Principles of Brain Computation, SS17

Robert Legenstein
Institute for Theoretical Computer Science
Graz University of Technology
A-8010 Graz, Austria
robert.legenstein@igi.tugraz.at

April 25, 2017

Last Name	First Name	Matrikelnmr	Team members

Handed out on 25.4.2017
Hand in on 18.5.2017

Task 3: Spike-timing-dependent plasticity (20+5* Points)

Use this page as the cover sheet of your submission.

In this exercise we use the NEST simulator to investigate the properties of spike-timing-dependent plasticity (STDP).

Spike-timing-dependent plasticity

The standard STDP rule specifies the change $W(\Delta t)$ of the synaptic weight of an excitatory synapse in dependence on the time difference $\Delta t = t_{post} - t_{pre}$ between the firing times t_{pre} and t_{post} of the pre- and postsynaptic neuron. It is commonly modeled by a so-called learning window of the form

$$W(\Delta t) = \begin{cases} A_+ e^{-\Delta t/\tau_+} & , \quad \text{if } \Delta t \geq 0 \\ -A_- e^{\Delta t/\tau_-} & , \quad \text{if } \Delta t < 0 \end{cases} , \tag{1}$$

where the positive constants A_+ and A_- scale the strength of potentiation and depression respectively, and τ_+ and τ_- are positive time constants defining the width of the positive and negative learning window. NEST defines a "learning rate" λ (scales positive increments) and a factor α such that $A_+ = \lambda$ and $A_- = \alpha\lambda$. The resulting weight change at time t of synapse ji for a presynaptic spike train S_i^{pre} and a postsynaptic spike train S_j^{post} is usually modeled by the instantaneous application of this learning rule to all spike pairings with the second spike at time t

$$\left[\frac{d}{dt} w_{ji}(t)\right]_{STDP} = \int_0^\infty dr\ W(r) S_j^{post}(t) S_i^{pre}(t-r) + \int_0^\infty dr\ W(-r) S_j^{post}(t-r) S_i^{pre}(t). \tag{2}$$

The spike train of a neuron i which fires action potentials at times $t_i^{(1)}, t_i^{(2)}, t_i^{(3)}, \ldots$ is formalized here by a sum of Dirac delta functions $S_i(t) = \sum_n \delta(t - t_i^{(n)})$.

Network model

The network is composed of 200 input neurons (generated in the function `construct_input_population`) connected to a leaky integrate-and-fire neuron (use the model `iaf_psc_exp`). The synapses at the connections are current-based synapses with an exponential kernel for the injected current and exhibit STDP according to the model described in the previous section (model `stdp_synapse`).

We consider two types of input:

Synchronous input: (argument `sequence=False`). The input neurons fire Poisson spike trains at a constant rate of 8Hz. Additionally to this underlying Poisson firing, half of the input neurons, i.e. the first 100, fire also when a stimulus event occurs (see Fig. 1 and 2). The stimulus events occur randomly in time according to a Poisson process with a constant rate of 2Hz. The time of the l^{th} stimulus event is denoted by $t_{\text{stim}}^{(l)}$. For the l^{th} event, each of the first 100 input neurons fire close to $t_{\text{stim}}^{(l)}$, but with a jitter drawn from a normal distribution with mean 0 and standard deviation σ. That is, for stimulus event l, input neuron j spikes at a time drawn from $t_s^{(l)} + \mathcal{N}(0, \sigma)$ where $\mathcal{N}(0, \sigma)$ is a normal distribution with mean 0 and standard deviation σ. The value of the standard deviation of the jitter is given in the Questions section.

Sequential input: (argument `sequence=True`). Same as above, but half of the input neurons, i.e. the first 100, fire additionally *in a sequence* when a stimulus event occurs. For the l^{th} event, neuron i of the first 100 input neurons fires close to $t_{\text{stim}}^{(l)} + i \cdot t_{\text{delay}}$, where $t_{\text{delay}} = 1$ ms (these timings can be jittered as above).

For the LIF neuron the following values of the parameters should be used: $V_{thresh} = -45$ mV, $V_{resting} = V_{reset} = -60$ mV, $R_m = 1$ MΩ, $C_m = 30$ nF, $t_{refract} = 2$ ms.

All synapses should have time constant set to $\tau_s = 10$ ms (this has to be set in the postsynaptic neuron) and initial synaptic weight set to $w_{init} = 2 \cdot 10^3$ pA.

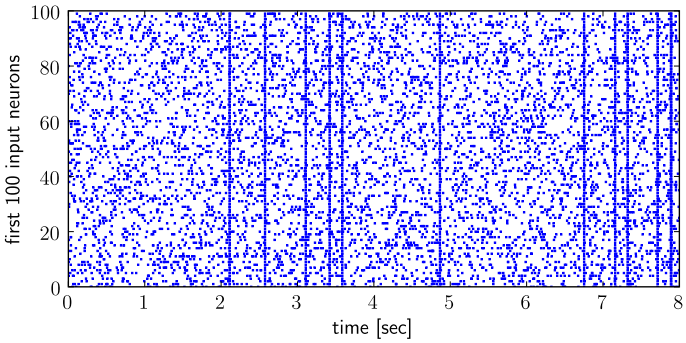

Figure 1: Spiking activity of the first 100 input neurons. In addition to the constant rate Poisson firing, each time a stimulus occurs all the neurons fire at times with a jitter around the stimulus time.

The STDP parameters of the synapses should be set as follows: $\lambda = 0.005$ and $\tau_+ = \tau_- = 30$ ms (τ_- has to be set in the postsynaptic neuron). The value of $\alpha = \frac{A_-}{A_+}$ will be given in the Questions section. The maximum value of the synaptic weight W_{max} should be set to twice the initial synaptic weight w_{init} (if not specified differently in the questions section). When initializing the synapses, set $\mu_+ = \mu_- = 0$ to obtain standard additive STDP.

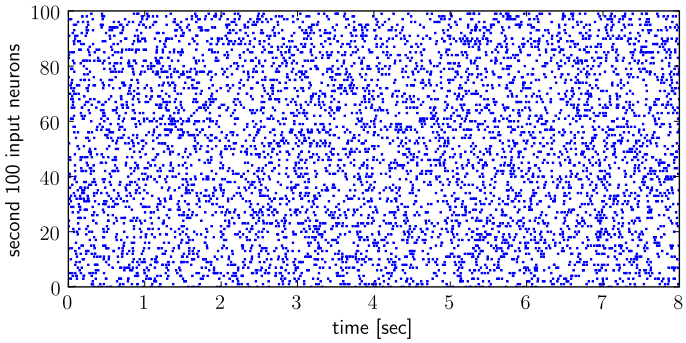

Figure 2: Spiking activity of the second 100 input neurons. All neurons emit Poisson spike trains at constant rate of 8Hz.

Template Script

On the web page you find two scripts, `ex3_stdp.py` that you are going to use as a template to implement the model in the exercise, and `pobc_utils.py` script with additional utility functions used in `ex3_stdp.py`. Make sure `pobc_utils.py` is accessible from `ex3_stdp.py` when you execute the script.

The script `ex3_stdp.py` contains the function `perform_simulation(sequence,jitter,alpha,Wmax_fact,Tsim,W)` where you should implement the model. The arguments in the function are the parameters that you are going to change while working on the exercise:

sequence: if `False`, the stimulus appears in input neurons at the same time (and is then jittered). If `True`, the simulus spike is induced in neuron i delayed by i msec (and is then jittered).

jitter: standard deviation of the normal distribution used to jitter the stimulus-induced spike times of input neurons.

alpha: α parameter giving the ratio of the strengths of the depression and potentiation part of the STDP window

W: Initial weights w_{init} of synapses.

Wmax_fact: The maximum weight is W_{max} = `Wmax_fact` $\cdot w_{init}$ for each synapse. If STDP attempts to increase the weight beyond that value, it is going to be clipped at W_{max}.

Tsim: is the simulation time (in seconds)

Currently the body of the function holds only the creation of input neurons and generation of their spikes. You should complete the implementation of this function by constructing the model described in the previous section, simulating it and gathering the recorded signals during the simulation. The output of the function consists of three variables:

spikes: numpy array of the recorded spike times of the LIF neuron

inp_spikes: list of numpy arrays of the spike times of the input neurons. Use the function `get_spike_times(spike_recorder)` (definied in `pobc_utils.py`) with a recorder that records the spikes of the input neurons to extract spikes in this format.

weights two dimensional numpy array with the recorded synaptic weights of all 200 input STDP synapses of the LIF neuron (each column corresponds to the weight vector at one time step). Record weights once every simulated second.

The function `plot_raster(spikes,tmax)` plots a raster plot for the given spike-datastructure. The function
`plot_figures(f1, f2, spikes, w, inp_spikes, Tsim, fn_f1, fn_f2, Tmax)` plots two figures from the data provided in the input arguments. Arguments:

f1, f2: integer IDs of the two figures

spikes: array of spike times of the LIF neuron

inp_spikes: list of numpy arrays of the spike times of the input neurons

weights: two dimensional numpy array with the recorded synaptic weights of the input synapses of the LIF neuron

Tsim: is the simulation time (in seconds)

fn_f1, fn_f2: Figures are saved under these file names

Tmax: Computation of cross-correlations (see below) may take quite long. Hence, we compute the correlations only over times $(0, T_{max})$. The default value (25 seconds) seems OK.

The first figure has two panels, on panel A there is a plot with the evolution of the synaptic weights as a function of time during the simulation. Panel B plots the average and the standard deviation of the synaptic weights as a function of time of the synapses connecting from the first 100 input neurons (blue color) and separately for the synapses connecting from the second 100 input neurons (red color).

The second figure has 4 panels, with cross-correlogram plots calculated from the spike trains of the input neurons and the LIF neuron. Panels A and B plot input-input cross-correlograms defined as the histogram of time differences between the spike times from two input neurons (5 ms bin size), calculated and summed over a set of randomly chosen pairs of input neurons (Panel A: first 100 input neurons; Panel B: second 100 input neurons). Panels C and D plot input-output cross-correlograms defined as the histogram of spike differences between the spike times of an input neuron and the spike times of the LIF neuron, calculated and summed over a subset of input neurons (Panel C: first 100 input neurons; Panel D: second 100 input neurons).

Cross-correlogram. The cross-correlogram plots give an estimate of the distribution of the time differences of the spike times of two neurons $C_{ji}(\Delta t)$, on the time scale comparable to the width of the STDP window function. One can approximately estimate the cummulative synaptic weight change induced by STDP during the simulation by summing the products of the cross-correlogram and the STDP window function at the time moments of the bins of the histogram

$$\Delta w_{ji} \approx \sum_{k=-L}^{L} W(k\Delta b) C_{ji}(k\Delta b)$$

where Δb is the size of the bin, and $[-L\Delta b, L\Delta b]$ is the range where the $W(s)$ has significant, non-negligible values. We have chosen $L\Delta b = 100$ ms in the plots. You should use the cross-correlogram plots for the interpretation of the cummulative synaptic weight changes in the simulations of the model.

Note that the cross-correlograms plotted in the figures are summed for multiple pairs of neurons, but since all the pairs of neurons have the same firing statistics (the first 100 input neurons have the same statistics, as well as the seconds 100 input neurons) the cross-correlation for all pairs is the same, and the summing is equivalent as calculating the cross-correlogram of the spiking outputs of just two neurons from the sets but for a longer period of time. Therefore, we are summing over multiple pairs of neurons with the same statistics in order to get a better estimate of the cross-correlogram for two neurons.

Questions

a) (5 points) Expand the `perform_simulation` function to include the construction and simulation of the model described above. Set `sequence=False` in a-c. In particular you should

- create the LIF neuron and the synaptic connections.
- setup recorders that will record the spiking of the LIF neuron and the input population. Record synaptic weights every simulated second.
- perform a simulation of the model. The time step of the simulation should be set to $DT = 0.1$ ms.

b) (5 points) Perform the simulation (for 200 simulated seconds) by setting `jitter` = 2 ms and `alpha` = 1.1. Use the `plot_figures` function to plot the figures. Explain the temporal evolution of the weights for the two groups of input neurons that you observe in Figure 1. Use the results in the cross-correlogram plots in Figure 2 in your explanation.

c) (5 points) Investigate the influence of the `alpha` parameter. Set this parameter to different values $1.0, 1.3, 2.5$. Perform simulations of 200 simulated seconds. How does the temporal evolution of the synaptic weights change? Perform the same analysis as in the previous point by explaining/interpreting the results and providing supporting figures.

d) (5 points) Set jitter to `jitter`=0 ms and `sequence=True`. Set the `alpha` parameter to 1.1. Perform simulations for 200 simulated seconds. How does the temporal evolution of the synaptic weights change in this case? Describe and explain the result. Set the weight factor `Wmax_fact` to 1.5. How does this change the result and why?

e) (5* points) Perform self-designed simulations that specifically help to explain the results in points c and d.

Submit your plots and interpretations on paper, and the code by email to `robert.legenstein@igi.tugraz.at` and `m.mietschnig@student.tugraz.at` (**include "PoBC ex3 submission" in the subject of the email**) until 8am on the day of submission. Write readable code with informative comments. All members of the same team are allowed to submit the same code (but not the same report). Do not send the report by email but hand in a printout.