Tutorial 8 - Options

Please complete this tutorial to get an overview of options and an implementation of SMDP Q-Learning and Intra-Option Q-Learning.

References:

Recent Advances in Hierarchical Reinforcement Learning is a strong recommendation for topics in HRL that was covered in class. Watch Prof. Ravi's lectures on moodle or nptel for further understanding the core concepts. Contact the TAs for further resources if needed.

```
!pip3 install gym==0.15.7
!pip3 install numpy==1.23.1
```

```
Requirement already satisfied: gym==0.15.7 in
/usr/local/lib/python3.10/dist-packages (0.15.7)
Requirement already satisfied: scipy in
/usr/local/lib/python3.10/dist-packages (from gym==0.15.7) (1.11.4)
Requirement already satisfied: numpy>=1.10.4 in
/usr/local/lib/python3.10/dist-packages (from gym==0.15.7) (1.23.1)
Requirement already satisfied: six in /usr/local/lib/python3.10/dist-
packages (from gym==0.15.7) (1.16.0)
Requirement already satisfied: pyglet<=1.5.0,>=1.4.0 in
/usr/local/lib/python3.10/dist-packages (from gym==0.15.7) (1.5.0)
Requirement already satisfied: cloudpickle~=1.2.0 in
/usr/local/lib/python3.10/dist-packages (from gym==0.15.7) (1.2.2)
Requirement already satisfied: future in
/usr/local/lib/python3.10/dist-packages (from pyglet<=1.5.0,>=1.4.0-
>gym==0.15.7) (0.18.3)
Requirement already satisfied: numpy==1.23.1 in
/usr/local/lib/python3.10/dist-packages (1.23.1)
```

```
'''
A bunch of imports, you don't have to worry about these
'''

import numpy as np
from tqdm import tqdm
import random
import gym
# from gym.wrappers import Monitor
import glob
import io
import matplotlib.pyplot as plt
from IPython.display import HTML

'''
The environment used here is extremely similar to the openai gym ones.
```

```
At first glance it might look slightly different.
The usual commands we use for our experiments are added to this cell
to aid you
work using this environment.
'''

#Setting up the environment
from gym.envs.toy_text.cliffwalking import CliffWalkingEnv
env = CliffWalkingEnv()

env.reset()

#Current State
print(env.s)

# 4x12 grid = 48 states
print ("Number of states:", env.nS)

# Primitive Actions
action = ["up", "right", "down", "left"]
#correspond to [0,1,2,3] that's actually passed to the environment

# either go left, up, down or right
print ("Number of actions that an agent can take:", env.nA)

# Example Transitions
rnd_action = random.randint(0, 3)
print ("Action taken:", action[rnd_action])
next_state, reward, is_terminal, t_prob= env.step(rnd_action)
print ("Transition probability:", t_prob)
print ("Next state:", next_state)
print ("Reward recieved:", reward)
print ("Terminal state:", is_terminal)
env.render()
```

```
36
Number of states: 48
Number of actions that an agent can take: 4
Action taken: left
Transition probability: {'prob': 1.0}
Next state: 36
Reward recieved: -1
Terminal state: False
o  o  o  o  o  o  o  o  o  o  o  o
o  o  o  o  o  o  o  o  o  o  o  o
o  o  o  o  o  o  o  o  o  o  o  o
x  C  C  C  C  C  C  C  C  C  C  T
```

Options

We custom define very simple options here. They might not be the logical options for this settings deliberately chosen to visualise the Q Table better.

```
# We are defining two more options here
# Option 1 ["Away"] - > Away from Cliff (ie keep going up)
# Option 2 ["Close"] - > Close to Cliff (ie keep going down)

def Away(env,state):

    optdone = False
    optact = 0

    if (int(state/12) == 0):
        optdone = True

    return [optact,optdone]

def Close(env,state):

    optdone = False
    optact = 2

    if (int(state/12) == 2):
        optdone = True

    if (int(state/12) == 3):
        optdone = True

    return [optact,optdone]


'''
Now the new action space will contain
Primitive Actions: ["up", "right", "down", "left"]
Options: ["Away","Close"]
Total Actions :["up", "right", "down", "left", "Away", "Close"]
Corresponding to [0,1,2,3,4,5]
'''

{"type":"string"}
```

Task 1

Complete the code cell below

```
#Q-Table: (States x Actions) === (env.ns(48) x total actions(6))

```

```
q_values_SMDP2 = np.zeros((48,6))
#Update_Frequency Data structure? Check TODO 4


ufd2 = np.zeros((48,6))#Update_Frequency Data structure

actions=[0,1,2,3,4,5]
# TODO: epsilon-greedy action selection function
seed = 36
rg = np.random.RandomState(seed)

def egreedy_policy(q_values,state,epsilon):
    if rg.rand() < epsilon:
        return rg.choice([0,1,2,3,4,5])
    else:
        return np.argmax(q_values[state])
```

Task 2

Below is an incomplete code cell with the flow of SMDP Q-Learning. Complete the cell and train the agent using SMDP Q-Learning algorithm. Keep the **final Q-table** and **Update Frequency** table handy (You'll need it in TODO 4)

```
#### SMDP Q-Learning

# Add parameters you might need here
gamma = 0.9
alpha = 0.4
q_values_SMDP = np.zeros((48,6))
ufd1 = np.zeros((48,6))#Update_Frequency Data structure
# Iterate over 1000 episodes
for _ in range(1000):
    state = env.reset()
    done = False

    # While episode is not over
    while not done:

        # Choose action
        action = egreedy_policy(q_values_SMDP, state, epsilon=0.1)

        # Checking if primitive action
        if action < 4:
            # Perform regular Q-Learning update for state-action pair

            next_state, reward, done,_= env.step(action)
            q_values_SMDP[state, action] += alpha*(reward +
gamma*np.max([q_values_SMDP[next_state, action] for action in
```

```
actions]) - q_values_SMDP[state, action])
            ufd1[state,action] += 1
            state = next_state

        # Checking if action chosen is an option
        reward_bar = 0
        if action == 4: # action => Away option

            initial_state = np.copy(state)
            optdone = False
            count=0
            while (optdone == False):

                # Think about what this function might do?
                optact,_ = Away(env,state)
                next_state, reward, done,_ = env.step(optact)


                _,optdone = Away(env,next_state)

                # Is this formulation right? What is this term?
                # Ans: the accumulates return for the entire option
                reward_bar = reward_bar +  (gamma**count)*reward
                count+=1
                # Complete SMDP Q-Learning Update
                # Remember SMDP Updates. When & What do you update?
                state = next_state

            q_values_SMDP[initial_state, action] += alpha*(reward_bar
+ (gamma**count)*np.max([q_values_SMDP[state, action] for action in
actions]) - q_values_SMDP[initial_state, action])
            ufd1[initial_state,action] += 1


        if action == 5: # action => Close option

            initial_state = np.copy(state)
            optdone = False
            count=0
            while (optdone == False):

                # Think about what this function might do?
                optact,_ = Close(env,state)
                next_state, reward, done,_ = env.step(optact)


                _,optdone = Close(env,next_state)

                # Is this formulation right? What is this term?
                # Ans: the accumulates return for the entire option
                reward_bar = reward_bar +  (gamma**count)*reward
```

```
                count+=1
                # Complete SMDP Q-Learning Update
                # Remember SMDP Updates. When & What do you update?
                state = next_state

            q_values_SMDP[initial_state, action] += alpha*(reward_bar
+ (gamma**count)*np.max([q_values_SMDP[state, action] for action in
actions]) - q_values_SMDP[initial_state, action])
            ufd1[initial_state,action] += 1
```

Task 3

Using the same options and the SMDP code, implement Intra Option Q-Learning (In the code cell below). You *might not* always have to search through options to find the options with similar policies, think about it. Keep the **final Q-table** and **Update Frequency** table handy (You'll need it in TODO 4)

```
#### Intra-Option Q-Learning



# Add parameters you might need here
gamma = 0.9
alpha = 0.4

# Iterate over 1000 episodes
for _ in range(1000):
    state = env.reset()
    done = False

    # While episode is not over
    while not done:

        # Choose action
        action = egreedy_policy(q_values_SMDP2, state, epsilon=0.1)

        # Checking if primitive action
        if action < 4:
            # Perform regular Q-Learning update for state-action pair

            next_state, reward, done,_ = env.step(action)
            q_values_SMDP2[state, action] += alpha*(reward +
gamma*np.max([q_values_SMDP2[next_state, action] for action in
actions]) - q_values_SMDP2[state, action])
            ufd2[state,action] += 1

            state = next_state
```

```
        # Checking if action chosen is an option
        reward_bar = 0
        if action == 4: # action => Away option

            #initial_state = state
            optdone = False
            #count=0
            while (optdone == False) :

                # Think about what this function might do?
                optact,_ = Away(env,state)
                #
                next_state, reward, done,_ = env.step(optact)
                _,optdone = Away(env,next_state)

                q_values_SMDP2[state, optact] += alpha*(reward +
gamma*np.max([q_values_SMDP2[next_state, action] for action in
actions]) - q_values_SMDP2[state, optact])
                ufd2[state,optact] += 1

                if not optdone:
                  q_values_SMDP2[state, action] += alpha*(reward +
gamma*q_values_SMDP2[next_state, action] - q_values_SMDP2[state,
action])
                  ufd2[state,action] += 1
                else:
                  q_values_SMDP2[state, action] += alpha*(reward +
gamma*np.max([q_values_SMDP2[next_state, action] for action in
actions]) - q_values_SMDP2[state, action])
                  ufd2[state,action] += 1


                # Complete SMDP Q-Learning Update
                # Remember SMDP Updates. When & What do you update?
                state = next_state




        if action == 5: # action => Close option

            #initial_state = state
            optdone = False
            #count=0
            while (optdone == False) :

                # Think about what this function might do?
                optact,_ = Close(env,state)
                next_state, reward, done,_ = env.step(optact)
```

```
                _,optdone = Close(env,next_state)

                q_values_SMDP2[state, optact] += alpha*(reward +
gamma*np.max([q_values_SMDP2[next_state, action] for action in
actions]) - q_values_SMDP2[state, optact])
                ufd2[state,optact] += 1

                if not optdone:
                  q_values_SMDP2[state, action] += alpha*(reward +
gamma*q_values_SMDP2[next_state, action] - q_values_SMDP2[state,
action])
                  ufd2[state,action] += 1
                else:
                  q_values_SMDP2[state, action] += alpha*(reward +
gamma*np.max([q_values_SMDP2[next_state, action] for action in
actions]) - q_values_SMDP2[state, action])
                  ufd2[state,action] += 1


                # Complete SMDP Q-Learning Update
                # Remember SMDP Updates. When & What do you update?
                state = next_state
```

Task 4

Compare the two Q-Tables and Update Frequencies and provide comments.

```
from pandas import DataFrame
def table_render(arr):
  print(DataFrame(arr,columns=["up", "right", "down", "left", "Away",
"Close"]))

table_render(q_values_SMDP)
```

```
           up       right        down       left       Away       Close
0   -7.729438   -7.685514   -7.681422 -7.699484 -7.693760   -7.682491
1   -7.519931   -7.439467   -7.436725 -7.479699 -7.514018   -7.444857
2   -7.201298   -7.161441   -7.166289 -7.365133 -7.250717   -7.165592
3   -7.060948   -6.849887   -6.851390 -7.016437 -6.891528   -6.855066
4   -6.609679   -6.503400   -6.504559 -6.765017 -6.648498   -6.502854
5   -6.176723   -6.117531   -6.119634 -6.537352 -6.312034   -6.120239
6   -5.954649   -5.689220   -5.691414 -6.049263 -5.854036   -5.690576
7   -5.395808   -5.213450   -5.214202 -5.433794 -5.389702   -5.213467
8   -4.795971   -4.684000   -4.684049 -5.009996 -5.007942   -4.683748
9   -4.479651   -4.094307   -4.094473 -4.394478 -4.273077   -4.094391
10  -3.594209   -3.438714   -3.438715 -4.177258 -3.611374   -3.438828
11  -3.195192   -3.047161   -2.709963 -3.561978 -2.786104   -2.709976
12  -7.662038   -7.444197   -7.449470 -7.526093 -7.595727   -7.447391
```

13	-7.277369	-7.171653	-7.171223	-7.273409	-7.383913	-7.172108
14	-7.053355	-6.859924	-6.860603	-7.169106	-7.243289	-6.860468
15	-6.788018	-6.511802	-6.512346	-6.779623	-6.877801	-6.512217
16	-6.248781	-6.125228	-6.125326	-6.211379	-6.321114	-6.125115
17	-6.072403	-5.695000	-5.695025	-6.067682	-5.906169	-5.695091
18	-5.538227	-5.216833	-5.216928	-5.668148	-5.362161	-5.216851
19	-5.062393	-4.685464	-4.685506	-5.357580	-5.246825	-4.685483
20	-4.535045	-4.095050	-4.095052	-4.492764	-4.838910	-4.095062
21	-4.106196	-3.438988	-3.438987	-4.399222	-3.520659	-3.438983
22	-2.863692	-2.709998	-2.709999	-3.568895	-2.982318	-2.709999
23	-2.393369	-2.050116	-1.900000	-3.201004	-2.584946	-1.900000
24	-7.689858	-7.175705	-7.712314	-7.458112	-8.101569	-7.712320
25	-7.450856	-6.861894	-106.712280	-7.458126	-7.898010	-106.699772
26	-7.172636	-6.513216	-106.712131	-7.175675	-7.695945	-106.711371
27	-6.857511	-6.125795	-106.710874	-6.861888	-7.428133	-106.712248
28	-6.511685	-5.695328	-106.712289	-6.513198	-7.159570	-106.699827
29	-6.125010	-5.217031	-106.705125	-6.125642	-6.834415	-106.700867
30	-5.693978	-4.685590	-106.708275	-5.695323	-6.499349	-106.712134
31	-5.216550	-4.095100	-106.701225	-5.216843	-6.115752	-106.707989
32	-4.684822	-3.439000	-106.626818	-4.685579	-5.690825	-106.681234
33	-4.092836	-2.710000	-106.620663	-4.095087	-5.215467	-106.711420
34	-3.438989	-1.900000	-106.626429	-3.438988	-4.684439	-106.701249
35	-2.709967	-1.899649	-1.000000	-2.709997	-4.094414	-1.000000
36	-7.458134	-106.712275	-7.712320	-7.712320	-8.301769	-7.712321
37	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
38	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
39	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
40	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
41	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
42	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
43	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
44	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
45	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
46	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
47	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000

```
table_render(q_values_SMDP2)
```

	up	right	down	left	Away	Close
0	-7.792267	-7.709757	-7.710474	-7.791814	-7.791814	-7.709723
1	-7.647197	-7.457365	-7.457592	-7.482925	-7.646183	-7.457277
2	-7.387604	-7.175244	-7.175535	-7.500283	-7.349866	-7.175349
3	-7.018157	-6.861658	-6.861833	-6.887870	-7.007155	-6.861724
4	-6.540943	-6.513052	-6.513181	-6.881067	-6.540192	-6.513109
5	-6.398230	-6.125691	-6.125710	-6.523192	-6.392955	-6.125699
6	-6.044907	-5.695276	-5.695315	-6.072767	-6.043893	-5.695288
7	-5.393733	-5.217012	-5.217022	-5.421611	-5.391594	-5.217016
8	-5.078640	-4.685583	-4.685585	-5.056706	-4.984866	-4.685582
9	-4.498153	-4.095098	-4.095099	-4.927027	-4.490912	-4.095098
10	-3.994070	-3.439000	-3.439000	-4.270104	-3.617607	-3.438999

```
11 -2.902852   -2.786104   -2.710000 -2.949415 -2.786104   -2.710000
12 -7.936144   -7.457883   -7.458061 -7.546898 -7.936144   -7.458061
13 -7.710566   -7.175695   -7.175703 -7.489496 -7.710363   -7.175697
14 -7.455317   -6.861894   -6.861894 -7.110841 -7.454715   -6.861894
15 -7.175118   -6.513216   -6.513216 -6.746089 -7.175114   -6.513216
16 -6.859984   -6.125795   -6.125795 -6.664280 -6.859742   -6.125795
17 -6.511123   -5.695328   -5.695328 -6.021644 -6.511123   -5.695328
18 -6.124908   -5.217031   -5.217031 -5.541031 -6.124638   -5.217031
19 -5.695260   -4.685590   -4.685590 -5.522642 -5.695198   -4.685590
20 -5.216633   -4.095100   -4.095100 -4.959590 -5.216633   -4.095100
21 -4.685461   -3.439000   -3.439000 -3.921385 -4.685388   -3.439000
22 -4.093956   -2.710000   -2.710000 -3.148989 -4.093811   -2.710000
23 -3.438772   -2.514791   -1.900000 -2.634014 -3.438767   -1.900000
24 -7.711809   -7.175705   -7.712321 -7.458134 -8.138773   -7.712320
25 -7.458094   -6.861894 -106.712321 -7.458133 -7.936056 -106.709692
26 -7.175703   -6.513216 -106.712321 -7.175699 -7.701831 -106.700325
27 -6.861894   -6.125795 -106.712321 -6.861382 -7.456183 -106.712312
28 -6.513215   -5.695328 -106.712320 -6.513214 -7.167657 -106.709889
29 -6.125795   -5.217031 -106.712321 -6.125054 -6.853207 -106.711769
30 -5.695327   -4.685590 -106.712319 -5.695327 -6.506895 -106.680971
31 -5.217031   -4.095100 -106.712276 -5.216890 -6.125274 -106.701226
32 -4.685590   -3.439000 -106.712320 -4.685522 -5.692505 -106.705275
33 -4.095100   -2.710000 -106.712319 -4.095045 -5.212194 -106.693863
34 -3.438996   -1.900000 -106.712277 -3.436701 -4.669204 -106.680574
35 -2.710000   -1.900000   -1.000000 -2.709140 -4.092291   -1.000000
36 -7.458134 -106.712127   -7.712321 -7.712320 -8.299279   -7.712295
37  0.000000    0.000000    0.000000  0.000000  0.000000    0.000000
38  0.000000    0.000000    0.000000  0.000000  0.000000    0.000000
39  0.000000    0.000000    0.000000  0.000000  0.000000    0.000000
40  0.000000    0.000000    0.000000  0.000000  0.000000    0.000000
41  0.000000    0.000000    0.000000  0.000000  0.000000    0.000000
42  0.000000    0.000000    0.000000  0.000000  0.000000    0.000000
43  0.000000    0.000000    0.000000  0.000000  0.000000    0.000000
44  0.000000    0.000000    0.000000  0.000000  0.000000    0.000000
45  0.000000    0.000000    0.000000  0.000000  0.000000    0.000000
46  0.000000    0.000000    0.000000  0.000000  0.000000    0.000000
47  0.000000    0.000000    0.000000  0.000000  0.000000    0.000000
```

Note that both the methods have converged to similar Q-values. The q-values are very low; close to -106 for action 'down' and option 'close' in states 25-35, since it represents the row above the cliff, and the agent has learnt to avoid those actions.

```
table_render(ufd1)
```

```
        up   right   down  left  Away  Close
0     37.0    90.0   59.0  37.0  36.0   20.0
1     35.0    89.0   56.0  24.0  35.0   22.0
2     32.0    98.0   54.0  23.0  32.0   22.0
3     32.0    97.0   50.0  21.0  29.0   22.0
```

4	27.0	100.0	46.0	19.0	28.0	20.0
5	24.0	88.0	44.0	19.0	25.0	20.0
6	23.0	79.0	42.0	16.0	22.0	19.0
7	19.0	71.0	37.0	13.0	19.0	19.0
8	16.0	61.0	35.0	12.0	18.0	19.0
9	15.0	55.0	34.0	10.0	14.0	19.0
10	11.0	34.0	31.0	9.0	11.0	21.0
11	10.0	9.0	31.0	8.0	8.0	24.0
12	29.0	73.0	27.0	35.0	28.0	27.0
13	24.0	88.0	29.0	23.0	25.0	29.0
14	23.0	94.0	31.0	21.0	25.0	30.0
15	21.0	92.0	30.0	19.0	21.0	29.0
16	18.0	91.0	30.0	16.0	19.0	29.0
17	17.0	88.0	29.0	16.0	16.0	29.0
18	15.0	84.0	30.0	14.0	14.0	27.0
19	13.0	74.0	28.0	13.0	13.0	27.0
20	12.0	64.0	27.0	10.0	12.0	27.0
21	10.0	46.0	28.0	11.0	8.0	27.0
22	6.0	37.0	31.0	8.0	6.0	30.0
23	5.0	6.0	38.0	8.0	5.0	37.0
24	102.0	1339.0	41.0	50.0	50.0	46.0
25	68.0	1271.0	29.0	41.0	29.0	18.0
26	63.0	1205.0	26.0	37.0	39.0	23.0
27	52.0	1149.0	22.0	39.0	31.0	28.0
28	49.0	1094.0	30.0	35.0	32.0	18.0
29	46.0	1072.0	19.0	30.0	22.0	18.0
30	41.0	1026.0	20.0	35.0	26.0	26.0
31	37.0	1006.0	18.0	26.0	23.0	20.0
32	32.0	993.0	14.0	30.0	22.0	16.0
33	24.0	975.0	14.0	28.0	30.0	23.0
34	33.0	973.0	14.0	27.0	20.0	18.0
35	25.0	18.0	913.0	28.0	19.0	87.0
36	1463.0	29.0	61.0	60.0	38.0	66.0
37	0.0	0.0	0.0	0.0	0.0	0.0
38	0.0	0.0	0.0	0.0	0.0	0.0
39	0.0	0.0	0.0	0.0	0.0	0.0
40	0.0	0.0	0.0	0.0	0.0	0.0
41	0.0	0.0	0.0	0.0	0.0	0.0
42	0.0	0.0	0.0	0.0	0.0	0.0
43	0.0	0.0	0.0	0.0	0.0	0.0
44	0.0	0.0	0.0	0.0	0.0	0.0
45	0.0	0.0	0.0	0.0	0.0	0.0
46	0.0	0.0	0.0	0.0	0.0	0.0
47	0.0	0.0	0.0	0.0	0.0	0.0

```
table_render(ufd2)
```

	up	right	down	left	Away	Close
0	39.0	69.0	51.0	37.0	37.0	36.0
1	42.0	85.0	51.0	25.0	37.0	36.0

2	36.0	87.0	51.0	24.0	33.0	38.0
3	36.0	93.0	51.0	21.0	30.0	40.0
4	31.0	93.0	50.0	20.0	26.0	39.0
5	29.0	87.0	45.0	19.0	26.0	36.0
6	28.0	83.0	45.0	17.0	24.0	36.0
7	23.0	83.0	43.0	14.0	19.0	36.0
8	22.0	76.0	40.0	12.0	18.0	35.0
9	19.0	64.0	42.0	12.0	16.0	35.0
10	17.0	41.0	44.0	10.0	11.0	36.0
11	10.0	8.0	57.0	7.0	8.0	52.0
12	95.0	70.0	44.0	35.0	94.0	43.0
13	68.0	81.0	56.0	24.0	65.0	52.0
14	56.0	97.0	59.0	21.0	53.0	58.0
15	58.0	101.0	58.0	18.0	52.0	55.0
16	51.0	103.0	59.0	19.0	48.0	56.0
17	41.0	101.0	61.0	15.0	40.0	58.0
18	44.0	93.0	60.0	13.0	39.0	56.0
19	45.0	90.0	59.0	14.0	42.0	55.0
20	33.0	81.0	59.0	12.0	32.0	56.0
21	32.0	68.0	60.0	8.0	29.0	56.0
22	24.0	58.0	60.0	7.0	22.0	56.0
23	23.0	8.0	128.0	6.0	22.0	97.0
24	134.0	1351.0	74.0	62.0	91.0	44.0
25	87.0	1269.0	47.0	44.0	54.0	21.0
26	83.0	1185.0	51.0	40.0	48.0	18.0
27	68.0	1139.0	45.0	31.0	45.0	32.0
28	66.0	1080.0	39.0	40.0	39.0	21.0
29	57.0	1048.0	47.0	27.0	31.0	24.0
30	50.0	1012.0	35.0	38.0	31.0	16.0
31	60.0	982.0	29.0	27.0	36.0	18.0
32	47.0	954.0	39.0	27.0	28.0	19.0
33	38.0	932.0	35.0	26.0	21.0	17.0
34	31.0	932.0	29.0	17.0	18.0	16.0
35	42.0	36.0	1001.0	17.0	20.0	83.0
36	1471.0	26.0	80.0	61.0	69.0	52.0
37	0.0	0.0	0.0	0.0	0.0	0.0
38	0.0	0.0	0.0	0.0	0.0	0.0
39	0.0	0.0	0.0	0.0	0.0	0.0
40	0.0	0.0	0.0	0.0	0.0	0.0
41	0.0	0.0	0.0	0.0	0.0	0.0
42	0.0	0.0	0.0	0.0	0.0	0.0
43	0.0	0.0	0.0	0.0	0.0	0.0
44	0.0	0.0	0.0	0.0	0.0	0.0
45	0.0	0.0	0.0	0.0	0.0	0.0
46	0.0	0.0	0.0	0.0	0.0	0.0
47	0.0	0.0	0.0	0.0	0.0	0.0

```
np.sum(ufd1),np.sum(ufd2)
```

```
(21188.0, 23541.0)
```

```
print(["up", "right", "down", "left", "Away", "Close"])
print(np.sum(ufd1,axis=0))
print(np.sum(ufd2,axis=0))
```

```
['up', 'right', 'down', 'left', 'Away', 'Close']
[ 2509. 13858.  2098.   871.   850.  1002.]
[ 3136. 13766.  2884.   867.  1354.  1534.]
```

The frequency of updates in intra-option Q-learning surpasses that of SMDP Q-learning. Pay particular attention to the occurrences of actions such as 'up' and 'down', and options like 'Away' and 'Close'; we will observe a notably higher frequency in intra-option Q-learning, as expected. This disparity arises because in intra-option Q-learning, actions 'up' and 'down' are updated even while executing options, and options themselves are updated at each intermediate step.

```
```