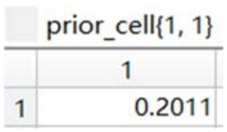
Machine Learning in Robotics Assignment 2

[redacted]

● Exercise 1

● Priors:

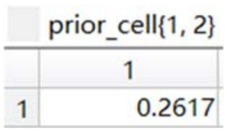
Prior 1



prior_cell{1, 1}

	1
1	0.2011

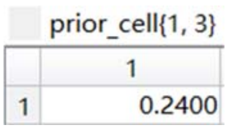
Prior 2



prior_cell{1, 2}

	1
1	0.2617

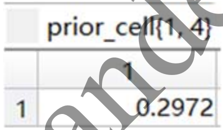
Prior 3



prior_cell{1, 3}

	1
1	0.2400

Prior 4

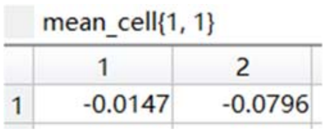


prior_cell{1, 4}

	1
1	0.2972

● Means:

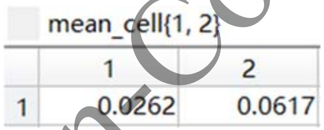
Mean 1



mean_cell{1, 1}

	1	2
1	-0.0147	-0.0796

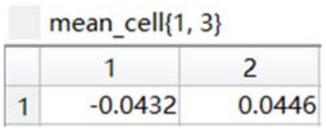
Mean 2



mean_cell{1, 2}

	1	2
1	0.0262	0.0617

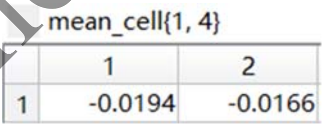
Mean 3



mean_cell{1, 3}

	1	2
1	-0.0432	0.0446

Mean 4

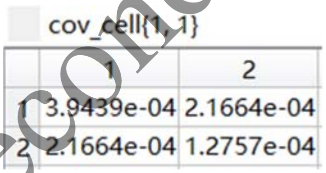


mean_cell{1, 4}

	1	2
1	-0.0194	-0.0166

● Covariance matrices:

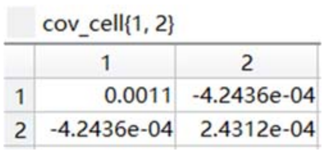
Covariance matrix 1



cov_cell{1, 1}

	1	2
1	3.9439e-04	2.1664e-04
2	2.1664e-04	1.2757e-04

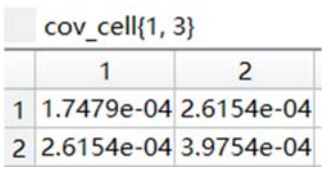
Covariance matrix 2



cov_cell{1, 2}

	1	2
1	0.0011	-4.2436e-04
2	-4.2436e-04	2.4312e-04

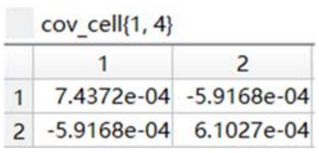
Covariance matrix 3



cov_cell{1, 3}

	1	2
1	1.7479e-04	2.6154e-04
2	2.6154e-04	3.9754e-04

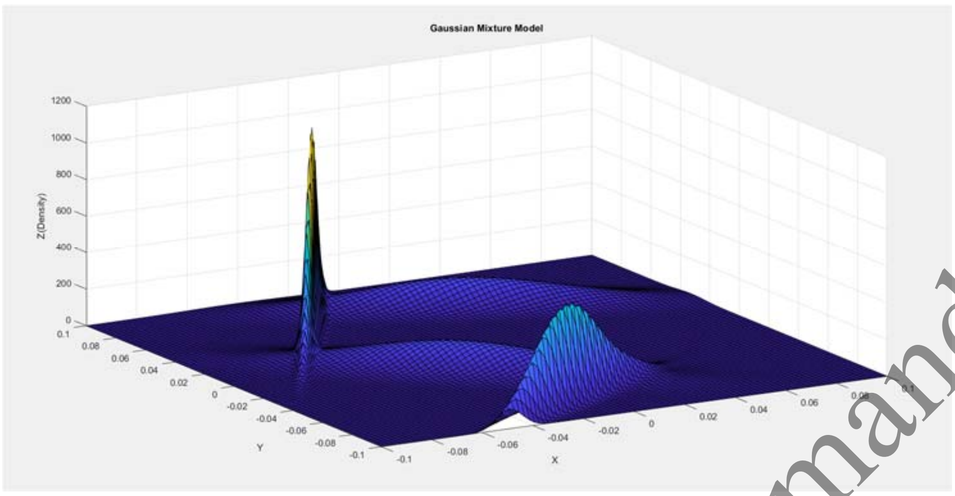
Covariance matrix 4



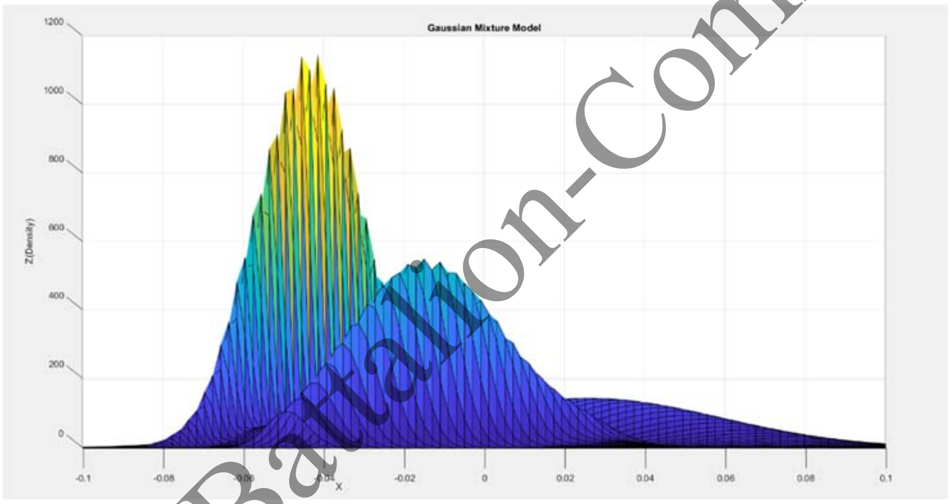
cov_cell{1, 4}

	1	2
1	7.4372e-04	-5.9168e-04
2	-5.9168e-04	6.1027e-04

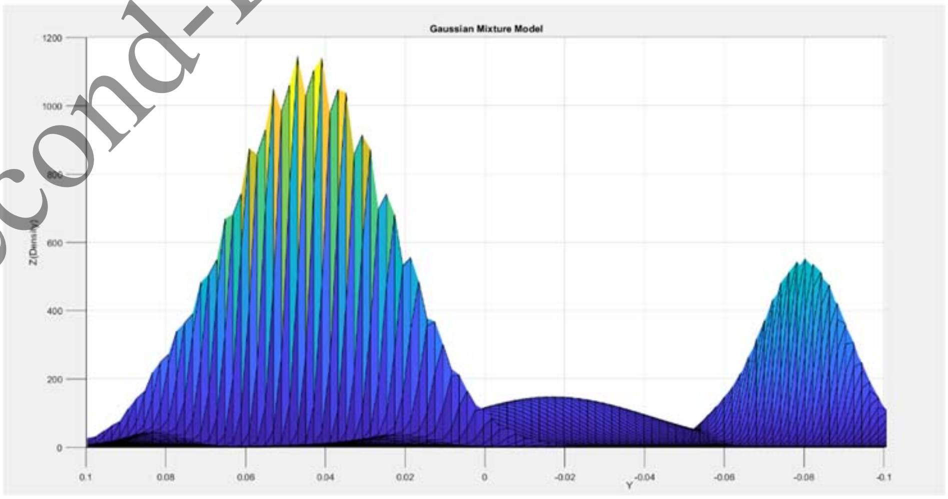
- **Plot of density:**
  We can easily see that there are four peaks and therefore, the probability density function of GMM is the combination of four Gaussian distribution.



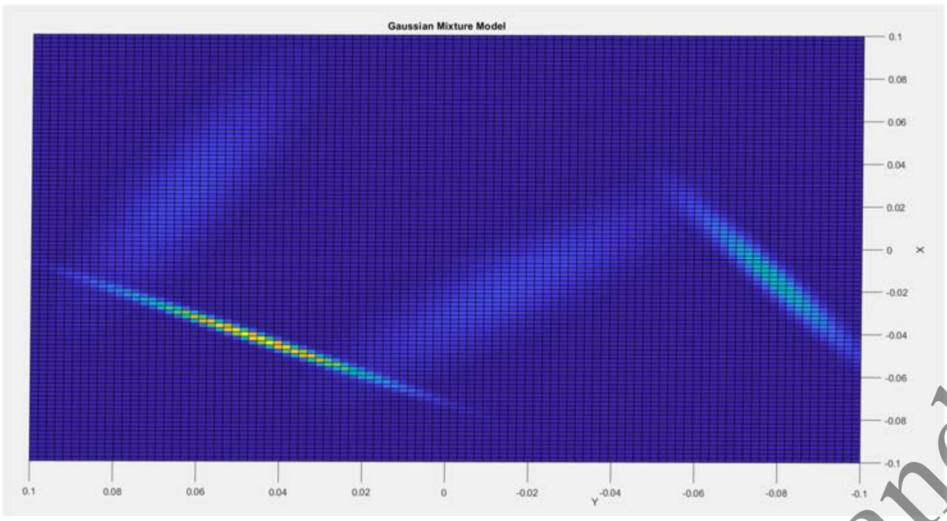
3D View



Front View



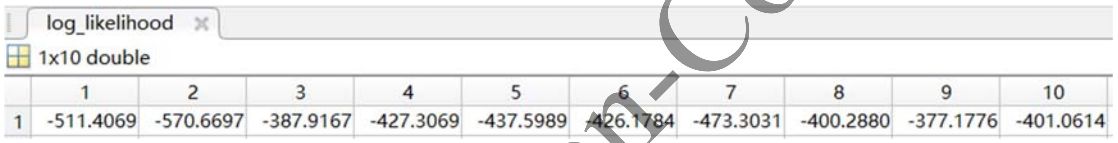
Side View



Top View

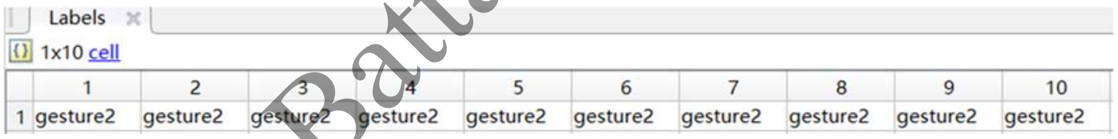
● Exercise 2

Log likelihood of 10 observation sequences are:

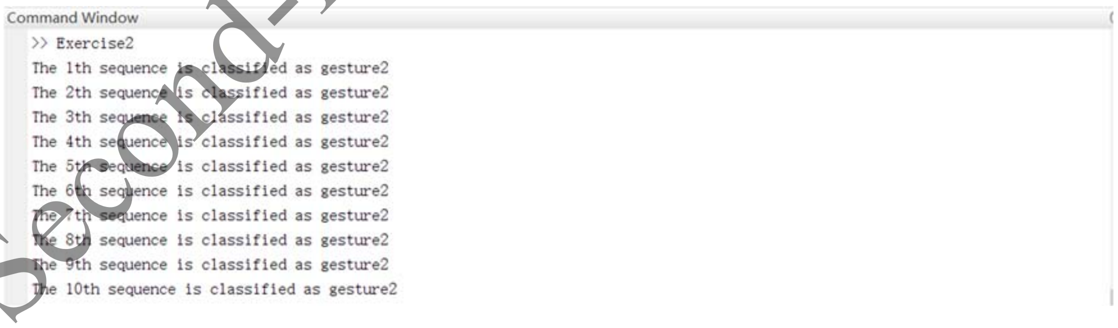

log_likelihood
1x10 double

	1	2	3	4	5	6	7	8	9	10
1	-511.4069	-570.6697	-387.9167	-427.3069	-437.5989	-426.1784	-473.3031	-400.2880	-377.1776	-401.0614

All log likelihoods are smaller than -115, so that all observation sequences are classified as gesture 2. The classification results are:


Labels
1x10 cell

	1	2	3	4	5	6	7	8	9	10
1	gesture2	gesture2	gesture2	gesture2	gesture2	gesture2	gesture2	gesture2	gesture2	gesture2


Command Window

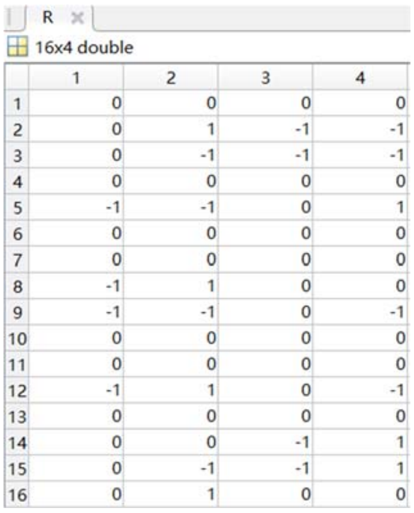
```
>> Exercise2
The 1th sequence is classified as gesture2
The 2th sequence is classified as gesture2
The 3th sequence is classified as gesture2
The 4th sequence is classified as gesture2
The 5th sequence is classified as gesture2
The 6th sequence is classified as gesture2
The 7th sequence is classified as gesture2
The 8th sequence is classified as gesture2
The 9th sequence is classified as gesture2
The 10th sequence is classified as gesture2
```

All 10 observation sequences belong to gesture 2.

● Exercise 3

● WalkPolicyIteration:

1. Reward Matrix:


R
16x4 double

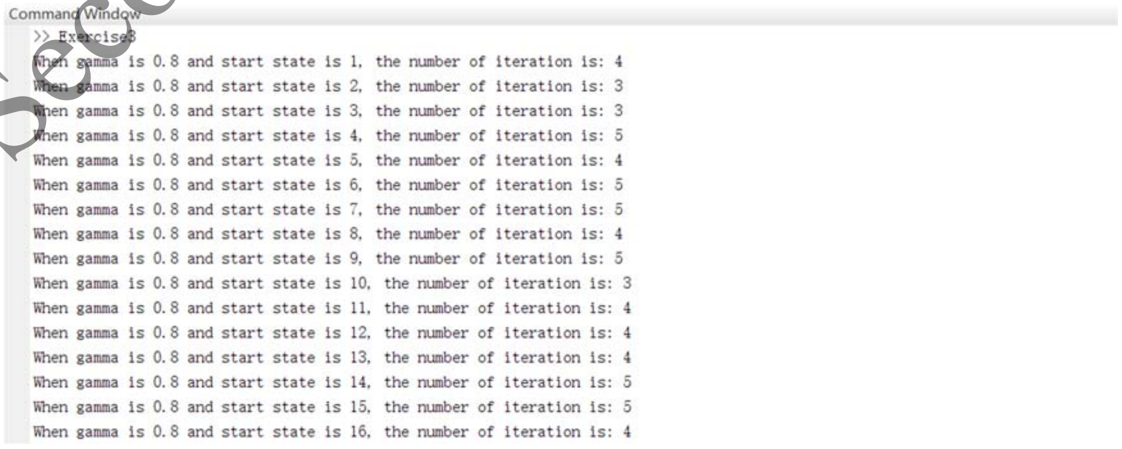
	1	2	3	4
1	0	0	0	0
2	0	1	-1	-1
3	0	-1	-1	-1
4	0	0	0	0
5	-1	-1	0	1
6	0	0	0	0
7	0	0	0	0
8	-1	1	0	0
9	-1	-1	0	-1
10	0	0	0	0
11	0	0	0	0
12	-1	1	0	-1
13	0	0	0	0
14	0	0	-1	1
15	0	-1	-1	1
16	0	1	0	0

2. What value of γ have you used and what is the result of increasing or decreasing γ?

I use γ = 0.8 in this task. γ is discount factor and it defines how important the future reward is. If we decrease γ, the importance of future rewards will decrease, and the algorithm will focus on short-term rewards. If γ = 0, the algorithm will only focus on immediate reward. If we increase γ, the importance of future rewards will increase, and the algorithm will not only focus on short-term rewards but also on long-term rewards. And if the γ = 1, the long-term rewards will play a big role in this case.

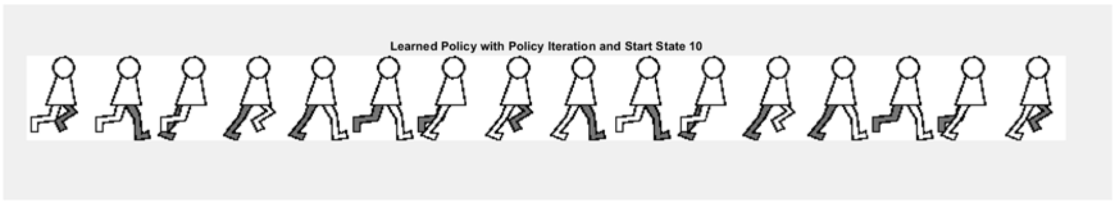
3. Approximately how many iterations are required for policy iteration to converge?

I use the criteria: when policy does not change anymore, which means the policy has converged, the algorithm will stop. And the total number of iterations depends on initial state and γ (one iteration contains policy evaluation and policy improvement). I set γ = 0.8 and required number of iterations are approximately from 3 to 5.

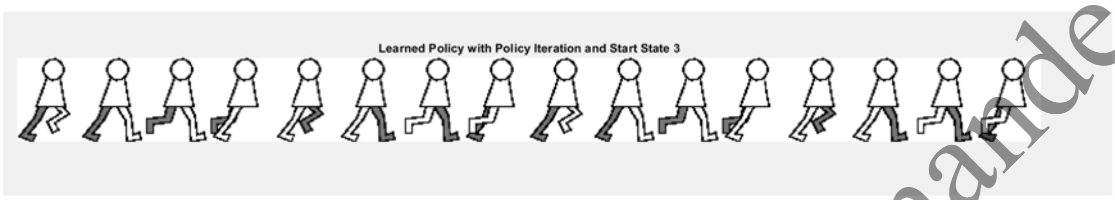


```
>> Exercise3
When gamma is 0.8 and start state is 1, the number of iteration is: 4
When gamma is 0.8 and start state is 2, the number of iteration is: 3
When gamma is 0.8 and start state is 3, the number of iteration is: 3
When gamma is 0.8 and start state is 4, the number of iteration is: 5
When gamma is 0.8 and start state is 5, the number of iteration is: 4
When gamma is 0.8 and start state is 6, the number of iteration is: 5
When gamma is 0.8 and start state is 7, the number of iteration is: 5
When gamma is 0.8 and start state is 8, the number of iteration is: 4
When gamma is 0.8 and start state is 9, the number of iteration is: 5
When gamma is 0.8 and start state is 10, the number of iteration is: 3
When gamma is 0.8 and start state is 11, the number of iteration is: 4
When gamma is 0.8 and start state is 12, the number of iteration is: 4
When gamma is 0.8 and start state is 13, the number of iteration is: 4
When gamma is 0.8 and start state is 14, the number of iteration is: 5
When gamma is 0.8 and start state is 15, the number of iteration is: 5
When gamma is 0.8 and start state is 16, the number of iteration is: 4
```

4. The result of WalkPolicyIteration(s) when starting from state 10 and 3.



Learned Policy with policy iteration and initial state 10 (γ = 0.8)



Learned Policy with policy iteration and initial state 3 (γ = 0.8)

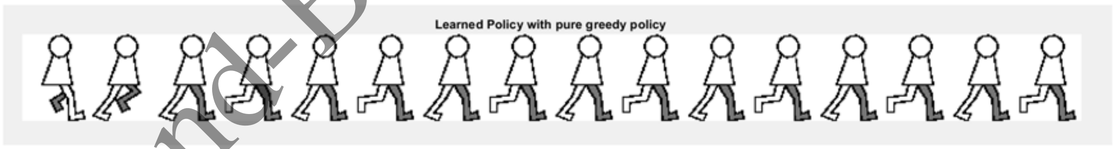
● WalkQLearning(s):

1. Report the value of ε and α.

In this task I use ε = 0.2 and α = 0.6.

2. What happens if a pure greedy policy is used? Implement and compare with the ε-greedy policy. Does it matter what value of ε you use?

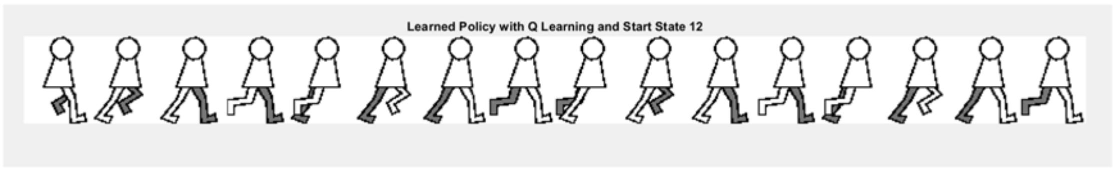
When ε is set to be 0, the policy is pure greedy policy. The algorithm will not tend to explore new state, once it finds a state with positive or big reward, it will always visit this state and therefore, it will converge to not an optimal policy just as follows: (algorithm is stuck in local minima)



Learned Policy with ε = 0 (pure greedy policy) algorithm is stuck in local minima

We can easily see that after two states, the states are same and the algorithm will always visit same states, and not visit new state, the algorithm cannot converge to an optimal policy. And required number of iterations is very small.

If we increase ε, the algorithm will tend to explore new state, and the greater the ε is, the more states the algorithm tend to explore, the faster the algorithm converges to optimal policy. In this case the algorithm can find an optimal policy as follows:



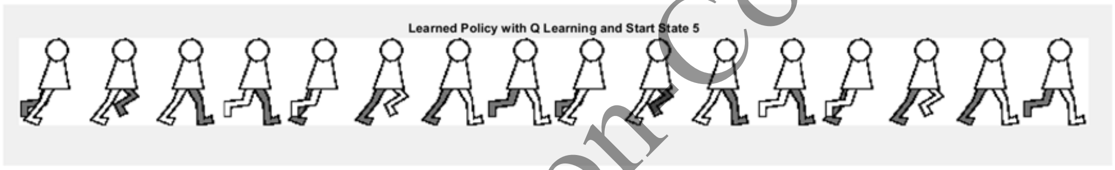
Learned Policy with ε = 0.2 algorithm finds an optimal policy

Large value of ε means the algorithm tend to explore new state and have higher probability to converge to optimal policy. Small value of ε means the algorithm focus on states which has been visited and will have relative higher reward, but it can be stuck in local minima. So that different value of ε can lead to different performance of algorithm. We should consider Explore-Exploit trade-off and pick an appropriate value of ε. In this task I set ε = 0.2.

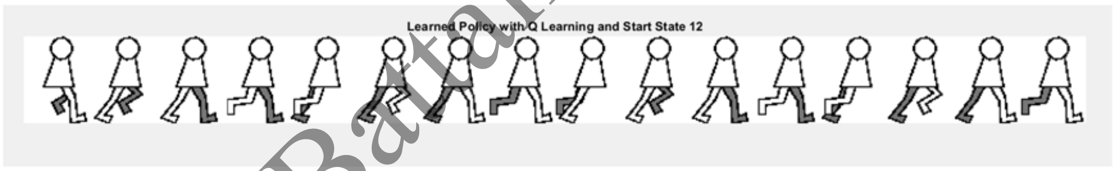
3. Approximately how many steps are necessary for the Q-learning algorithm to find an optimal policy?

I set ε = 0.2, α = 0.6, γ=0.8. I find that the number of iterations with different initial state is different. I try different number of iterations and different initial states and run the function WalkQLearning(s) multiple times and observe whether the output policy changes.
For example, if the initial state is 5 or 12, the required number of iterations is approximately 8000.
But if the initial state is 1, the required number of iterations is approximately 50000.
In order to ensure the convergence, I set the number of iterations is 80000 in this task.

4. The result of WalkQLearning(s) when starting from states 5 and 12.



Learned Policy with Q-Learning and initial state 5 (ε = 0.2, α = 0.6, γ=0.8)



Learned Policy with Q-Learning and initial state 12 (ε = 0.2, α = 0.6, γ=0.8)