suppose that the spike trains of $X_1,\ldots,X_K$ are $x_1,\ldots,x_k$ and Y has a spike train y over an interval I. The update rules for the Hebbian learning followed by the normalisation update are

$$w_i := w_i + \phi L(x_i, y) \quad \text{for } i = 1,\ldots,k$$

$$W := R * W/\|W\|$$

3. LSM generation parameters

All experiments are performed using CSIM, see [7], under MATLAB. CSIM allows for the simulation of many types of neuron and synapse models, and enables the creation of pools of recurrently connected neurons.