# گزارش پروژه سری 3

در بخش اول پروژه، باید الگوریتم STDP را روی یک شبکه عصبی اسپایکی با نورون های شبیه سازی شده توسط مدل LIF پیاده سازی کنیم.

یک کلاس شمارنده NeuronClass برای مشخص کردن نوع نورون (مهاری یا تحریکی) در نظر می گیریم.

کلاس Simple LIF که از پروژه اول استفاده شده است، شامل شبیه سازی مدل نورونی LIF است.
کلاس SimpleLIFWithSingleStep همان مدل لیف، ولی با تابع جریان ورودی Single Step است.

کلاس NeuronsGroup، برای گروه کردن نورون ها با توجه به مهاری یا تحریکی بودن آن ها است.

تابع potential_plot برای رسم نمودار پتانسیل زمان به کار می رود.

حال، در کلاس SpikingTimeDependantPlasticity، مدل STDP خود را پیاده سازی می کنیم. تابع train_the_network، وظیفه آموزش شبکه (آپدیت کردن وزن ها) را بر عهده دارد. تابع weight_plot، نمودار تغییرات وزن در زمان بین دو نورون را بررسی می کند.

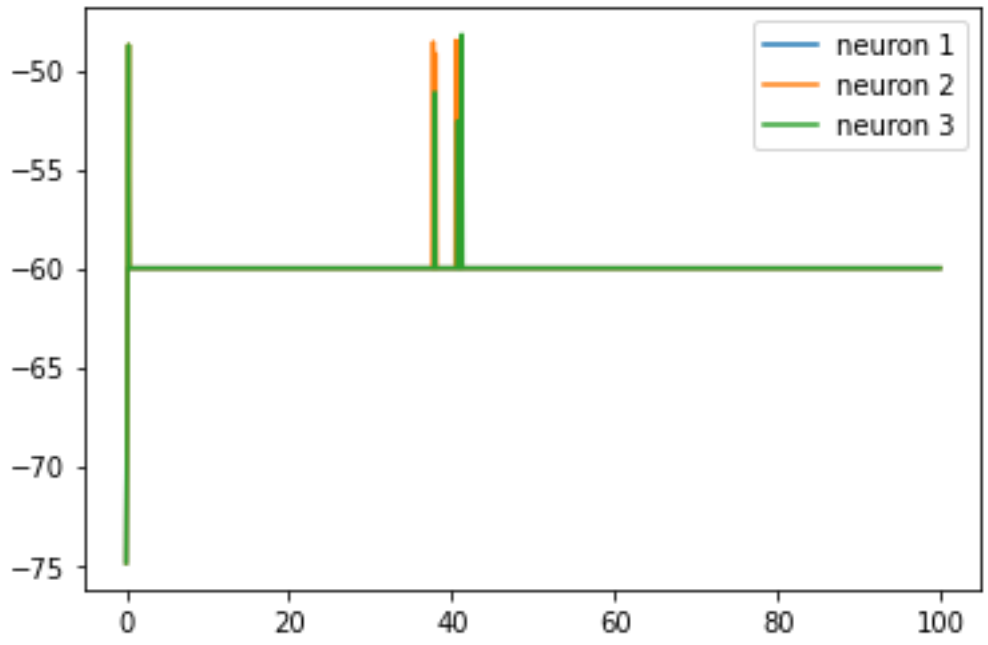
حال بیایید با استفاده از توابع رسم نمودار ها، نتایج به دست آمده را بررسی کنیم.

وزن های اولیه بین نورون ها به صورت زیر تعریف شده است:

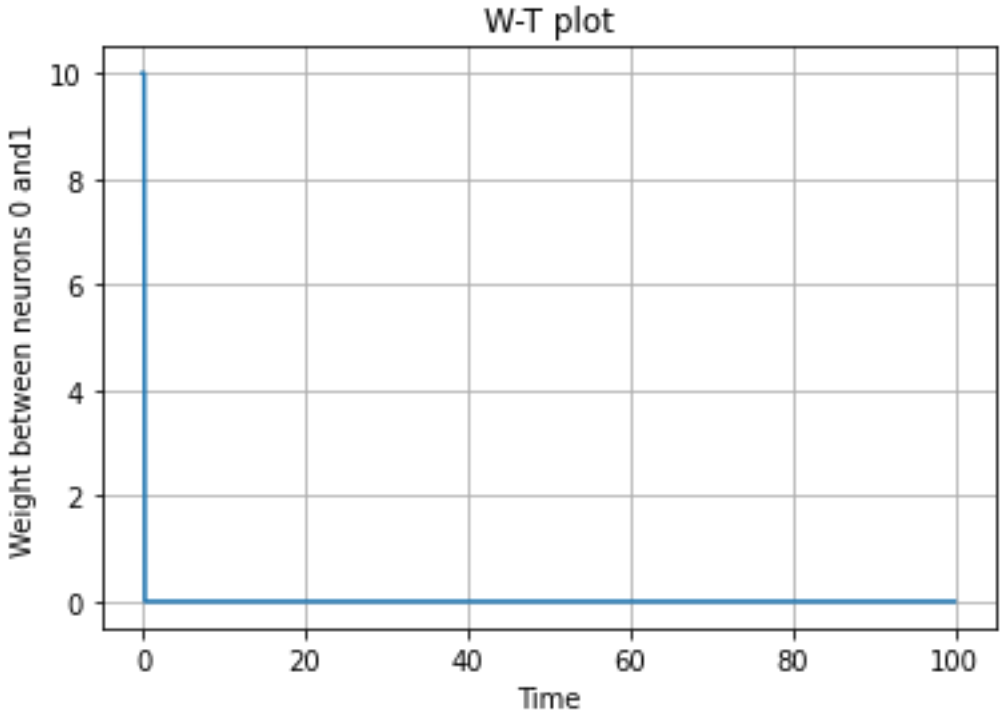
```
[[0, 10, 9], [14, 0, 12], [13, 11, 0]]
```

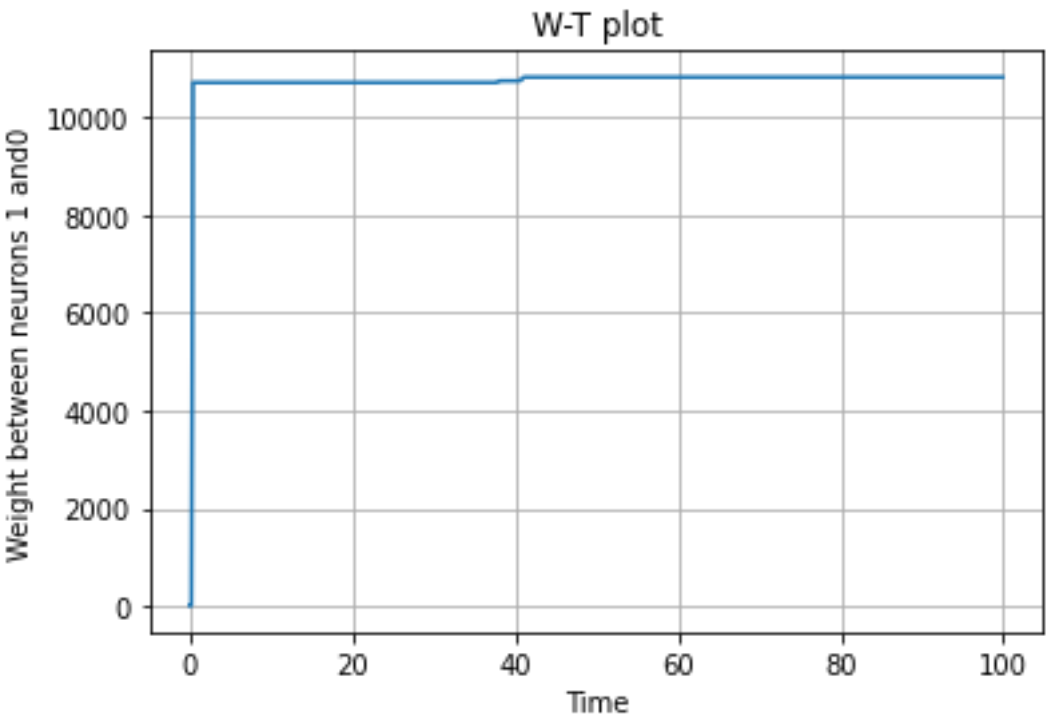
عضو اول این آرایه نشان دهنده این است که نورون اول با خودش وزن 0، با نورون دوم وزن 10 و با نورون سوم وزن 9 را دارد.

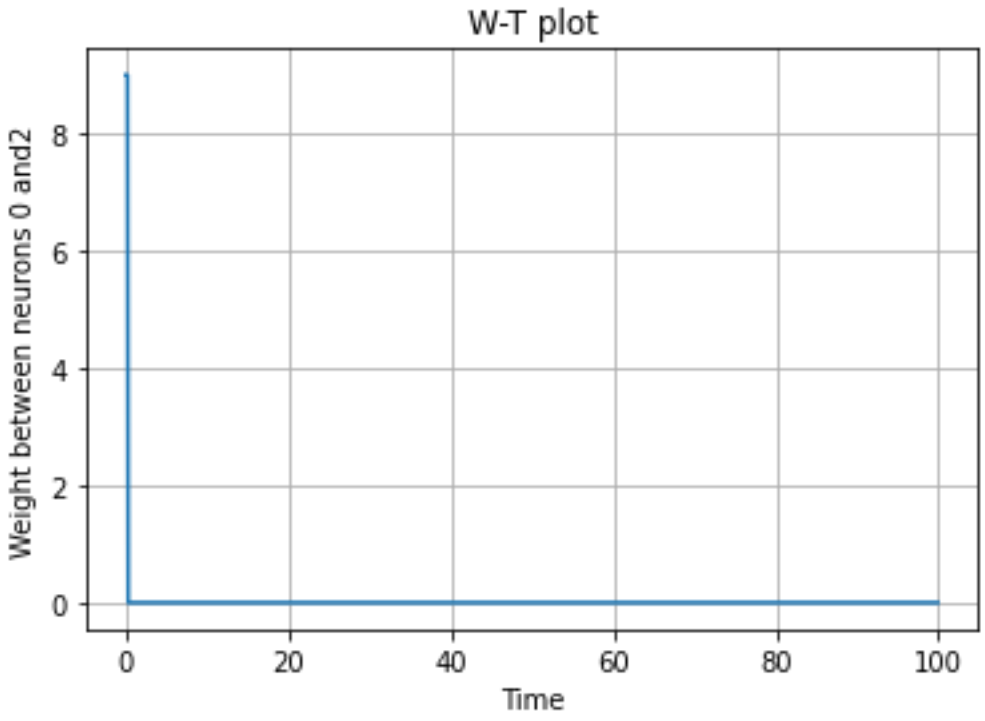
نمودار تغییرات پتانسیل نسبت به زمان (اسپایک های هر سه نورون) به صورت زیر است:

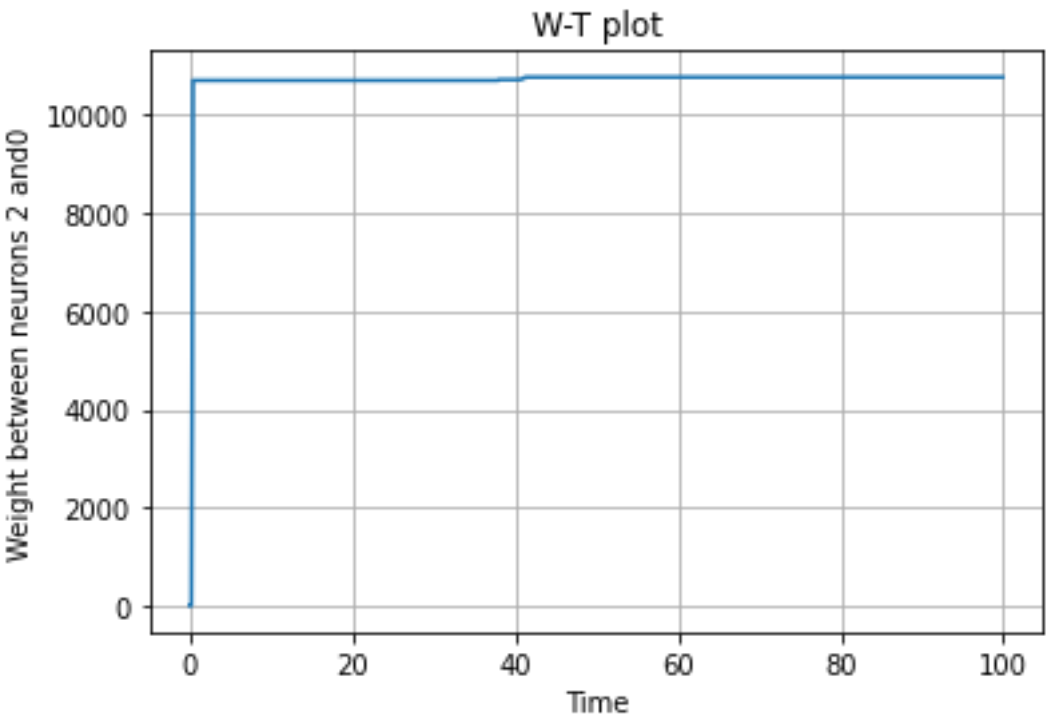


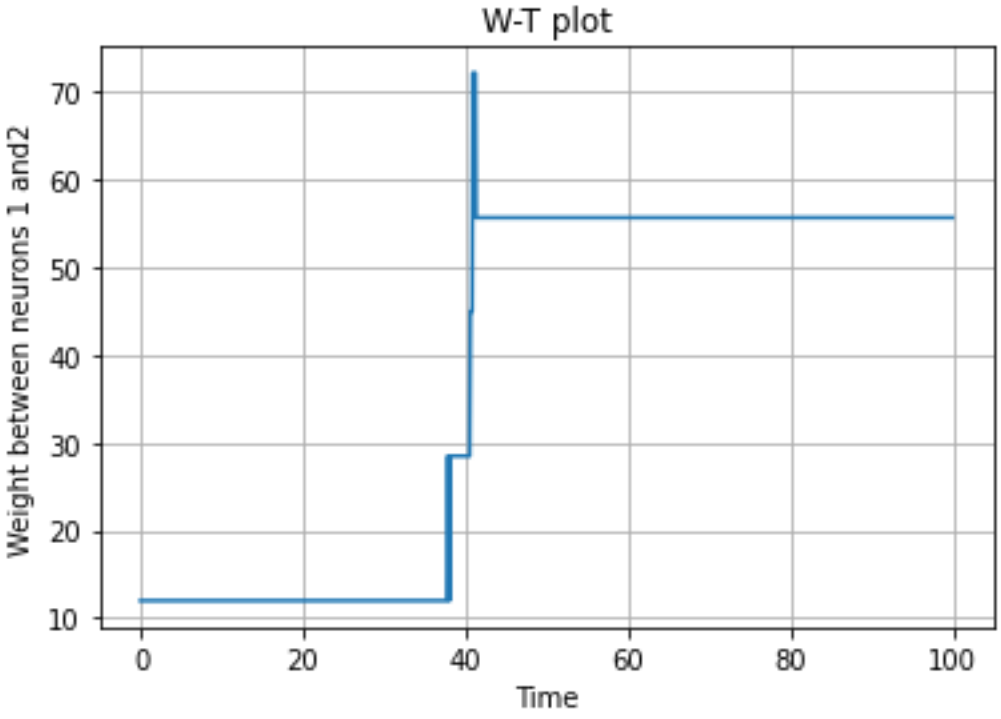

نمودارهای بعدی، تغییران وزن های بین نورون ها در طول زمان، بین هر جفت نورون ها (خاصیت جابجایی ندارد) را بررسی می کند.

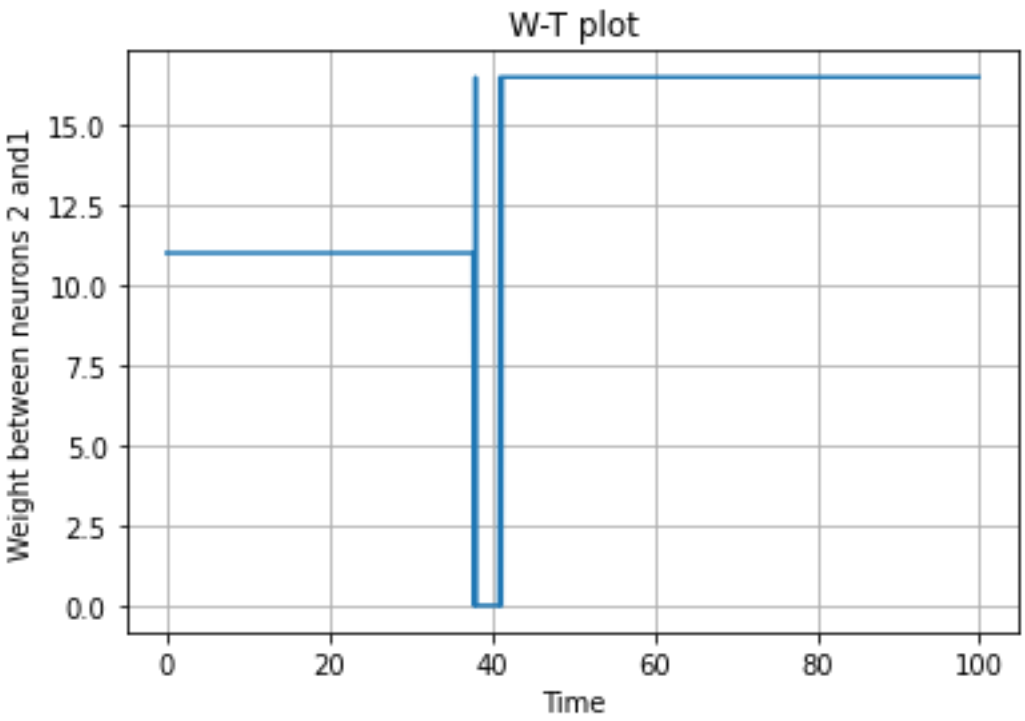



W-T plot
Weight between neurons 1 and0
10000
8000
6000
4000
2000
0
0
20
40
60
80
100
Time


W-T plot
Weight between neurons 0 and2
8
6
4
2
0
0
20
40
60
80
100
Time


W-T plot
Weight between neurons 2 and0
10000
8000
6000
4000
2000
0
0
20
40
60
80
100
Time


W-T plot
Weight between neurons 1 and2
70
60
50
40
30
20
10
0
20
40
60
80
100
Time




مشاهده می شود که با اسپایک نورون تحریکی پیشین قبل از نورون پسین، وزن بین این دو نورون افزایش می یابد.

حال برای بخش دوم پروژه، کلاس SpikingNeuralNetwork را تعریف می کنیم که شامل آرگومان های ثابت تاو، نورون ها، ابعاد شبکه، و میزان دوپامین و در نتیجه reward دریافتی است. تابع connect_the_neurons، نورون های ما را به هم متصل می کند. تابع fit با استفاده از تابع train_the_network، داده ها را به تعداد iteration یا epochها، به خورد شبکه عصبی ما می دهد. تابع activity_history سابقه و تاریخچه فعالیت های لایه ها را در خود ذخیره می کند و تابع test_the_model، برای پیشبینی داده های تست در نظر گرفته شده است.

تابع جریان ورودی به شبکه ما به صورت زیر است:

```
const = lambda x: 180 * (math.sqrt(math.fabs(math.sin(x))))
```

وزن های اولیه همان وزن های بالایی هستند.

داده ها را از فایل اکسل می خوانیم و در 4 دسته فیچرها و نتیجه های آموزش و تست ذخیره می کنیم.

داده ها را به مدل فیت کرده، و سپس تابع پیشبینی را روی داده های آموزشی و تست اجرا می کنیم.

دقت مدل روی داده های آموزشی 0.7 و روی داده های تست 0.8 می باشد.