University of Tehran

COMPUTER SCIENCE DEPARTMENT

COMPUTATIONAL NEUROSCIENCE

REPORT 5

FlatSTDP and FaltRSTDP

Student Name	Student ID
AMIR NADERI	610398126

Lecturer in charge:

M.Ganjtabesh

Submission Date : 25/07/2023

1 Introduction

In this project we are going to analyze the two rules of learning, FlatSTDP and FlatRSTDP. Here we make a model with two layers which the neurons are LIF, the first layer gets the inputs (spike trains) and the second layer includes two neurons which they try to learn the patterns. The connections between layers is fully connected, also we have two different patterns generated from poisson distribution. We injected the inputs to the first layer and let the output layer which has two neurons to learn the patterns using the FlastSTDP and FlatRSTDP learning rules thorough iterations. At the end we analyzed the changes of weights through the process of training and see the results when parameters changes.

2 LIF Neuron Model

The leaky integrate-and-fire model which can be traced back to Louis Lapicque, contains, compared to the non-leaky integrate-and-fire model a "leak" term in the membrane potential equation, reflecting the diffusion of ions through the membrane. It takes the sum of weighted inputs, much like the artificial neuron. But rather than passing it directly to an activation function, it will integrate the input over time with a leakage, much like an RC circuit. If the integrated value exceeds a threshold, then the LIF neuron will emit a voltage spike. The model equation looks like:

$$\tau.\frac{du}{dt} = -(u - u_{rest}) + R.I(t) if\ u(t) = \theta Fire + Reset(u = u_{reset}) \tag{1}$$

$$if\ u(t) = \theta \longrightarrow\ Fire + Reset\ (u = u_{reset}) \tag{2}$$

Here are the explanation of the parameters:

- τ: time constant controlling the rise and decay-time.
- u: voltage across the cell membrane.
- R: membrane resistance. The non-leaky integrate-and-fire model is retrieved in the limit Rm to infinity.
- θ: threshold, when the voltage of a neuron reaches to the threshold, the neuron will fire and the will reset.
- u_{rest}: the resting potential of the neuron.

The biggest disadvantage of the Leaky integrate-and-fire neuron is that it does not contain neuronal adaptation so that it cannot describe an experimentally measured spike train in response to constant input current. This disadvantage is removed in generalized integrate-and-fire models that also contain one or several adaptation-variables and are able to predict spike times of cortical neurons under current injection to a high degree of accuracy.

3 STDP

Stimulation can induce changes of the postsynaptic response that last for hours or days:

- The stimulation leads to a persistent increase of the synaptic efficacy is called long-term potentiation of synapses (LTP).
- If the result is a decrease of the synaptic efficacy, it is called long-term depression (LTD).

Now what is STDP ?
Spike-timing-dependent plasticity (STDP) is a biological process that adjusts the strength of connections between neurons in the brain or a computational model. The process adjusts the connection strengths based on the relative timing of a particular neuron's output and input action potentials (or spikes). The STDP process partially explains the activity-dependent development of nervous systems, especially with regard to long-term potentiation and long-term depression. So the STDP function shows the change of synaptic connections as a function of the relative timing of pre-synaptic and post-synaptic spikes. You can see the function in the picture below:

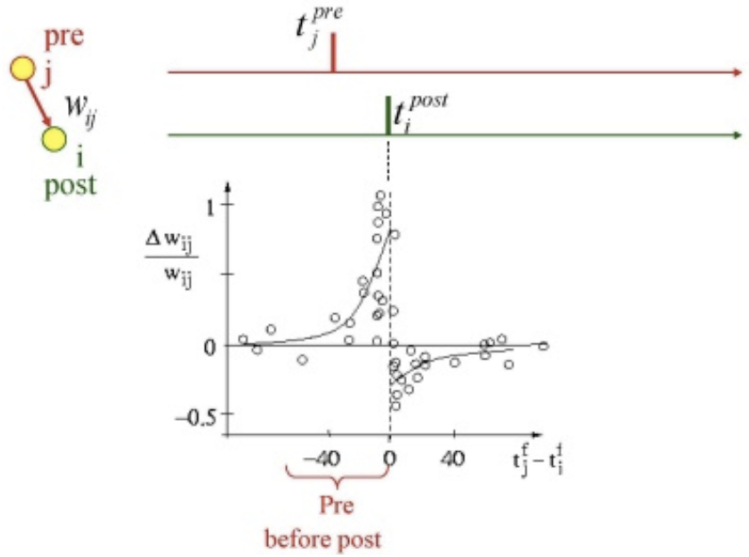


Figure 1: STDP Function

According to the Hebbian rule, synapses increase their efficiency if the synapse persistently takes part in firing the postsynaptic target neuron. Similarly, the efficiency of synapses decreases when the firing of their presynaptic targets is persistently independent of firing their postsynaptic ones.
The change in weight of a synapse depends on the temporal difference $|\Delta t| = |t_{post} - t_{pre}|$:

$$\Delta W_+ = A_+(W).exp(-|\Delta t|/\tau_+) \ at\ post, \quad for\ t_{pre} < t_{post}$$

$$\Delta W_- = A_-(W).exp(-|\Delta t|/\tau_-) \ at\ pre, \quad for\ t_{pre} > t_{post}$$

It happens immediately after each spike at times t_{pre} and t_{post}. This rule is fully specified by defining the weight-dependence of the amplitude parameter $A_\pm(W)$ and which spike pairs are taken into consideration (all pairs or nearest one).
Spike-Timing Dependent Plasticity with an STDP function as the last equations can be implemented in an online update rule using the following assumptions. Each pre-synaptic spike arrival leaves a trace x_j which is updated by an amount

$A_+(W)$ at the moment of spike arrival and decays exponentially in the absence of spikes:

$$\frac{dx_j}{dt} = -\frac{x_j}{\tau_+} + \sum_f \delta(t - t_j^f)$$

The biophysical nature of the variable x need not be specified, but potential candidates are the amount of glutamate bound to postsynaptic receptors; or the fraction of NMDA receptors in the open state. Similarly, each post-synaptic spike leaves a trace y_i:

$$\frac{dy_i}{dt} = -\frac{y_i}{\tau_-} + \sum_f \delta(t - t_i^f)$$

which increases by an amount $A_-(W)$ at the moment of postsynaptic spikes. This trace could possibly be interpreted as the voltage at the synapse caused by a backpropagating action potential, or by calcium entry due to a backpropagating action potential.
Now The values of x and y determine the weight change:

$$\frac{dW_{ij}}{dt} = -A_-(W_{ij})y_i(t)\sum_f \delta(t - t_i^f) + A_+(Wij)x_j(t)\sum_f \delta(t - t_j^f)$$

Now in FlatSTDP, we don't use the parameter trace and instead we consider a constant number, but the effect of a trace is still there in the model. It means if a pre-synaptic neuron spikes and then a post synaptic neuron spikes, the trace of the pre-synaptic one is still in the process of training but instead of the actual value of the trace, a constant number multiply to the change of weights.

3.1 Results

Now we construct a LIF model with two layers. The first layer has ten neurons and it gets the two different patterns generated with poisson distribution. The second layer has two neurons which they learn the patterns using the FlatSTDP rule. The parameters using for this learning rule are:

- $A_+ = 0.310$, we made a soft bound for this part which you can see in the code.
- $A_- = 0.301$, we also made a soft bound for this part which you can see in the code.
- $cte_{pre} = 0.007$. This parameter is replaced instead of the trace of pre-synaptic neuron. This part is related to strengthening the synaptic weights.
- $cte_{post} = 0.0065$. This parameter is replaced instead of the trace of post-synaptic neuron. This part is related to weakening the synaptic weights.

Now Let's see the parameters of the LIF model:

- τ: 10
- R: 5
- θ: -37
- u_{rest}: -67

So we start training the neurons in the second layer through 2000 iterations. This is the raster plot of the neurons of two layers:

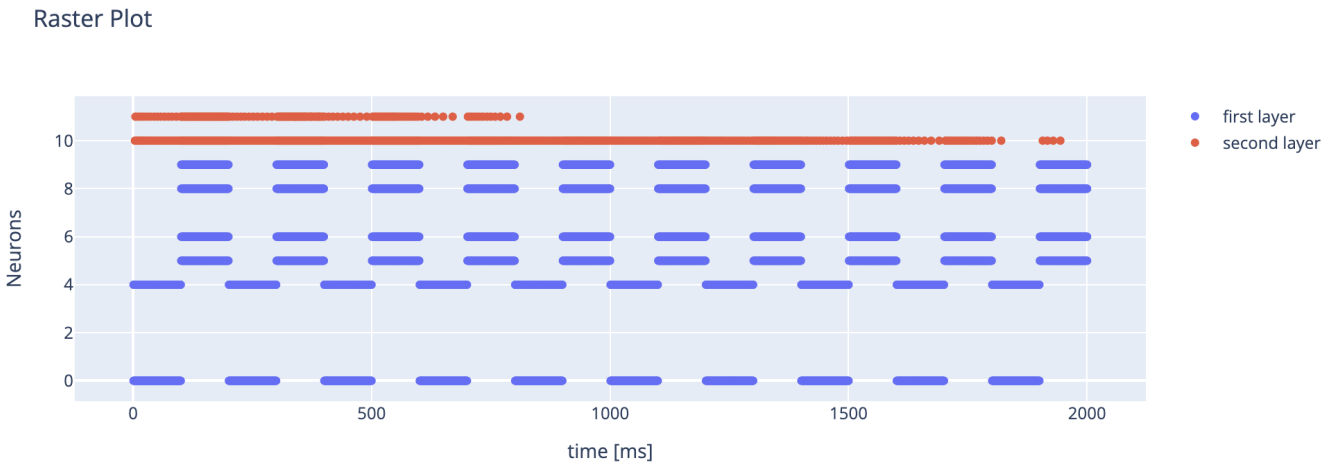
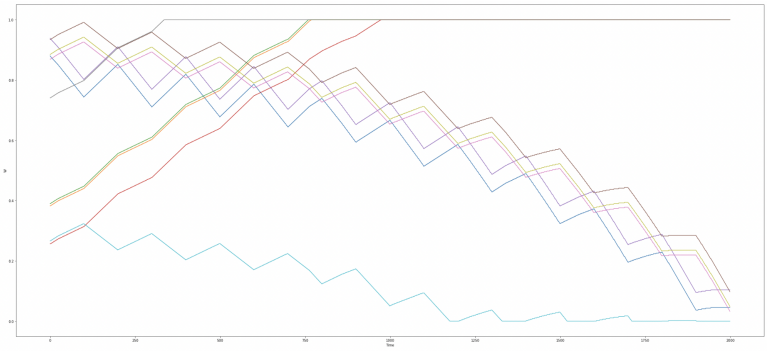
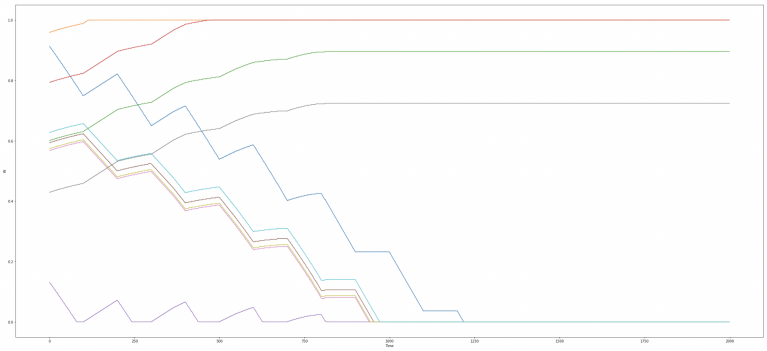


Figure 2: Raster Plot

As you can see we have two different patterns. In pattern number 1, the last five neurons don't spike and in the second pattern the first five neurons don't spike neither. Also you can see changes in the activity of the second layer which is came from the synaptic change (STDP learning rule). Actually some weights in both neuron got stronger and some weights got weaker. You can see the changes of the weights in the plot below:



(a) change of weights in the first neuron of the second layer



(b) change of weights in the second neuron of the second layer

Figure 3: weight changes

There are ten lines in each plot which indicates the change of weights during

the simulation. As expected for the first neuron, some weights are losing their strength and the other are getting stronger. We expected this to happen, so we can conclude that neurons learned something because some weights converged to 1 (got stronger) and the other converged to 0 (got weaker). This process happened in the second neuron too (figure 3 part(b)). Now let's see the changes of the weights in an another shape:

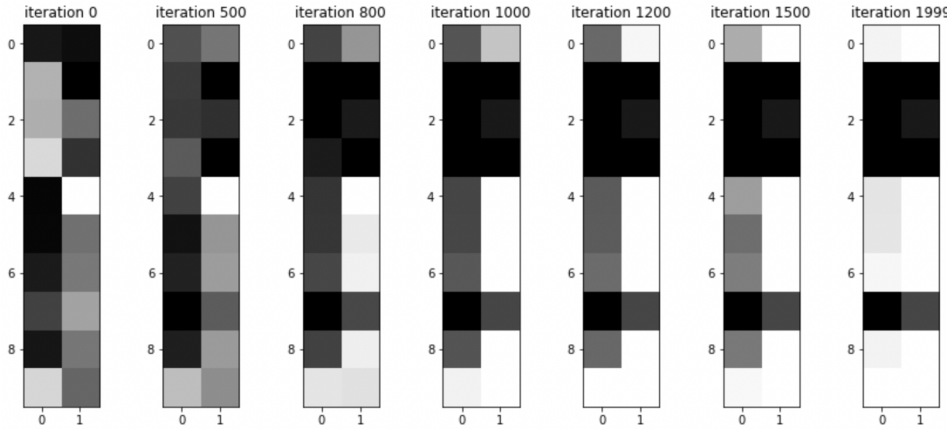


Figure 4: weight changes

This is the representation of synaptic change for this simulation. we chose the weights in some selected iterations (times). The white part shows the weights are getting weaker and converging to zero and the darker pixels shows that the weights are getting stronger and converging to one. The first column indicates the first neuron and the second column indicates the second one. As you see both neurons are learning one thing because the weights of same connections in the model are getting stringer and weaker and we don't want this. So we can use the RSTDP learning rule to prevent this to happening. The one thing that we can expected from STDP to do for us is to extract frequent features.

4 RSTDP

Reward-modulated Spike-Timing-Dependent Plasticity (R-STDP) is a learning method for Spiking Neural Network (SNN) that makes use of an external learning signal to modulate the synaptic plasticity produced by Spike-Timing-Dependent Plasticity (STDP). Combining the advantages of reinforcement learning and the biological plausibility of STDP, we can learn features by our selective neurons in the output layer.
Here we do this by the Dopamine neuro-modulator and a reward/punishment function. Dopamine is a neuro-modulator, affecting synaptic plasticity (LTP/LTD). STDP involves both LTP and LTD of synapses which we saw in the last section. Now we modulate by the function DA. DA can reinforce firing patterns even when they are followed by reward that is delayed by seconds. It means we can say which neuron learn which pattern. In this project for example we can train the output layer in a way that the first neuron of the output layer learns the first pattern and the second neuron learns the second pattern. This property relies on the existence of slow synaptic processes that act as synaptic eligibility

traces or synaptic tags.
The state of each synapse is represented by two variables:

- synaptic strength/weight, s
- activation of an enzyme important for plasticity, c (relatively slow process acting as a synaptic tag)

The dynamics of these variables follow:

$$\frac{dc}{dt} = -\frac{c}{\tau_c} + STDP(t)\delta(t - t_{pre/post})$$

$$\frac{ds}{dt} = cd$$

Here d describes the extracellular concentration of DA and $\sigma(t)$ is the delta function which shows the firing times of the pre-synaptic or the post-synaptic neurons. The variable d describes the concentration of extracellular DA:

$$\frac{dd}{dt} = -\frac{d}{\tau_d} + DA(t)$$

DA(t) models the source of DA due to the activity of dopaminergic neurons. Also τ_d is the time constant of DA.
So in this project we can select a neuron to learn the specific pattern (can be used in the classification tasks). For example we say that our first neuron of the output layer of the neuron should learn the first pattern. We can do this by regulating the reward function in a way that if the first pattern occurs and our first neuron spikes, the dopamine of the network will increase and the synaptic strength will increase too by the value of cd. Here you can see an image describing the reward modulated STDP which we talked about:

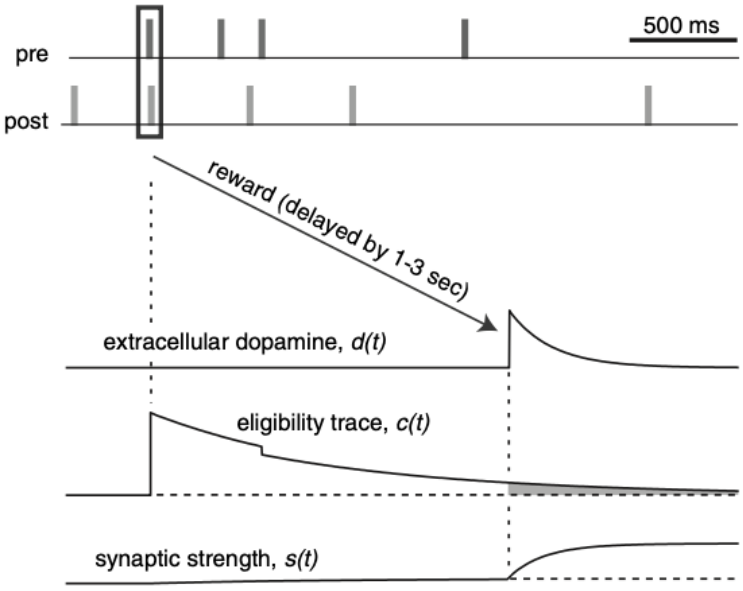


Figure 5: RSTDP

Now we see the results in our project:

4.1 Results

The structure of the model and the parameters of the model is remained the same. The number of iterations for this part is 15000. Also we have two patterns generated from poisson distribution. The result of the dopamine trace is in the image below:

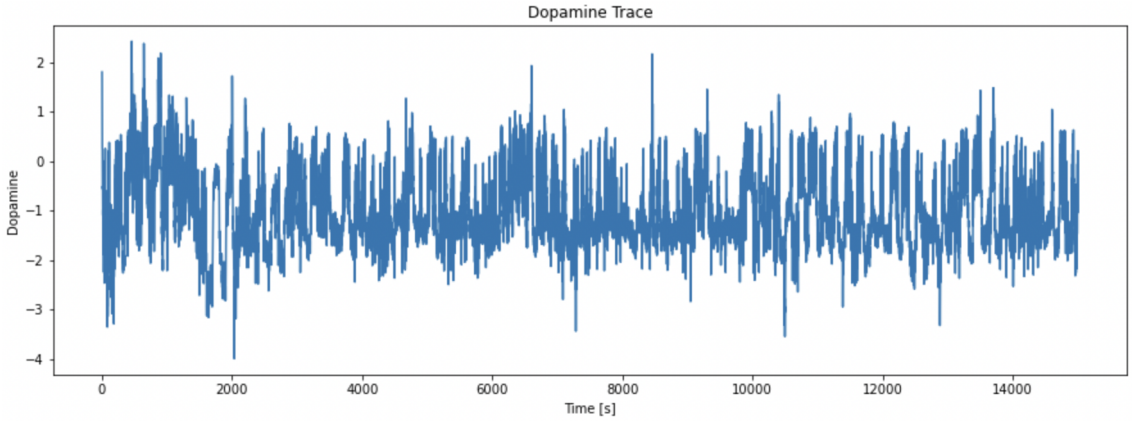


Figure 6: Dopamine Trace

As you see we always have changes in the dopamine (has reward and punishment) which is not good, because the frequency of changes should get lower.

5 References

[1] Biological neuron model https://en.wikipedia.org/wiki/Biological_neuron_model

[2] Biological neuron model https://en.wikipedia.org/wiki/Spike-timing-dependent_plasticity

[3] LIF Neuron Model https://maktabkhooneh.org/course/%D8%A2%D9%85%D9%88%D8%B2%D8%B4-%D8%B1%D8%A7%DB%8C%DA%AF%D8%A7%D9%86-%D8%B9%D9%84%D9%88%D9%85-%D8%A7%D8%B9%D8%B5%D8%A7%D8%A8-%D9%85%D8%AD%D8%A7%D8%B3%D8%A8%D8%A7%D8%AA%DB%8C-mk744/%D9%81%D8%B5%D9%84-%D8%A7%D9%88%D9%84-%D8%B9%D9%84%D9%88%D9%85-%D8%A7%D8%B9%D8%B5%D8%A7%D8%A8-%D9%85%D8%AD%D8%A7%D8%B3%D8%A8%D8%A7%D8%AA%DB%8C-ch2077/%D9%88%DB%8C%D8%AF%DB%8C%D9%88-%D9%85%D8%AF%D9%84-%D9%86%D9%88%D8%B1%D9%88%D9%86%DB%8C-lif/

[4] Unsupervised Learning https://maktabkhooneh.org/course/%D8%A2%D9%85%D9%88%D8%B2%D8%B4-%D8%B1%D8%A7%DB%8C%DA%AF%D8%A7%D9%86-%D8%B9%D9%84%D9%88%D9%85-%D8%A7%D8%B9%D8%B5%D8%A7%D8%A8-%D9%85%D8%AD%D8%A7%D8%B3%D8%A8%D8%A7%D8%AA%DB%8C-mk744/%D9%81%D8%B5%D9%84-%D8%A7%D9%88%D9%84-%D8%B9%D9%84%D9%88%D9%85-%D8%A7%D8%B9%D8%B5%D8%A7%D8%A8-%D9%85%D8%AD%D8%A7%D8%B3%D8%A8%D8%A7%D8%AA%DB%8C-ch2077/%D9%88%DB%8C%D8%AF%DB%8C%D9%88-%DB%8C%D8%A7%D8%AF%DA%AF%DB%8C%D8%B1%DB%8C-%D8%A8%D8%AF%D9%88%D9%86-%D9%86%D8%A7%D8%B8%D8%B1-%D9%82%D9%88%D8%A7%D9%86%DB%8C%D9%86-%D9%85%D8%B1%D8%A8%D9%88%D8%B7-%D8%A2%D9%86/

[5] ReInforcement Learning https://maktabkhooneh.org/course/%D8%A2%D9%85%D9%88%D8%B2%D8%B4-%D8%B1%D8%A7%DB%8C%DA%AF%D8%A7%D9%86-%D8%B9%D9%84%D9%88%D9%85-%D8%A7%D8%B9%D8%B5%D8%A7%D8%A8-%D9%85%D8%AD%D8%A7%D8%B3%D8%A8%D8%A7%D8%AA%DB%8C-mk744/%D9%81%D8%B5%D9%84-%D8%A7%D9%88%D9%84-%D8%B9%D9%84%D9%88%D9%85-%D8%A7%D8%B9%D8%B5%D8%A7%D8%A8-%D9%85%D8%AD%D8%A7%D8%B3%D8%A8%D8%A7%D8%AA%DB%8C-ch2077/%D9%88%DB%8C%D8%AF%DB%8C%D9%88-%DB%8C%D8%A7%D8%AF%DA%AF%DB%8C%D8%B1%DB%8C-%D8%AA%D9%82%D9%88%DB%8C%D8%AA%DB%8C-%D9%82%D9%88%D8%A7%D9%86%DB%8C%D9%86-%D9%85%D8%B1%D8%A8%D9%88%D8%B7-%D8%A2%D9%86/