Project 03

STDP LEARNING RULE

Hosein Khodadi | Computational Neuro-Science | 1400/2/31
95222028

STDP Unsupervised Learning Rule

In 1st exercise of this project we were supposed to implement STDP unsupervised learning rule in python and plot weight differences per various values of time differences. And here's the results:

- Parameters:

  A+ = 5 , A- = -5 , tau+ = 2 , tau- = 2 , initial weight = 5

  R = 0.1 , C = 0.4 , threshold = -55 , rest potential = -70

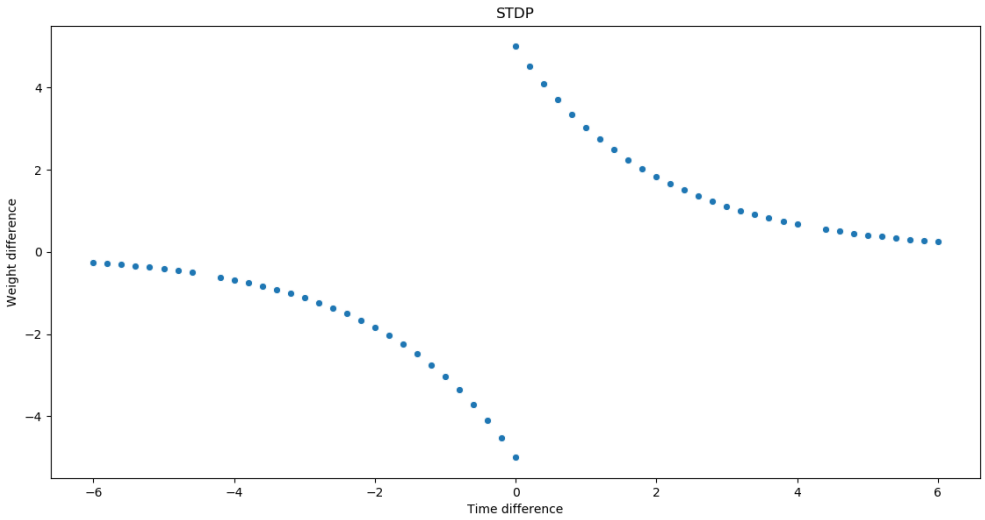

Figure 1

- Parameters:

A+ = 5 , A- = -5 , tau+ = 1 , tau- = 4 , initial weight = 5

R = 0.1 , C = 0.4 , threshold = -55 , rest potential = -70

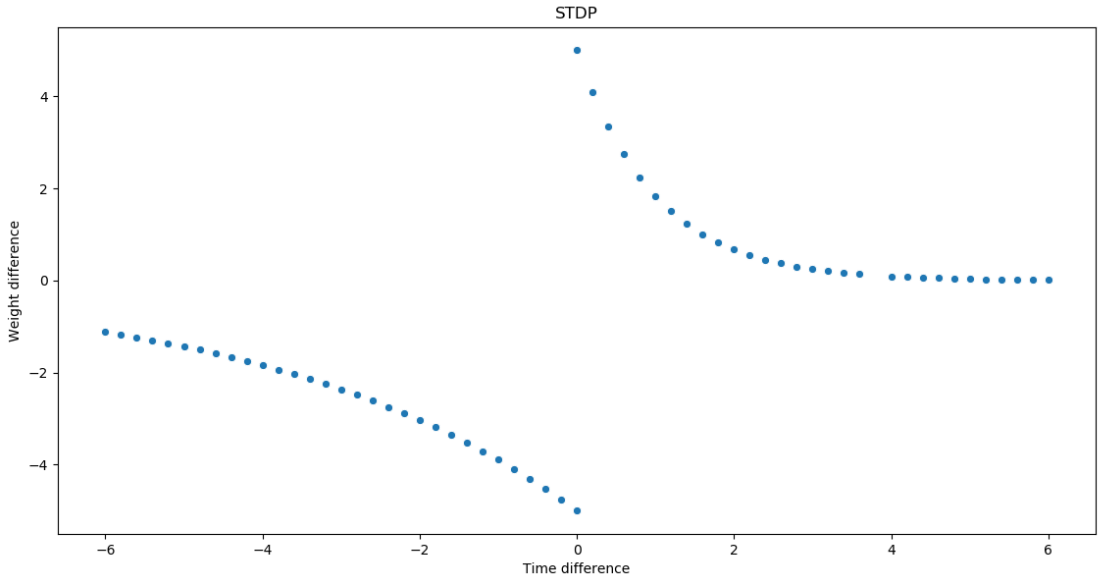

Figure 2

- Parameters:

A+ = 2 , A- = -4 , tau+ = 1 , tau- = 1 , initial weight = 1

R = 0.1 , C = 0.4 , threshold = -55 , rest potential = -70

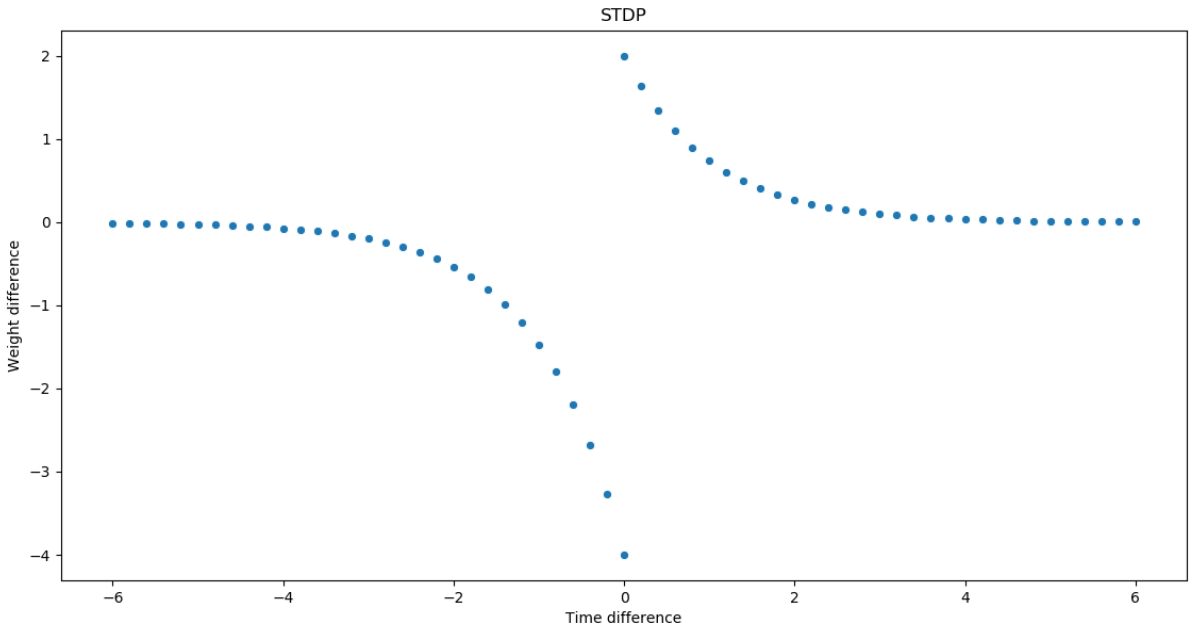

Figure 3.

As we can see in the above plots, the A+ and A- parameters determine the maximum and minimum value of the weight differences that actually happens when time difference is about zero (respectively).(It's obvious according to formulas too!)

And the tau parameter actually influences the slope of the curves. As the value of this parameter increases, the curves will be smoother and less sloping. (see Figure3)

SPIKING NEURAL NETWORK

In 2nd exercise of this project we were supposed to make a spiking neural network with excitatory neurons which has 2 layers, input and output. The input layer has 10 neurons and the output layer has 2 neurons.

We have to pass two different input patterns in specified distances. And we expect each of the output neurons learn one of the input patterns.

For this purpose I used (normal) Gaussian distribution for random initial weights.

- Parameters :

Time → T = 10000, dt = 0.1 , between pattern distance = 2

Weights → mu = 1, sigma = 0.25

Initial random weights =

```
[[1.14593345  1.44421825]
 [1.19003622  1.47681242]
 [1.1187294   0.71126059]
 [0.57795523  0.71400719]
 [1.57771627  1.09690716]
 [0.73734623  1.39396207]
 [0.80752106  0.74859935]
 [0.94842908  0.9291287 ]
 [0.79924486  1.1727716 ]
 [0.47380368  1.15905091]]
```

Final weights =

```
[[88.6826518  88.9810369]
 [65.5150078  64.6858848]
 [87.2760545  86.9706605]
 [64.9029268  63.9230795]
 [89.1144346  88.6337258]
 [87.1041536  87.7608442]
 [88.4009395  88.3421182]
 [87.3152365  87.2960108]
 [88.3926633  88.7662905]
 [64.7887192  63.6480674]]
```

Patterns → pattern1 = [0, 0, 0, 0, 0, 3, 1, 3, 1, 3] pattern2 = [1, 2, 3, 2, 1, 0, 0, 0, 0, 0]

Neuron → R = 0.4 C = 0.1 threshold = -55 u_rest = -70 alpha = 0.8

Learning → A+ = 0.001 A- = 0.001 tau+ = 1000 tau- = 1000

- Parameters :

Time → T = 10000, dt = 0.1 , between pattern distance = 2

Weights → mu = 1, sigma = 0.25

Initial random weights =

```
[[0.22909422  0.88247571]
 [1.58008655  1.25181331]
 [0.75150489  0.928775  ]
 [0.9734172   0.55462373]
 [0.70673328  1.08774961]
 [1.233864    0.93135555]
 [0.83148271  1.07267115]
 [1.214666    1.35181282]
 [1.01935229  0.79159291]]
```

Final weights =

```
[[87.82251578  88.47595762]
 [65.93100567  64.48479285]
 [88.34492645  88.52225691]
 [65.32433632  63.78760328]
 [88.30015484  88.68123152]
 [88.77058548  88.4681373 ]
 [65.15245792  64.27770661]
 [88.75138748  88.88859457]
 [65.07685328  64.19762952]
 [88.55607377  88.32837466]]
```

Patterns → pattern1 = [1, 2, 1, 2, 1, 0, 0, 0, 0, 0] pattern2 = [0, 0, 0, 0, 0, 1, 2, 1, 2, 1]

Neuron → R = 0.4 C = 0.1 threshold = -55 u_rest = -70 alpha = 0.7

Learning → A+ = 0.001 A- = 0.01 tau+ = 1000 tau- = 100

- Parameters :

Time → T = 10000, dt = 0.1 , between pattern distance = 2

Weights → mu = 1, sigma = 0.25

Initial random weights =

```
[[0.67605296 0.61464157]
 [1.11226719 0.72526769]
 [0.55116799 1.12035966]
 [1.10781065 1.39715989]
 [1.20964698 1.06685188]
 [0.77663751 0.65424702]
 [0.96428846 1.56898547]
 [0.93634102 1.32285098]
 [0.76623838 0.80457598]
 [1.15427243 1.04782725]
```

Final weights =

```
[[88.26944477 88.20805254]
 [65.4664232  64.02439279]
 [86.91795376 87.4871621 ]
 [87.47459642 87.76396233]
 [65.56380299 64.36597697]
 [88.37002932 88.24765799]
 [65.29850061 64.84816668]
 [88.47303275 88.85956186]
 [65.10045053 64.08375718]
 [88.69096416 88.58453813]]
```

Patterns → pattern1 = [1, 2, 1, 1, 2, 1,0, 0, 0, 0] pattern2 = [0, 0, 0, 0, 0, 0, 2, 1, 2, 1]

Neuron → R = 0.4 C = 0.1 threshold = -55 u_rest = -70 alpha = 0.7

Learning → A+ = 0.001 A- = 0.01 tau+ = 1000 tau- = 100

Since it's an unsupervised learning method it seems that this doesn't work properly well and it usually learns only one pattern (first pattern) or both patterns at the same time. But it's learning something at least! :D