Principles of Brain Computation KU

708.086 18S

Homework Sheet 4

Problems marked with * are optional.

Spike timing-dependent plasticity [8+1*P]

In this task, we use NEST to investigate the effects of spike timing-dependent plasticity on the synapses of a single LIF neurons.

The model we will use (cf. Fig. 1) consists of two groups of 100 input neurons each which spike randomly (background firing). Neurons within the first group also receive additional input which triggers special events (synchronized or sequential firing of all neurons in the group).

These input neurons are connected to a single LIF neuron via plastic synapses with STDP. We will investigate the changes of the weights of the two groups of input neurons over time.

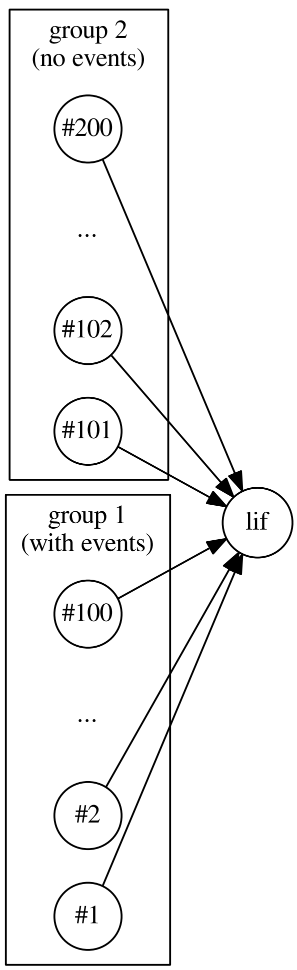

Figure 1: Network architecture.

Model details

All input neurons generate spikes according to Poisson point processes with a constant rate $r_{\mathrm{background}} = 8\,\mathrm{Hz}$. Neurons in the first group additional fire in an organized fashion, these events occur randomly (Poisson process with rate $r_{\mathrm{event}} = 2\,\mathrm{Hz}$).

The spike input for the network is pre-generated and can be fed into NEST using `spike_generator` nodes. Details on the plastic synapses are given below. Since these spike generators cannot have plastic synapses, you need to use `parrot_neuron`s as input neurons and connect one spike generator to each of them.

The LIF neurons should have exponential PSCs and use parameters $u_{\mathrm{rest}} = -65\,\mathrm{mV}$, $R_{\mathrm{m}} = 0.001\,\mathrm{G\Omega}$, $C_{\mathrm{m}} = 20000\,\mathrm{pF}$, $\vartheta = -45\,\mathrm{mV}$, $u_{\mathrm{reset}} = u_{\mathrm{rest}}$. Set the absolute refractory period to 2 ms. The time constant of the PSCs ($\tau_{\mathrm{syn}} = 10\,\mathrm{ms}$) also needs to be set in the neuron model.

Synapse model

We use the NEST model `stdp_synapse` for dynamical synapses, which implements a phenomenological model of STDP. The change of a synapse's weight arising from a pre- and a postsynaptic spike depends on the timing difference $\Delta t = t_{\mathrm{post}} - t_{\mathrm{pre}}$ and the current weight value w and is computed as

$$\Delta W(\Delta t, w) = \begin{cases} \lambda\,(1-w)^{\mu_+}\,\mathrm{e}^{-|\Delta t|/\tau_+} & \text{if } \Delta t \geq 0 \\ -\lambda\,\alpha\,w^{\mu_-}\,\mathrm{e}^{-|\Delta t|/\tau_-} & \text{if } \Delta t < 0 \end{cases}, \tag{1}$$

where

- τ_+ and τ_- are time constants governing the width of the learning window,
- μ_+ and μ_- determine the weight dependency of the update,
- α sets the shape of the depression part relative to the facilitation part, and
- λ is a learning rate.

We will use updates without weight dependency ($\mu_+ = \mu_- = 0$) and a learning rate of $\lambda = 0.005$. The time constants should be $\tau_+ = \tau_- = 40\,\mathrm{ms}$ (note that in NEST, τ_- has to be set in the postsynaptic neuron). Furthermore, we will use initial weight values of $w_0 = 2000\,\mathrm{pA}$ for all synapses and maximum weight values of $2 \cdot w_0$.

Input spikes

Input spikes for the model are generated by the function `generate_stimulus`, which generates a list containing a spike list for each input neuron. Different types of events can be set depending on the parameters:

- If `sequence=False`, all neurons in the first group will fire at the same time when an event occurs. Each spike can be jittered using the `jitter` parameter which sets the standard deviation to use.
- If `sequence=True`, the neurons in the first group fire sequentially with a delay of 1 ms per neuron, i.e. neuron 1 fires at t_0, neuron 2 at $t_0 + 1\,\mathrm{ms}$, neuron 3 at $t_0 + 2\,\mathrm{ms}$, and so on (for an event at time t_0). Again, jitter can be added.

Fig. 2 shows a spike raster for 5 s of input spikes generated with `sequence=False` and `jitter=0`.

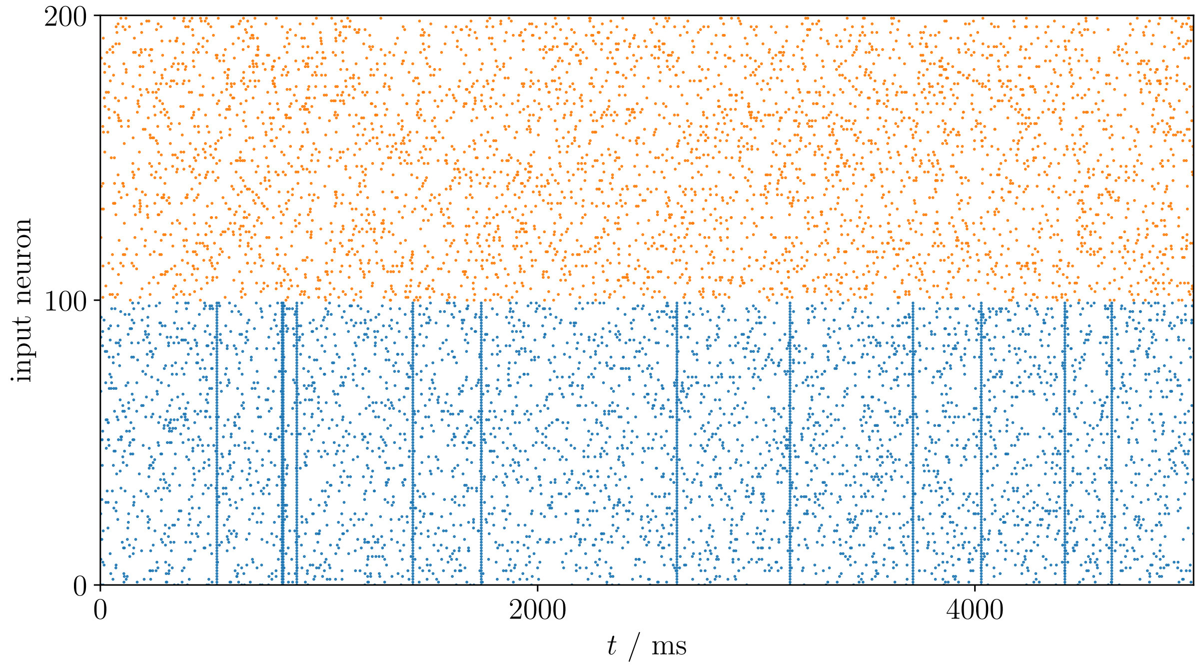

Figure 2: example spike input to the LIF neuron. Neurons 1–100 (blue) form group 1, neurons 101–200 (orange) group 2 (cf. Fig. 1).

Cross-correlogram

The provided code plots distributions of the time difference of spikes of two neurons on the time scale comparable to the width of the STDP window. The correlations of input spikes (i/i correlation) shows temporal relationships between spikes within the two groups of input neurons. The input-output (i/o) correlations relates spikes in these groups to the spikes of the postsynaptic (LIF) neuron. Each correlogram gives a histogram of timing differences (between different input neurons in the i/i case, and between input neurons and the LIF neuron in the i/o case).

Using the cross-correlogram, one can approximately estimate the cumulative synaptic weight change induced by STDP: time differences as shown in the correlogram lead to weight updates given by the learning window at the corresponding timing difference. The time axis gives the difference between spikes in the second set and the first set, e.g. for the i/o correlogram of group 1, we have $\Delta t = t_{\mathrm{LIF}} - t_{\mathrm{grp1},k}$.

The cross-correlograms plotted by the code are computed using the spikes of 200 randomly drawn pairs of neurons (in the i/i case, for the i/o correlogram, all possible pairs are considered). Only spikes within the first 25 s are used to save computation time. Since all neurons within a group have the same statistics, this correlogram gives us a better estimate for the cross-correlogram between two individual neurons over a longer period of time.

Code

The provided template script performs the simulation and perform the required analysis (computing mean weights, correlograms, etc.). You need to complete the `experiment` function: set up the network nodes and the connections as described above at the indicated location (between `TODO beginning` and `TODO end` in the code). You can then call the `experiment` function in the main block for all subtasks described below. You can also put your code for task 4a there.

Note: to speed up execution, the code uses the Python library `joblib` (which has to be installed with pip: run `pip3 install --user --upgrade joblib` in the console). If you do not want to use it, you need to change two things: delete the import line and search for `use_mp=True` in the code and change the value to `False`.

Task 4a [1P]

Plot the learning window for the given parameters and $\alpha = 1.1$. Note that you should explicitly define Δt so the plot is unambiguous. Briefly discuss the salient features.

Task 4b [2P]

Run an experiment with `sequence=False` and `jitter=0` using $\alpha = 1.1$. Plot the resulting input spikes, weight development, and correlations. Explain the weight development of the two groups of input neurons using the correlation statistics.

Task 4c [2P]

Re-run the same experiment using jitter of 5 ms and 15 ms. What has changed? Plot the resulting weight development and correlation statistics and explain how the differences to the results in the previous task arise.

Task 4d [1P]

Set `sequence=True`. Run the experiment with `jitter=0`. Plot the input signal and the weight development. Explain the time course of the weight development.

Task 4e [1P]

Finally, set $\alpha = 0$ and re-run the experiment with `sequence = False` and a jitter of 50 ms. Explain the development of the weights over time.

Task 4f [1*P]

We saw above in the second case of task 4c (simultaneous spiking with a jitter of 15 ms) that the mean weight value of the neurons in group 1 decreases at first, but then starts to increase after a while. Give an explanation for this phenomenon.

Submit the code until 8:00 AM of the day of submission to `mueller@igi.tugraz.at` and `lydia.lindner@student.tugraz.at`. Use `PoBC HW4,` ⟨`name team member 1`⟩ ⟨`name team member 2`⟩ as email subject. Only one email per team is necessary. Submit regular Python code files (`*.py`). You need to hand in a printed version of your report at the submission session. Each team member needs to write their own report. Use the cover sheet provided on the course website.