Assignment – Examining the STDP mechanism

Mattias Borg, 2020-03-17

This assignment will train you to use a very useful Python software package called Brian2 (which will also be used for the Project work later in the course) as well as playing around with the dynamics of the STDP learning mechanism and comparing it to pure Hebbian learning. There is no "correct" answers in this exercise, but the idea is to gain an understanding of the importance of each parameter. Your results for each Task should thus be motivated carefully and clearly in a report which is to be submitted on Canvas.

Download and open "EITP25_STDP_assignment.py" into Python and examine the code

The first important definitions are the equations for the synapses:

```
# STDP equations
Apre = Apost = 0.03
wmax = 1.0
tau_pre = 20*ms
tau_post = 35*ms #depression decays slower
stdp_eq = '''
    w : 1
    dapre/dt = -apre/tau_pre : 1 (event-driven)
    dapost/dt = -apost/tau_post : 1 (event-driven)
    '''
on_pre_eq = '''
    v_post += w
    apre += Apre
    w = clip(w-apost, 0, wmax)
    '''
on_post_eq =  '''
    apost += Apost
    w = clip(w+apre,0, wmax)
    '''
```

This is an event-based way of defining STDP behaviour, which we will go through now. Here, stdp_eq defines the weight variable w (unit 1) and two decaying state diff equations for a_{pre} and a_{post} as:

$$\frac{da_{pre}}{dt} = -\frac{a_{pre}}{\tau_{pre}}, \frac{da_{post}}{dt} = -\frac{a_{post}}{\tau_{post}}$$

These states decay over time and are increased upon preneuron spike (a_{pre} by an amount A_{pre}) and postneuron spike (a_{post} by an amount A_{post}) as is defined in on_pre_eq and on_post_eq, respectively.

On a pre-spike event the post-neuron potential is increased by w, and w in turn is decreased by a_{post} (clipped between 0 and 1). Thus if a post-event just occurred then a_{post} is large and the weight is strongly decreased, otherwise the impact is minimal.

On a post-spike event w is increased by a_{pre} , which if a pre-event just happened has not decayed much and will strongly increase the weight.

The next definition is that of the neuron model:

```
#Neuron differential equations

tau_v = 20*ms #20ms
T = 1.0

# neuron model
eqs = '''
dv/dt = -v/tau_v : 1
'''
thres = 'v > T'
res = '''
v=0
'''
```

Here we define the membrane potential v diff equation in `eqs`:

$$\frac{dv}{dt} = -v/\tau_v$$

Which describes the "leakiness" of the neuron. thres defines what the condition for spiking should be, set to $v > T = 1.0$, but this value you may play around with.

Finally we build the network:

```
#INPUT LAYER
num_inputs = 10
input_rate = 50*Hz
P = PoissonGroup(num_inputs, rates=input_rate)

#SINGLE POSTNEURON
G = NeuronGroup(1, eqs, threshold=thres, reset=res, method='exact')

#SYNAPSES
S = Synapses(P, G, stdp_eq, on_pre=on_pre_eq, on_post=on_post_eq)
S.connect()
S.w = 'rand()*wmax' #random initial weight assignment

#MONITORS
M = PopulationRateMonitor(G)
WM = StateMonitor(S,'w', record=True)
```

Here we define the preneurons as a number (`num_inputs`) of Neurons firing randomly with an average frequency of `input_rate` (50 Hz). These are represented by the PoissonGroup object P.

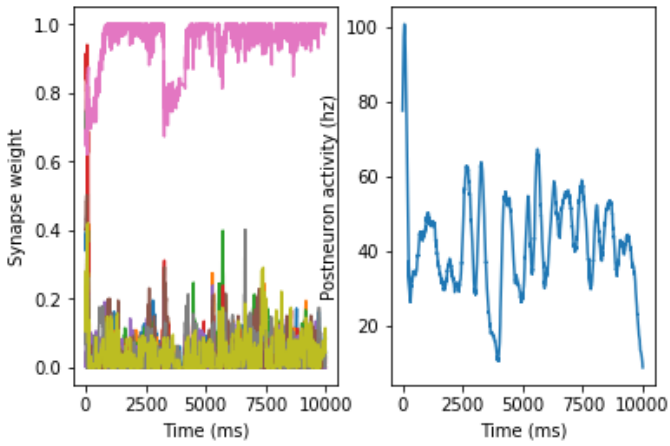
The single postneuron is defined as a NeuronGroup object G defined by the equations eqs, thres and res.

The Synapses object S connects the two neuron groups P and G, and uses the synapse models we defined in equations stdp_eq, on_pre_eq and on_post_eq. We connect all inputs to the output with S.connect(), and then sets the weights to a random value in the range [0, w_{max}].

Finally, we define two Monitors that record things during the simulation. This is a convenient way to collect data to use to analyse the network afterwards. For large simulations they slow down the simulation too much, however.

The PopulationRateMonitor M measures the spiking rate of G as a function of simulation time. The StateMonitor WM records the synapse weights, and how they change over time.

Task 1. Run the code as it is and see the output. It should look something like the following:



Each color in the left plot indicates the weight of one synapse. The graph to the right indicates change in the postneuron spiking rate over time.

Question: If we had only 1 pre-neuron in the network, for a 50 Hz average input rate and threshold of 1, what is the rough firing rate that we could expect?

Task 2. Vary the value of the variable tau_post.

Question: What is the impact on the weights and activity? Why?

Task 3: Vary the magnitude of Apre and Apost (while keeping them equal).

Question: What is the impact on the weights and activity? Why?

Task 4: Vary the threshold T.

Question: What is the impact on the weights and activity? Why?

Task 5: Vary the input rate and the number of input neurons

Question: What is the impact on the weights and activity? Why?

Task 6: Alter the synaptic learning rules to conform with pure Hebbian learning instead:

```
on_pre_eq = '''
    v_post += w
    apre += Apre
    w = clip(w+apre*apost,0, wmax)
    '''
on_post_eq =  '''
    apost += Apost
    '''
```

Question: What is the impact on the weights and activity? Why?

Task 7: Now alter this Hebbian rule to use Oja's rule instead.

Question: What is the impact on the weights and activity? Why?