Machine Learning in Robotics - Assignment 2

(Zhen Zhou 03721400)

Exercise1:

* Parameters (with threshold = 1e-20)

1. Pi :

Model.Pi

	1	2	3	4
1	0.2972	0.2400	0.2617	0.2011

2. Means (Miu):

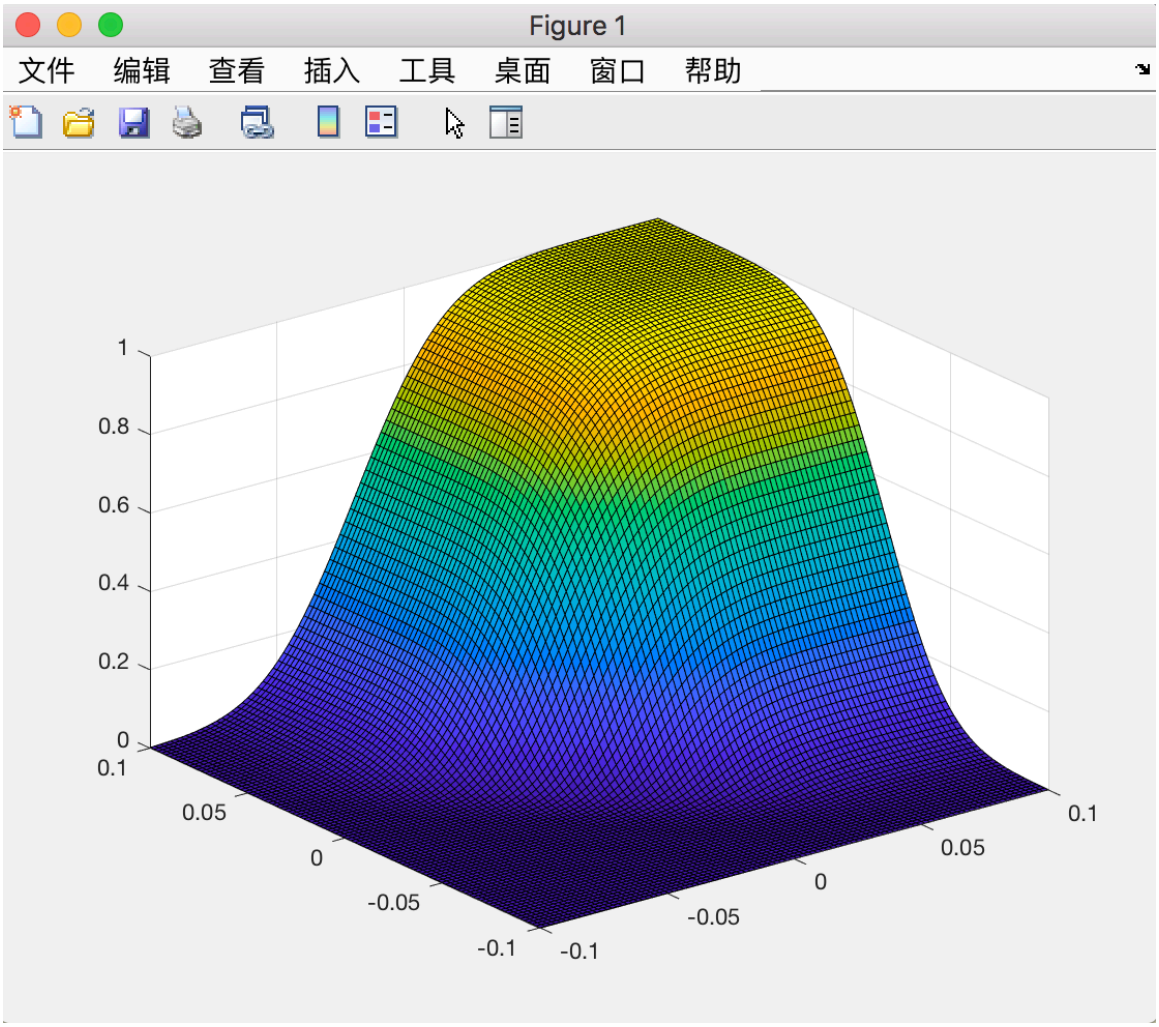
Model.Miu

	1	2
1	-0.0194	-0.0166
2	-0.0432	0.0446
3	0.0262	0.0617
4	-0.0147	-0.0796

3. covariance matrices (Sigma):

```
val(:,:,1) =

   1.0e-03 *

    0.7437   -0.5917
   -0.5917    0.6103

val(:,:,2) =

   1.0e-03 *

    0.1748    0.2615
    0.2615    0.3975

val(:,:,3) =

   1.0e-03 *

    0.0011   -0.0004
   -0.0004    0.0002

val(:,:,4) =

   1.0e-03 *

    0.3944    0.2166
    0.2166    0.1276
```

* Plot



the density values

Exercise2:

```
>> HMM

gesture =

     2     2     2     2     2     2     2     2     2     2
```

All the 10 sequences belong to gesture2.

Exercise3:

* the answers of questions

Policy Iteration:

1. Reward matrix

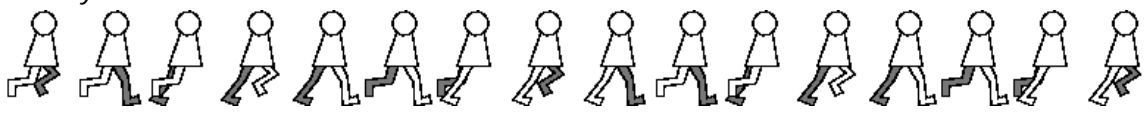
```
reward_def =

     0    -1     0    -1
     0     0    -1    -1
     0     0    -1    -1
     0    -1     0    -1
    -1    -1     0     0
     0    -1     0    -1
     0    -1     0    -1
    -1     1     0     0
    -1    -1     0     0
     0    -1     0    -1
     0    -1     0    -1
    -1     1     0     0
     0    -1     0    -1
     0     0    -1     1
     0     0    -1     1
     0    -1     0    -1
```

2. I chose gamma = 0.7 (i.e. discount factor γ). When we increase gamma (close to 1), the trained policy optimizes long-term gains. When we decrease gamma (close to 0), the trained policy can only optimize short-term gains.

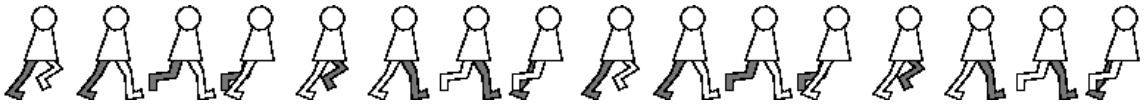
3. When we chose gamma = 0.1, 4 iterations are required for the policy iteration to converge. When we chose gamma = 0.7, 5 iterations are required for the policy iteration to converge. (but also depends on different states)

4. Two plots:

Policy iteration with initial state 10:



Policy iteration with initial state 3:



Q-Learning:

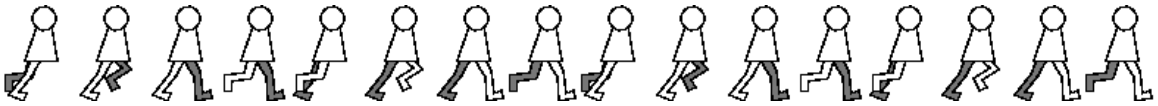
1. I chose epsilon = 0.25, alpha = 0.4 .

2. If a pure greedy policy is used (i.e. epsilon = 0), the robot would be stuck in a loop and unable to explore the entire state space, because epsilon defines the balance between exploitation and exploration.
The value of epsilon does matter: When I chose it close to 0, it would make exploration too rare, which is although bad for start training(because unknown states would be visited), but good for later training (because policy should converge to pure greedy policy). So it is important to adjust the epsilon-value to the be best.

3. When I chose epsilon = 0.25, alpha = 0.4 , around 900-1500 steps are necessary for the Q-learning algorithm to find an optimal policy. As you can see, it depends totally on epsilon I chose and the size of the state space. So I chose 3000 iterations as maximal learning iterations at last.

4. Two Plots:

Q-Learning with initial state 5:



Q-Learning with initial state 12: