Computation NeuroScience, Homework 3

장지원 202211167

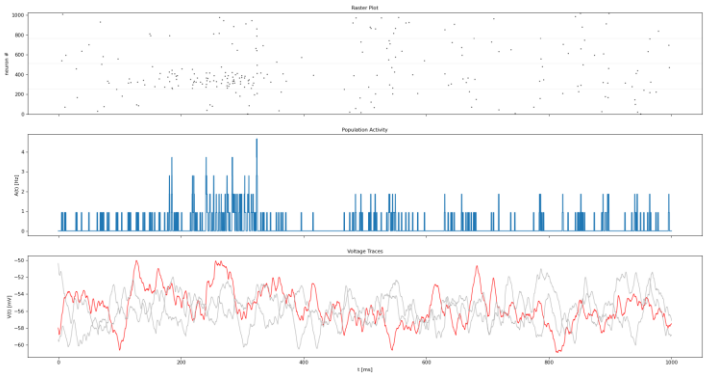
[Exercise 11.2]

Input stimulus 를 가해보자. `stimulus_strength, t_stimulus_duration, sim_time` 이용해서 input stimulus 를 조절할 수 있다. (이 때 `stimulus_center_deg=120, stimulus_width_deg=30` 으로 설정했다)

11.2.1.

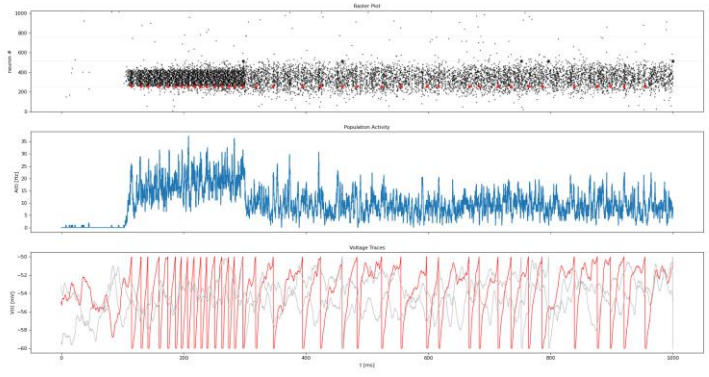
a. Population 변화를 볼 수 있는 time은 언제인가? Stimulus가 적용된 시간과 비교해 보면 어떤가?

: `stimulus_strength, t_stimulus_duration, sim_time`을 각각 0.06, 200, 100으로 설정해서 simulation을 진행해 보았다. 결과는 밑과 같다. 100ms~300ms 정도에서 Population activity가 변화하는 것(firing 된다)을 확인할 수 있다. 이는 Stimulus가 적용된 시간과 일치한다.



b. Simulation 끝 쪽의 population은 어떠한가?

: Simulation 끝 쪽에서도 population activity가 지속 되는 것을 확인할 수 있다. 단 위에서는 0.06nA 정도의 strength만 주었기 때문에 명확한 확인이 불가능하다. 0.3nA 정도의 strength를 준다면, 밑과 같은 결과가 나오게 된다. 여기서는 population activity가 유지된다는 것을 명확히 확인할 수 있다.

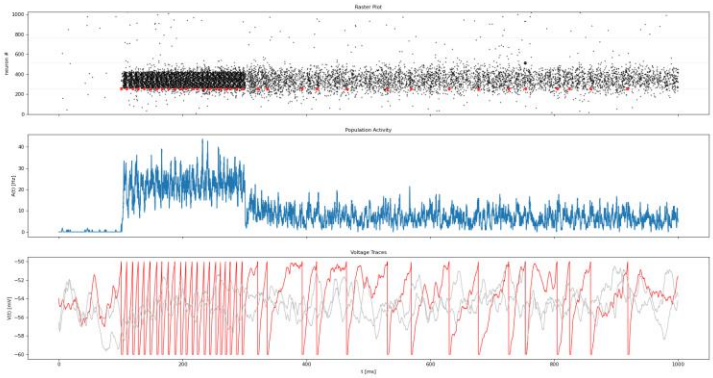


c. 400ms에서의 firing rate를 관찰해 보자.

: 위의 두 결과의 세 번째 plot에서 firing rate를 확인할 수 있다. 400ms에서는 stimulus가 있을 때 만큼 firing이 자주 되지는 않지만, 어느 정도 firing이 유지 되는 것을 확인할 수 있다. 마찬가지로 stimulus가 0.06nA일 때는 명확히 확인할 수 없으나, 0.3nA일 때는 이를 잘 확인할 수 있다.

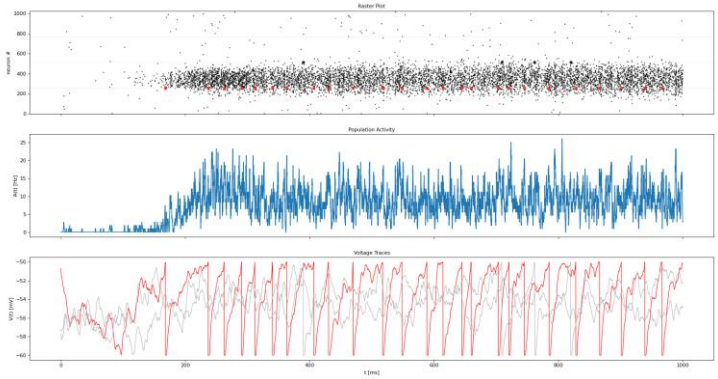
d. Simulus를 0.5nA로 주었을 때는 어떤가?

: 밑과 같은 결과가 나오게 된다. 0.3nA를 주었을 때와 마찬가지로 population activity가 지속되는 것을 명확히 확인할 수 있다.



e. `stimulus_width_deg=60, stimulus_strength=0.1 * b2.namp, stimulus_center_deg = 120` 와 같이 param을 설정했을 때 어떤 변화가 나타나는지 확인해 보자(width를 높여보자).

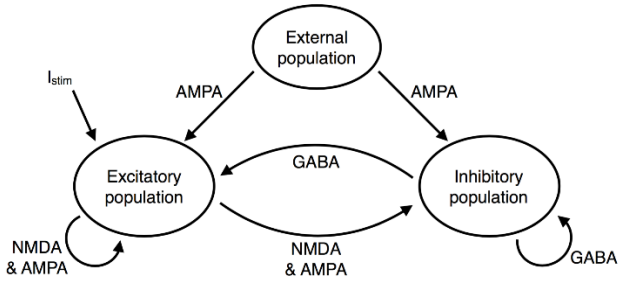
: 밑과 같은 결과가 나온다. width가 커지면 폭이 조금 더 넓어지는 것을 확인할 수 있다.



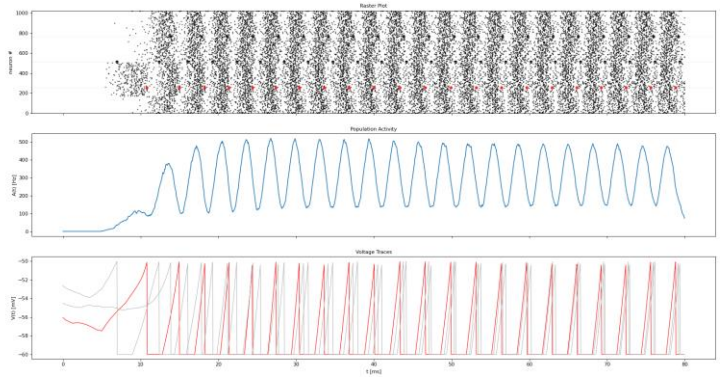
11.2.2.

inhibitory population을 줄여보자.

a. `N_inhibitory = 1, stimulus_strength=0.65 * b2.namp, t_stimulus_start=5 * b2.ms, t_stimulus_duration= 25 * b2.ms, sim_time=80. * b2` 로 파라미터를 설정했을 때 결과가 어떻게 나오게 될 지 예측해 보자. 시뮬레이션 결과는 어떤가?



: 우리는 위 그림에 따라서 inhititory가 적어지면, 더 firing을 많이 할 것임을 예측할 수 있다. 실제로도 밑과 같이 비슷한 결과가 나온다.



b. Inhibitory nueron은 어떤 역할을 하는가?

: Exhibition population, inhibition population에 대한 plot은 밑과 같다. Inhibitory neuron은 exhibitory neuron을 억제하여 네트워크가 균형을 이룰 수 있게 해준다.

[Exercise 12.3]

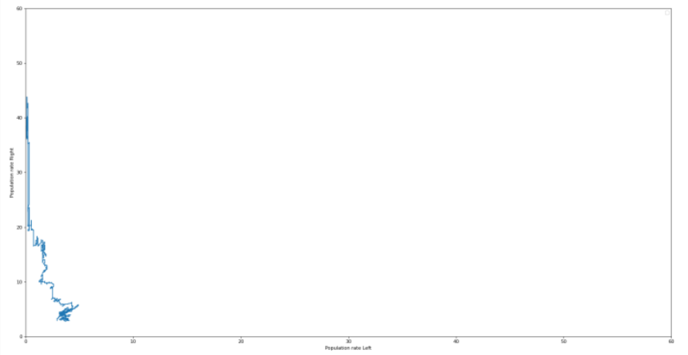
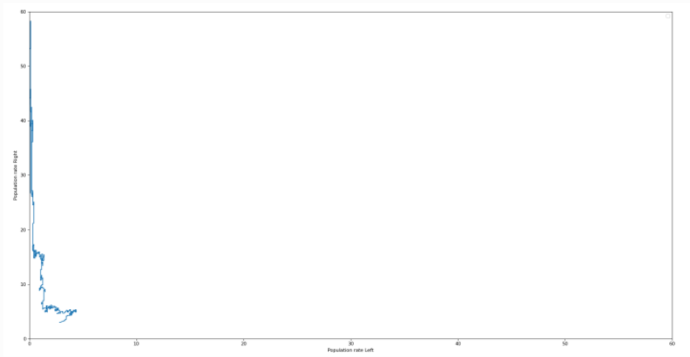
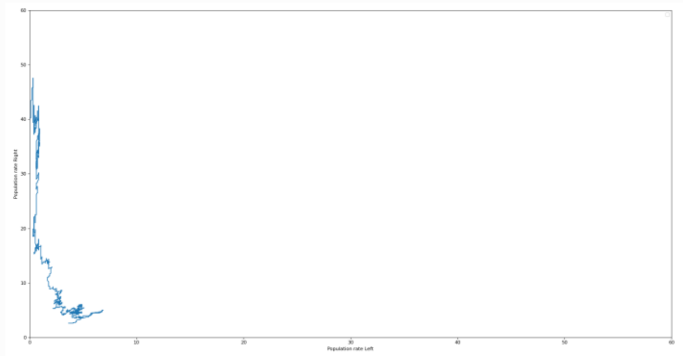
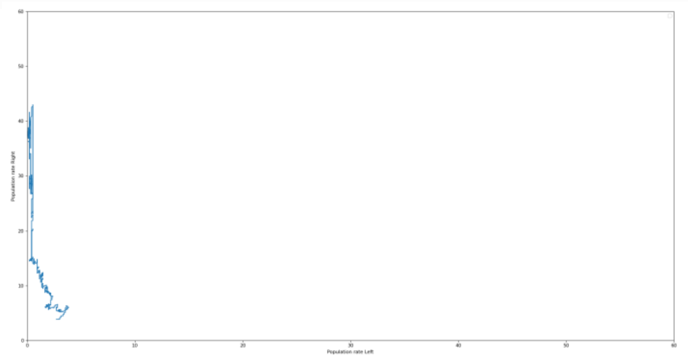
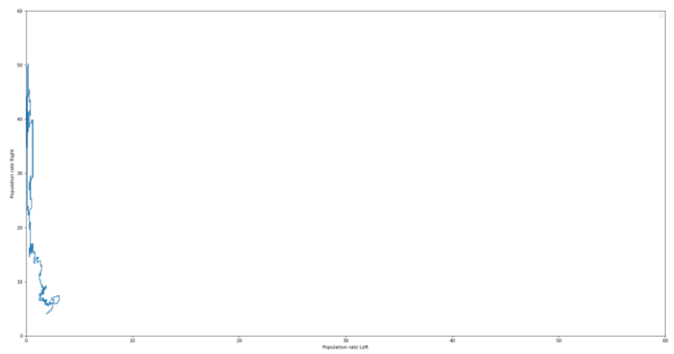
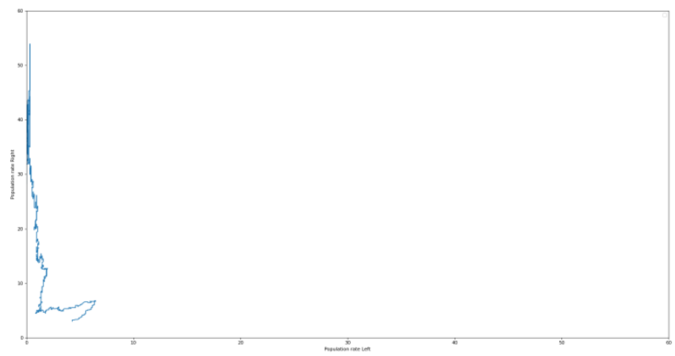
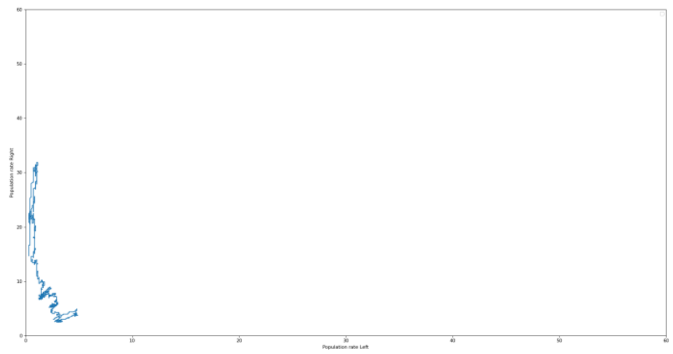
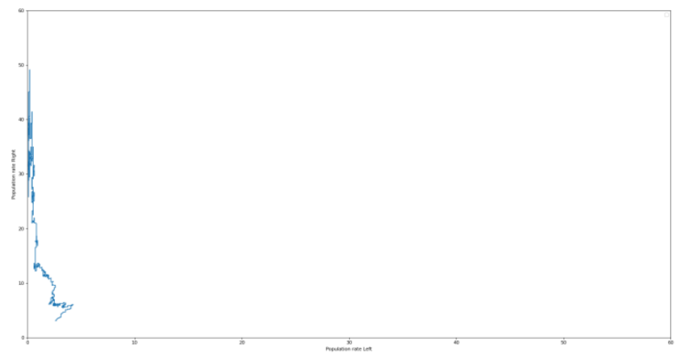
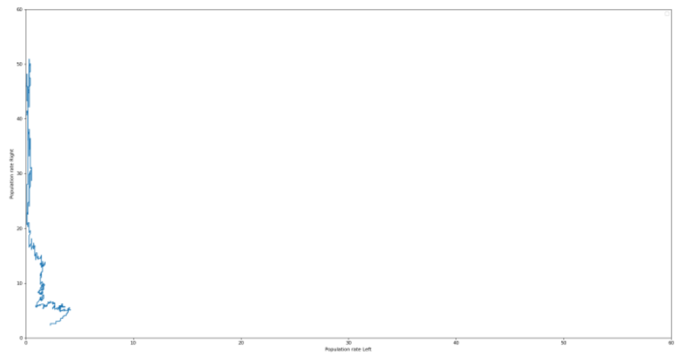
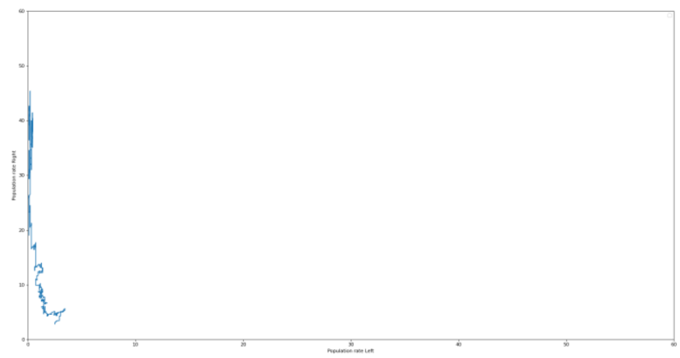
Decision space를 그려보자.

12.3.1

avg_window_width를 정하고, decision space를 바탕으로 이에 따른 threshold 값을 정해보자.

: avg window width는 100으로 정하였다. 이를 바탕으로 10개의 plot를 구해서 threshold를 구해보았다. 10개의 plot는 밑과 같이 나왔다. Plot을 보면 대체로 20정도를 넘어가면 확실히 결정을 하는 것을 확인할 수 있다. 따라서 threshold를 20으로 결정하였다.

(threshold를 10으로 설정하면 '잘못된 예측'을 많이하게 될 수 있다. 따라서 여유있게 threshold를 20정도로 설정하였다)


Population rate Right
Population rate Left

Population rate Right
Population rate Left

Population rate Right
Population rate Left

Population rate Right
Population rate Left

Population rate Right
Population rate Left

Population rate Right
Population rate Left

Population rate Right
Population rate Left

Population rate Right
Population rate Left

Population rate Right
Population rate Left

Population rate Right
Population rate Left

12.3.2

C=-0.1, 0.1, 0.3, 0.5에 대해서 decision making simulation을 10회씩 실행해 보고, 위에서 구한 기준에 따라서 percent correct와 average decision time을 구해보자.

: 기준은 위에서 20으로 정하였다. avg_window_width를 100으로 하고, 위에서 정한 20을 threshold로 하여서 시뮬레이션을 진행해 보았다.

먼저 C=-0.1일 때의 결과는 다음과 같다. 우리는 C가 음수 이므로 right로 가야함을 예측해 볼 수 있다. 또한 C 값이 작으므로 decision 시간도 상대적으로 느릴 것 임을 예측해 볼 수도 있다.

Period	Correct (Yes/No/None)	Decision time (s)
1	Y	0.499
2	Y	0.580
3	N	0.948
4	Y	0.352
5	Y	0.773
6	Y	0.853
7	None	0.000
8	Y	0.384
9	None	0.000
10	Y	0.820

이 때는 10번의 시도 중 7번 옳은 방향을 예측했고, 즉 대부분 옳은 방향을 찾는 것을 확인할 수 있었다. Decision time은 None(결정하지 못한 경우)일 때를 제외하고, 평균 0.651s정도가 나왔고, C값이 클 때(C=0.5)일 때보다 상대적으로 느림을 확인할 수 있다.

C=0.1일 때의 결과는 다음과 같다. 우리는 C가 양수 이므로 left로 가야함을 예측해 볼 수 있다. 또한 마찬가지로 C 값이 작으므로 decision 시간도 상대적으로 느릴 것 임을 예측해 볼 수도 있다.

Period	Correct (Yes/No/None)	Decision time (s)
1	Y	0.457
2	None	0.000
3	None	0.000
4	None	0.000
5	Y	0.569
6	N	0.748
7	N	0.491
8	N	0.921
9	N	0.466
10	Y	0.645

이 때는 10번의 시도 중 3번 옳은 방향을 예측했고, 옳은 방향을 잘 찾지 못하는 것을 확인할 수 있었다. 이는 C 값이 충분히 크지 않았기 때문일 것이다. Decision time은 None(결정하지 못한 경우)일 때를 제외하고, 평균 0.613s정도가 나왔고, C값이 클 때(C=0.5)일 때보다 상대적으로 느림을 확인할 수 있다.

C=0.3일 때의 결과는 다음과 같다. 우리는 C가 양수 이므로 left로 가야함을 예측해 볼 수 있다. 또한 C 값이 0.1과 0.5 사이에 있으므로 중간의 decision time을 가질 것임도 예측해 볼 수 있다.

Period	Correct (Yes/No/None)	Decision time (s)
1	Y	0.674
2	Y	0.569
3	Y	0.442
4	None	0.000
5	Y	0.441
6	None	0.000
7	None	0.000
8	None	0.000
9	None	0.000

10	None	0.000

이 때는 10번의 시도 중 4번 옳은 방향을 예측했고, 40%의 확률로 옳은 방향을 찾는 것을 확인할 수 있었다. 이 때는 잘못된 결정을 하지는 않지만, 결정을 하지 '못하는' 경우가 많았다. 이는 theshold를 높게 설정해 주었기 때문일 것이다. 우리는 threshold를 높게 줌으로써 잘못된 결정에 대한 오류를 피했지만, 결정을 하지 못하는 오류를 얻게 된 것이다. 이 둘은 trade off 관계인 것으로 예상된다. Decision time은 None(결정하지 못한 경우)일 때를 제외하고, 평균 0.531s정도가 나왔고, C값이 0.1일 때보다 상대적으로 빠름을 확인할 수 있다.

C=0.5일 때의 결과는 다음과 같다. 우리는 C가 양수 이므로 left로 가야함을 예측해 볼 수 있다. 또한 C 값이 크므로 상대적으로 빠른 decision time을 가질 것임도 예측해 볼 수도 있다.

Period	Correct (Yes/No/None)	Decision time (s)
1	Y	0.496
2	None	0.000
3	None	0.000
4	Y	0.527
5	Y	0.594
6	Y	0.467
7	Y	0.518
8	None	0.000
9	Y	0.476
10	Y	0.448

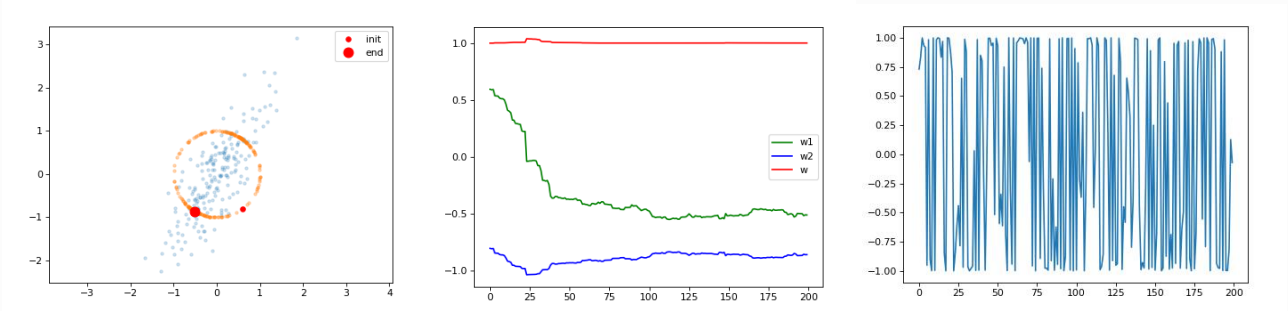
이 때는 10번의 시도 중 7번 옳은 방향을 예측했고, 대부분 옳은 방향을 찾는 것을 확인할 수 있었다. Decision time은 None(결정하지 못한 경우)일 때를 제외하고, 평균 0.499s정도가 나왔다. C=0.1, C=0.3일 때에 비해서 빠른 예측을 하는 것을 확인할 수 있었다.

[Exercise 9.3]

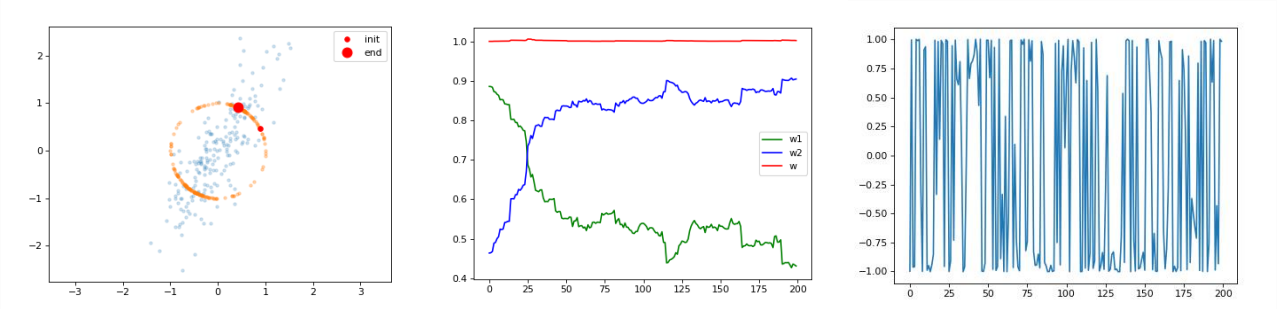
ratio=0.3, 0.5, 1 일 때 문제를 풀어보자.

a. 세 개의 ratio 에 대해서 plotting 해보자. 이 때 eta 는 0.04 로 설정하였다.

ratio = 0.3.



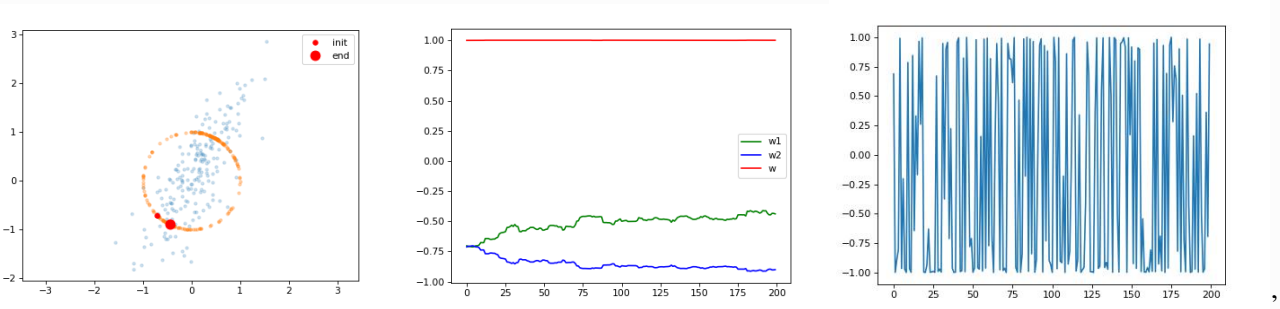

```
The final weight vector w is: (-0.511613704890855,-0.8602938595696323)

Max y =  1.000909047814851

Min y =  -1.0009243995156778
```



The final weight vector w is: (0.43106257813852766,0.9046536430567025)

Max y = 1.0020992720000552

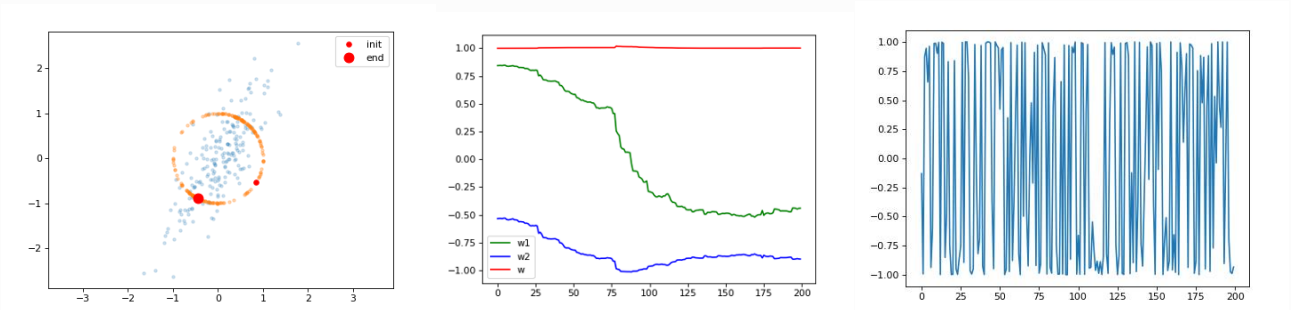
Min y = -1.00209996382079



The final weight vector w is: (-0.43769512140088984,-0.9002445494333495)

Max y = 1.0010060637295783

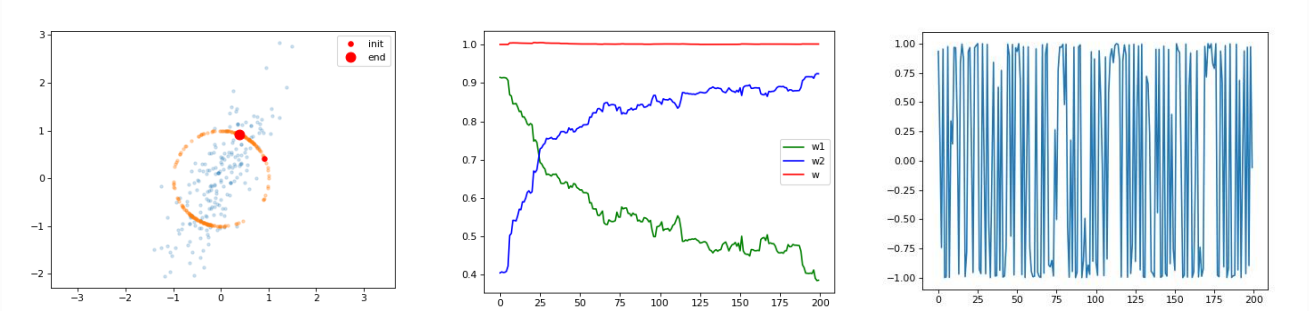
Min y = -1.0009911106069267



The final weight vector w is: (-0.44075577588042236,-0.8994302901139081)

Max y = 1.0016189324955649

Min y = -1.0016069558568863

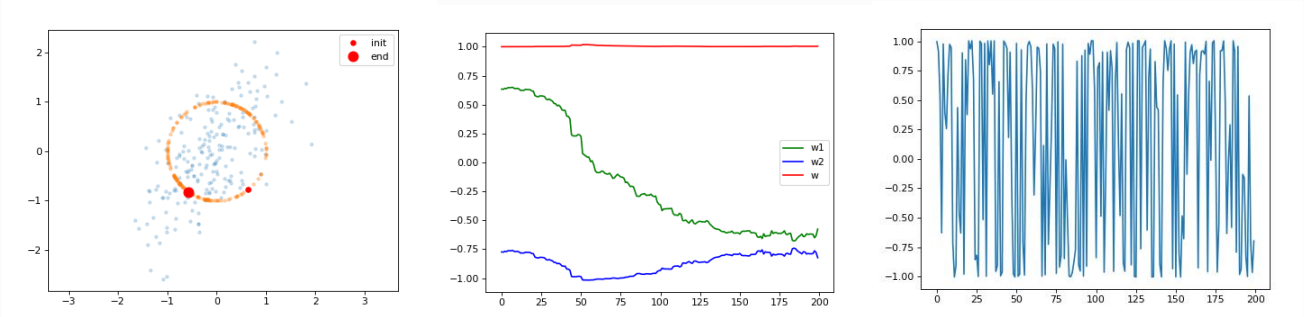


The final weight vector w is: (0.3859594954473071,0.9237800541405595)

Max y = 1.0011599424513324

Min y = -1.0011430666110803

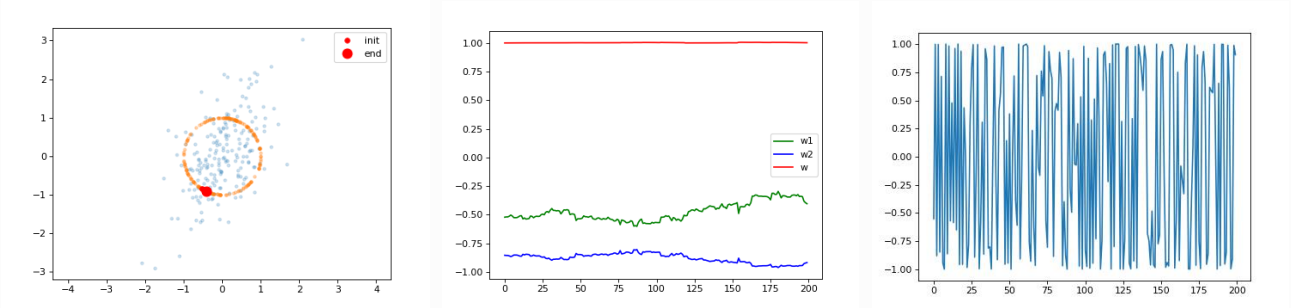
ratio = 0.5.



The final weight vector w is: (-0.5760072435609349,-0.8231379516750618)

Max y = 1.0046255791776493

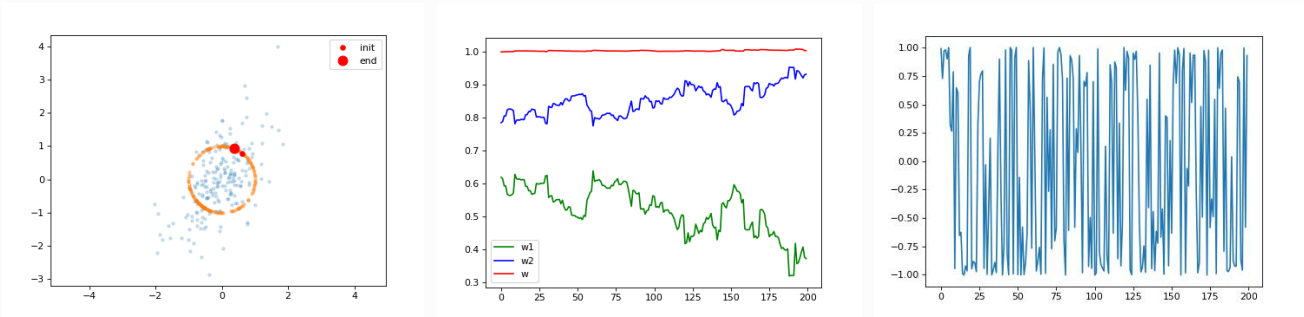
Min y = -1.0044001898158796



The final weight vector w is: (-0.40367492156502466,-0.9170191339036011)

Max y = 1.001901660924199

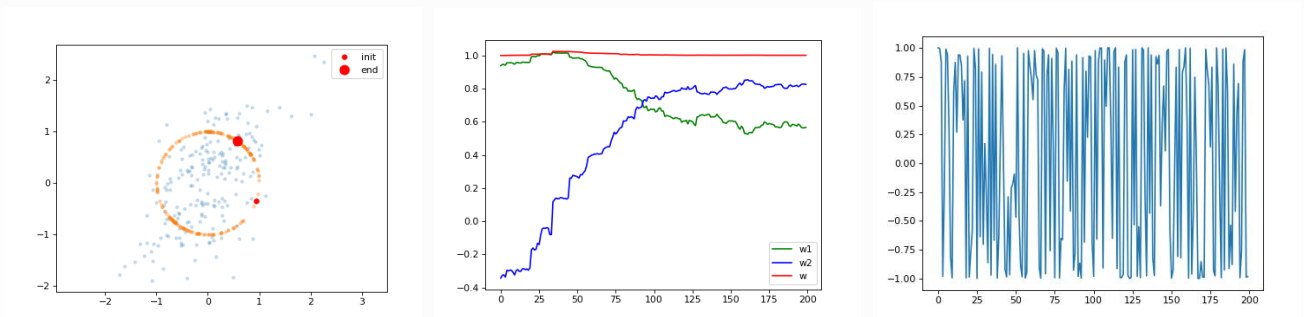
^[[AMin y = -1.0018791265607059



The final weight vector w is: (0.372662227982299,0.93210252583237)

Max y = 1.0038368607823211

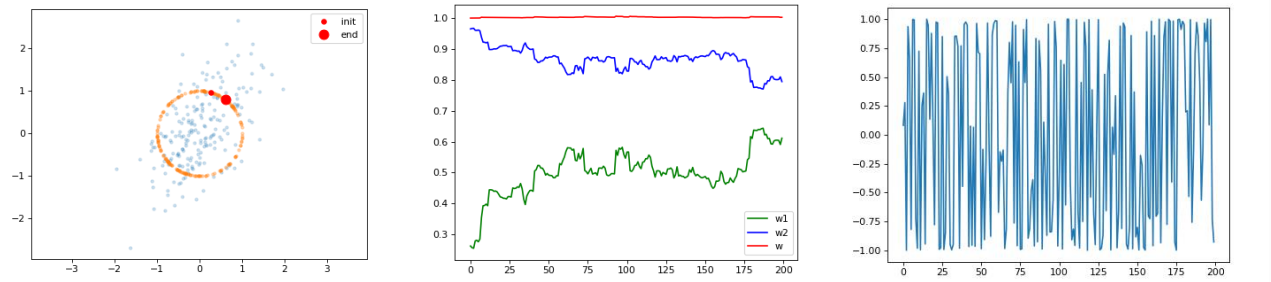
Min y = -1.0035727369727117



The final weight vector w is: (0.5663554590110566,0.8255938815330929)

Max y = 1.0011776482450356

Min y = -1.0010674807517526

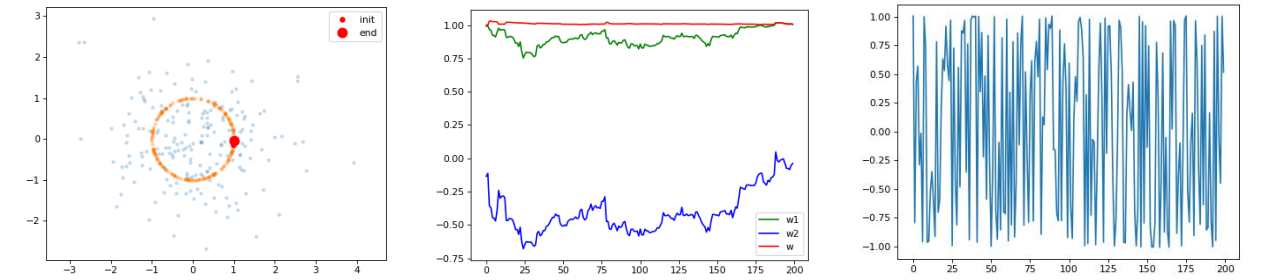


The final weight vector w is: (0.6113990095191917,0.7945102505150426)

Max y = 1.0022509813189695

Min y = -1.0007378941004286

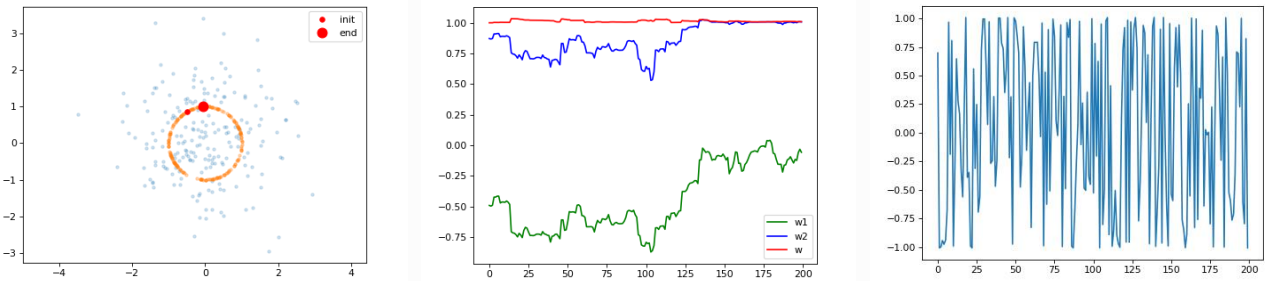
ratio = 1.



The final weight vector w is: (1.0055129200879696,-0.039238476563798)

Max y = 1.0062764422562065

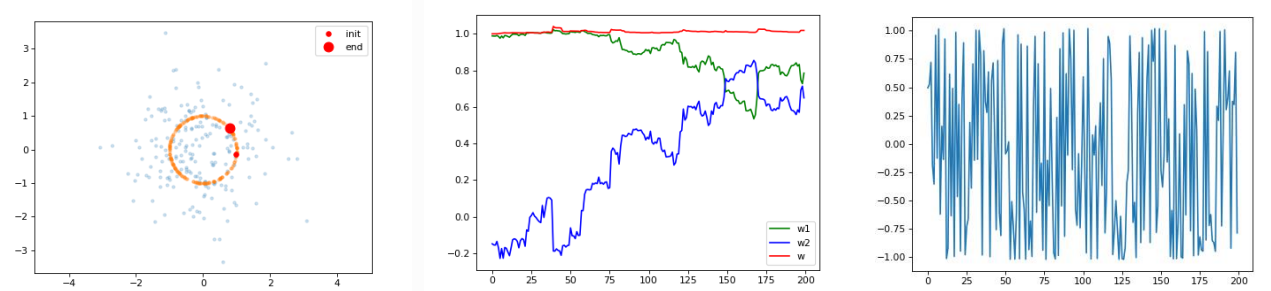
Min y = -1.0061028892192034



The final weight vector w is: (-0.057622171308384054,1.0072274037205198)

Max y = 1.008791604964742

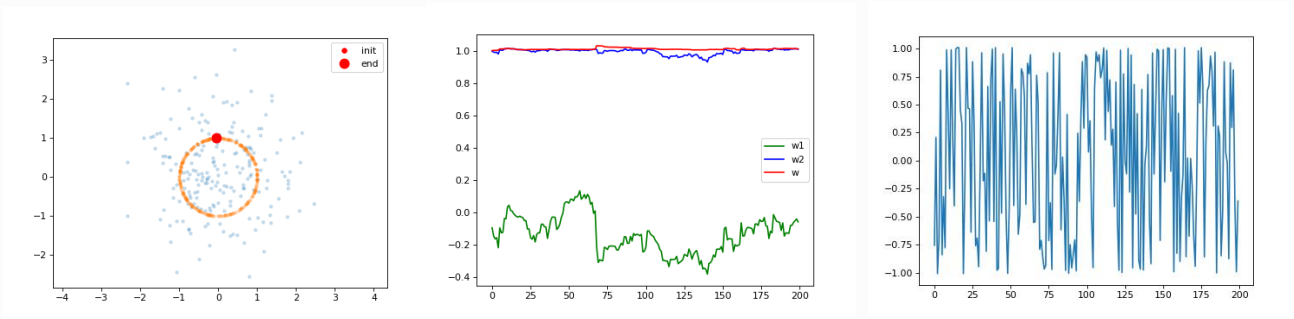
Min y = -1.0087331512429822



The final weight vector w is: (0.7849195193947727,0.6495885881615008)

Max y = 1.018460468856492

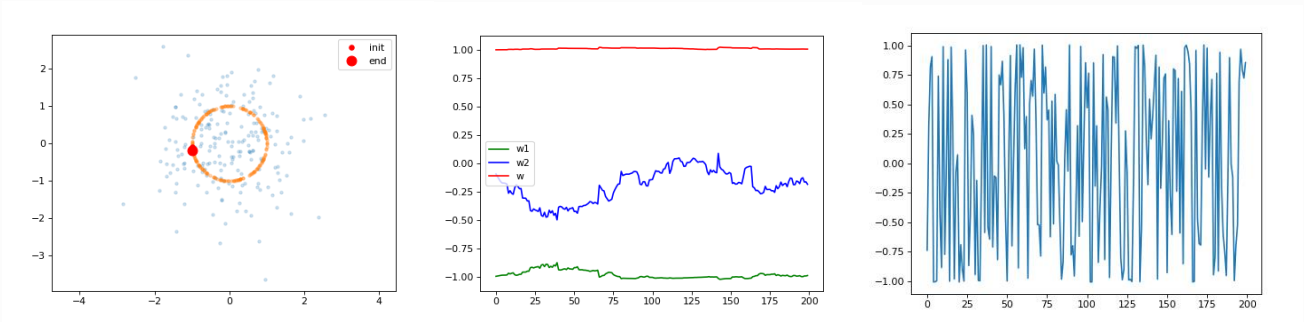
Min y = -1.0187221573137606



The final weight vector w is: (-0.058426285537880464,1.0082296130704596)

Max y = 1.009906774221495

Min y = -1.009614433060446



he final weight vector w is: (-0.9888530652567655,-0.18496832342764 13)

Max y = 1.005526623403155

Min y = -1.0058556979897184

b. 학습 측면에서 각 ratio 는 어떤 차이를 가지는가?

: ratio 가 0.3, 0.5, 즉 타원일 때는 학습을 위한 가중치와 결과가 타원의 축을 따르는 경향이 생긴다.

additional.

: eta, 즉 learning 가 작으면 local minima 에 빠질 수 있다. 이렇게 되면 각 시행 마다의 w_{ij}값이 다르게 된다. 위에서 우리는 eta = 0.04 로 설정했다. 이는 충분히 작은 값이고, 이에 따라서 w_{ij}가 각 시행마다 다른 경우가 꽤 있음을 확인할 수 있다.