tutorial-7-options-intro-1

April 6, 2023

#Tutorial 5 - Options Intro

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

0.0.1 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.

[1]:
```
'''
A bunch of imports, you don't have to worry about these
'''

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

[2]:
```
'''
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)
```

```python
# 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 = 0
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: up
Transition probability: False
Next state: 24
Reward recieved: -1
Terminal state: False
```

```
/usr/local/lib/python3.9/dist-packages/ipykernel/ipkernel.py:283:
DeprecationWarning: `should_run_async` will not call `transform_cell`
automatically in the future. Please pass the result to `transformed_cell`
argument and any exception that happen during thetransform in
`preprocessing_exc_tuple` in IPython 7.17 and above.
  and should_run_async(code)
```

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.

[3]:
```python
# 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 or int(state/12) == 3): #the option should end if
 ↪the state is >= 24, if the condition is only state/12 == 2, then it means we
 ↪are allowing
                                    #the agent to choose this option and thus, the
 ↪agent might run into an infinite loop as the option does not end
        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]
'''
```

[3]:
```
'\nNow the new action space will contain\nPrimitive Actions: ["up", "right",
"down", "left"]\nOptions: ["Away","Close"]\nTotal Actions :["up", "right",
"down", "left", "Away", "Close"]\nCorresponding to [0,1,2,3,4,5]\n'
```

1 Task 1

Complete the code cell below

[4]:
```
#Q-Table: (States x Actions) === (env.ns(48) x total actions(6))
q_values_SMDP = np.zeros((48,6))
q_values_intra_q_learn = np.zeros((48,6))
#Update_Frequency Data structure? Check TODO 4

Update_Frequency_SMDP_q_learning = np.zeros((48,6))
Update_Frequency_Intra_option_q_learning = np.zeros((48,6))
```

```python
# TODO: epsilon-greedy action selection function
def egreedy_policy(q_values,state,epsilon):
  q_values_state = q_values[state]
  actions = [0,1,2,3,4,5]
  best_action    = np.argmax(q_values_state)
  if int(state/12) == 0:     #when the state space is between "0-11", the␣
↪option "Away" is not available, as it terminates in these states
    actions = [0,1,2,3,5]
    q_values_state = q_values[state][actions]
    best_action = actions[np.where(q_values_state==max(q_values_state))[0][0]]
  if int(state/12) >=2:      #when the state space is between "24-47", the␣
↪option "Close" is not available, as it terminates in these states
    actions = [0,1,2,3,4]
    q_values_state = q_values[state][actions]
    best_action = actions[np.where(q_values_state==max(q_values_state))[0][0]]

  if np.random.rand() > epsilon:
    return best_action
  else:
    return np.random.choice(actions,1)[0]
```

2 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)

[5]:
```python
#### SMDP Q-Learning
def return_max_state_SMDP(state):
  q_values_state = q_values_SMDP[state]
  actions = [0,1,2,3,4,5]
  if int(state/12) == 0:     #when the state space is between "0-11", the␣
↪option "Away" is not available, as it terminates in these states
    actions = [0,1,2,3,5]
    q_values_state = q_values_SMDP[state][actions]

  if int(state/12) >=2:      #when the state space is between "24-47", the␣
↪option "Close" is not available, as it terminates in these states
    actions = [0,1,2,3,4]
    q_values_state = q_values_SMDP[state][actions]
  return np.max(q_values_state)

# Add parameters you might need here
gamma = 0.9
alpha = 0.5
```

```
# Iterate over 1000 episodes
for ep in range(1000):
    #print(ep)
    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)
            update = reward + gamma * return_max_state_SMDP(next_state) -␣
↪q_values_SMDP[state][action]
            q_values_SMDP[state][action] +=  alpha*(update)
            Update_Frequency_SMDP_q_learning[state][action] += 1
            state = next_state

        # Checking if action chosen is an option

        if action == 4: # action => Away option
            reward_bar = 0
            curr_state = state
            optdone = False
            tau = 0    # variable defined to keep track of number of time steps␣
↪in the option
            while (optdone == False):

                # Think about what this function might do?-> returns the action␣
↪following the policy defined for the option and also returns "beta" i.e.␣
↪optdone == True indicates option is terminated
                optact,optdone = Away(env,state)
                if optdone == False:
                  next_state, reward, done,_,_ = env.step(optact)
                  tau += 1
                  # Is this formulation right? What is this term? -> Given␣
↪formulation is incorrect
                  #reward_bar = gamma*reward_bar + reward
                  reward_bar = reward_bar + (gamma**(tau-1))*reward

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

```
            #SMDP Q learning
            update = reward_bar + (gamma**(tau) * return_max_state_SMDP(state))␣
↪- q_values_SMDP[curr_state][4]
            q_values_SMDP[curr_state][4] +=  alpha * update
            Update_Frequency_SMDP_q_learning[curr_state][4] += 1
        if action == 5: # action => Close option

            reward_bar = 0
            curr_state = state
            optdone = False
            tau = 0     # variable defined to keep track of number of time steps␣
↪in the option
            while (optdone == False):

                # Think about what this function might do?-> returns the action␣
↪following the policy defined for the option and also returns "beta" i.e.␣
↪optdone == True indicates option is terminated
                optact,optdone = Close(env,state)
                if optdone == False:
                  next_state, reward, done,_,_ = env.step(optact)
                  tau += 1
                  # Is this formulation right? What is this term? -> Given␣
↪formulation is incorrect
                  #reward_bar = gamma*reward_bar + reward
                  reward_bar = reward_bar + (gamma**(tau-1))*reward

                  # Complete SMDP Q-Learning Update
                  # Remember SMDP Updates. When & What do you update?
                  state = next_state
            #SMDP Q learning
            update = reward_bar + ((gamma**(tau)) *␣
↪return_max_state_SMDP(state)) - q_values_SMDP[curr_state][action]
            q_values_SMDP[curr_state][action] +=  alpha * update
            Update_Frequency_SMDP_q_learning[curr_state][action] += 1


print(q_values_SMDP)

print(Update_Frequency_SMDP_q_learning)
```

```
[[ -7.81490676  -7.7070225   -7.7084161   -7.77980834   0.
   -7.70767752]
 [ -7.56243913  -7.45589527  -7.45601519  -7.86270587   0.
   -7.45643482]
 [ -7.25034562  -7.1743979   -7.174326    -7.43858782   0.
   -7.17484678]
 [ -7.09406979  -6.86098047  -6.86097738  -6.95573081   0.
```

```
   -6.8617581 ]
 [  -6.55129235   -6.51254815   -6.51269626   -7.05264587    0.
   -6.51267868]
 [  -6.1417073    -6.12524609   -6.12551124   -6.38627716    0.
   -6.12550278]
 [  -5.94232175   -5.69500932   -5.69512947   -5.77085795    0.
   -5.69510155]
 [  -5.58808448   -5.21683319   -5.21694451   -5.84607698    0.
   -5.21685939]
 [  -4.82368282   -4.68550283   -4.68549003   -5.45412084    0.
   -4.68550497]
 [  -4.49869622   -4.09505949   -4.09504239   -4.93072085    0.
   -4.09506405]
 [  -3.57076324   -3.43899524   -3.43899191   -3.7248032     0.
   -3.43899367]
 [  -3.01662704   -2.92190333   -2.70999996   -3.72902566    0.
   -2.70999997]
 [  -7.63852563   -7.45653234   -7.45697482   -7.47178137   -7.55389951
   -7.45723294]
 [  -7.27065444   -7.17552273   -7.17559218   -7.30609762   -7.23174048
   -7.17554322]
 [  -7.26330617   -6.86183884   -6.86180451   -7.29255681   -6.87561178
   -6.86182836]
 [  -6.52000192   -6.51318733   -6.51318858   -6.7581862    -6.70717244
   -6.51319776]
 [  -6.59213748   -6.12578238   -6.12578375   -6.52184694   -6.21421456
   -6.12578713]
 [  -6.20381241   -5.69532509   -5.69532377   -6.02852294   -5.70222311
   -5.69532535]
 [  -6.06065718   -5.21702973   -5.21703015   -5.81963854   -5.35414764
   -5.21702993]
 [  -5.04595271   -4.68558927   -4.68558947   -5.23578845   -5.25539756
   -4.68558919]
 [  -5.05274537   -4.09509953   -4.09509973   -5.07090541   -4.76561108
   -4.09509958]
 [  -4.21411243   -3.43899986   -3.43899983   -4.41689156   -4.52650362
   -3.43899981]
 [  -3.59931918   -2.70999997   -2.70999998   -3.51518016   -3.68459963
   -2.70999997]
 [  -2.40857953   -2.59804743   -1.9           -2.33931779   -3.38294424
   -1.9       ]
 [  -7.70844377   -7.17570464   -7.71231499   -7.45813417   -8.13438508
    0.        ]
 [  -7.45782123   -6.86189404 -106.71231516   -7.45813403   -7.93797653
    0.        ]
 [  -7.17515601   -6.5132156  -106.71232073   -7.17570456   -7.71021633
    0.        ]
 [  -6.86178485   -6.12579511 -106.71231989   -6.86188765   -7.45661892
```

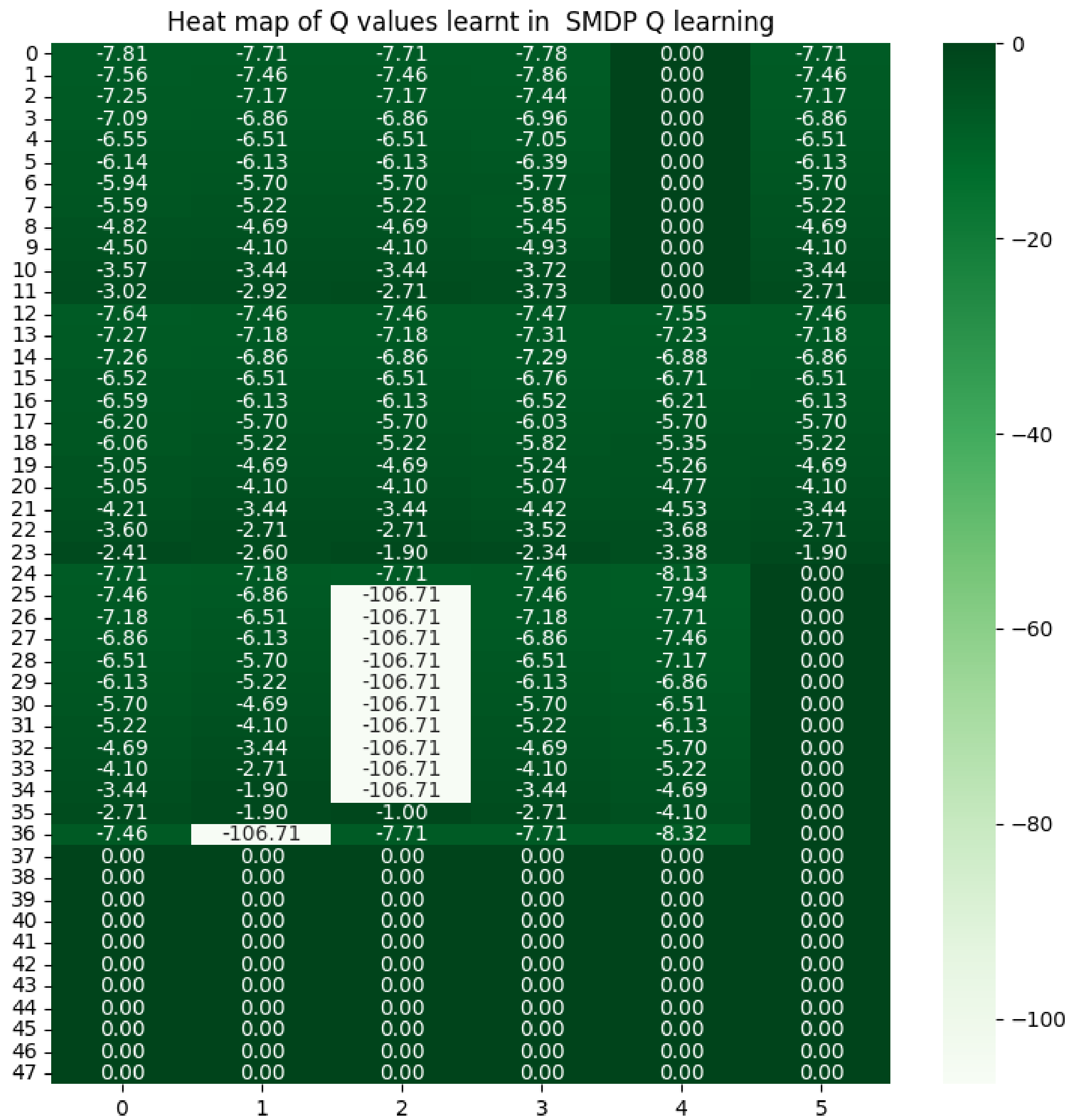
```
     0.        ]
 [  -6.51311133   -5.6953279  -106.71232075   -6.51318235   -7.17470399
     0.        ]
 [  -6.12578517   -5.217031   -106.71229417   -6.12579511   -6.85944506
     0.        ]
 [  -5.69532549   -4.68559    -106.71229188   -5.6953279    -6.5120263
     0.        ]
 [  -5.21702786   -4.0951     -106.71231408   -5.21700458   -6.12544767
     0.        ]
 [  -4.68557073   -3.439      -106.71231296   -4.68558885   -5.69502413
     0.        ]
 [  -4.09509898   -2.71       -106.71051539   -4.09509995   -5.21662263
     0.        ]
 [  -3.43899889   -1.9       -106.71221448   -3.43899037   -4.68556689
     0.        ]
 [  -2.70999983   -1.89999997   -1.           -2.71         -4.09509918
     0.        ]
 [  -7.45813417 -106.71232059   -7.71232054   -7.71232075   -8.32467103
     0.        ]
 [   0.            0.            0.            0.            0.
     0.        ]
 [   0.            0.            0.            0.            0.
     0.        ]
 [   0.            0.            0.            0.            0.
     0.        ]
 [   0.            0.            0.            0.            0.
     0.        ]
 [   0.            0.            0.            0.            0.
     0.        ]
 [   0.            0.            0.            0.            0.
     0.        ]
 [   0.            0.            0.            0.            0.
     0.        ]
 [   0.            0.            0.            0.            0.
     0.        ]
 [   0.            0.            0.            0.            0.
     0.        ]
 [   0.            0.            0.            0.            0.
     0.        ]
 [   0.            0.            0.            0.            0.
     0.        ]]
[[ 30.   70.   49.   30.    0.   19.]
 [ 28.   83.   48.   23.    0.   20.]
 [ 26.   88.   46.   19.    0.   21.]
 [ 25.   85.   44.   16.    0.   22.]
 [ 23.   84.   40.   18.    0.   19.]
 [ 19.   79.   38.   14.    0.   19.]
 [ 18.   72.   36.   12.    0.   19.]
```

```
[  17.   69.   34.   12.    0.   18.]
[  13.   60.   31.   12.    0.   18.]
[  12.   44.   29.   10.    0.   19.]
[   9.   31.   29.    7.    0.   20.]
[   7.    7.   33.    7.    0.   27.]
[  22.   63.   24.   27.   21.   24.]
[  20.   81.   29.   17.   19.   28.]
[  20.   77.   27.   19.   17.   27.]
[  16.   80.   28.   15.   16.   28.]
[  16.   71.   29.   14.   14.   29.]
[  14.   79.   28.   13.   12.   29.]
[  15.   76.   29.   12.   11.   29.]
[  10.   74.   29.   11.   10.   27.]
[  11.   61.   28.   10.    9.   27.]
[   8.   49.   27.    8.    9.   27.]
[   7.   35.   29.    6.    6.   28.]
[   4.    7.   39.    4.    8.   39.]
[  88. 1194.   32.   53.   37.    0.]
[  64. 1134.   27.   37.   46.    0.]
[  45. 1087.   32.   37.   38.    0.]
[  48. 1056.   29.   30.   29.    0.]
[  40. 1039.   34.   27.   32.    0.]
[  54. 1010.   22.   39.   28.    0.]
[  46.  982.   22.   38.   22.    0.]
[  41.  965.   24.   23.   29.    0.]
[  33.  963.   24.   26.   22.    0.]
[  32.  961.   16.   30.   18.    0.]
[  29.  977.   20.   21.   23.    0.]
[  26.   27. 1000.   32.   24.    0.]
[1247.   30.   47.   55.   35.    0.]
[   0.    0.    0.    0.    0.    0.]
[   0.    0.    0.    0.    0.    0.]
[   0.    0.    0.    0.    0.    0.]
[   0.    0.    0.    0.    0.    0.]
[   0.    0.    0.    0.    0.    0.]
[   0.    0.    0.    0.    0.    0.]
[   0.    0.    0.    0.    0.    0.]
[   0.    0.    0.    0.    0.    0.]
[   0.    0.    0.    0.    0.    0.]
[   0.    0.    0.    0.    0.    0.]
[   0.    0.    0.    0.    0.    0.]]
```

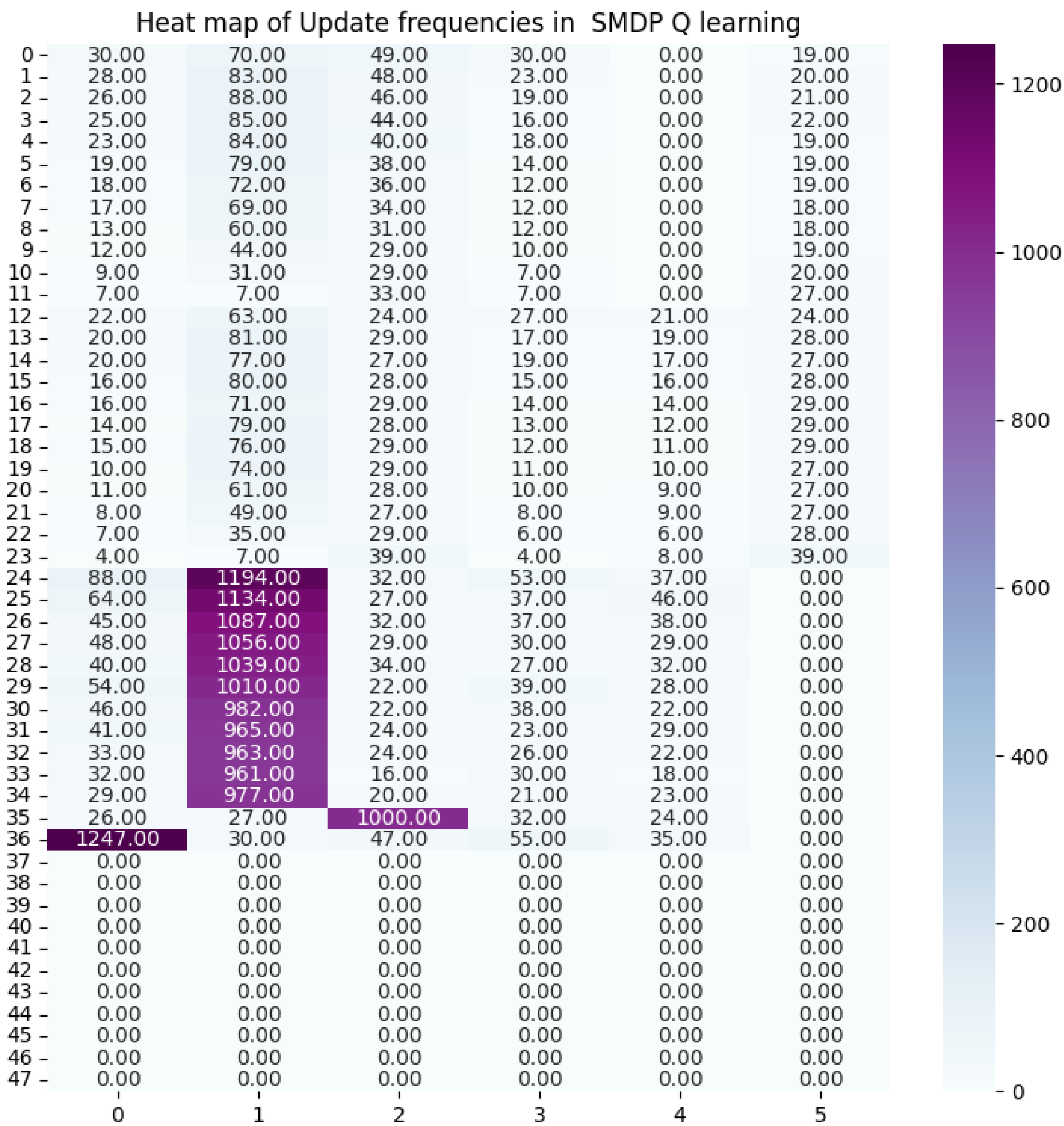
[6]:

```
import seaborn as sns
plt.figure(figsize = (9,9))
plt.title("Heat map of Q values learnt in  SMDP Q learning")
sns.heatmap(q_values_SMDP,annot = True,cmap = "Greens", fmt = '0.2f')
plt.show()
```

```python
plt.figure(figsize = (9,9))
plt.title("Heat map of Update frequencies in  SMDP Q learning")
sns.heatmap(Update_Frequency_SMDP_q_learning,annot = True, cmap = "BuPu", fmt =
 '0.2f')
```



[6]: <Axes: title={'center': 'Heat map of Update frequencies in SMDP Q learning'}>



3 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)

[7]:
```
#### Intra-Option Q-Learning
#logic when the primitive action taken is up -> update the option Away as well
#logic when the primitive action taken is down -> update the option Close as
    well
```

```
#when an option is executed, update all the Q values for both options and
↪primitive actions that are consistent with the observed states in the
↪trajectory

def return_max_state_Intra_Q(state):
  q_values_state = q_values_intra_q_learn[state]
  actions = [0,1,2,3,4,5]
  if int(state/12) == 0:     #when the state space is between "0-11", the
 ↪option "Away" is not available, as it terminates in these states
    actions = [0,1,2,3,5]
    q_values_state = q_values_intra_q_learn[state][actions]

  if int(state/12) >=2:      #when the state space is between "24-47", the
 ↪option "Close" is not available, as it terminates in these states
    actions = [0,1,2,3,4]
    q_values_state = q_values_intra_q_learn[state][actions]
  return np.max(q_values_state)

#define functions that returns  the target for updates
def update_for_Away(env,state,next_state,reward,gamma):
    update_away = 0
    _, beta_st_1 = Away(env,next_state)
    update_away = None
    if beta_st_1 == False:
       update_away      = reward +
 ↪(gamma*q_values_intra_q_learn[next_state][4]) -
 ↪q_values_intra_q_learn[state][4]
    else:
       update_away      = reward + (gamma*
 ↪return_max_state_Intra_Q(next_state)) - q_values_intra_q_learn[state][4]

    return update_away

def update_for_Close(env,state,next_state,reward,gamma):
    update_close = 0
    _, beta_st_1 = Close(env,next_state)
    update_close = None
    if beta_st_1 == False:
       update_close      = reward +
 ↪(gamma*q_values_intra_q_learn[next_state][5]) -
 ↪q_values_intra_q_learn[state][5]
    else:
       update_close      = reward +
 ↪(gamma*return_max_state_Intra_Q(next_state)) -
 ↪q_values_intra_q_learn[state][5]
```

```
    return update_close



# Add parameters you might need here

gamma = 0.9
alpha = 0.5

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

    # While episode is not over
    while not done:

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

#       # Checking if primitive action
        if action < 4:
            # Perform regular Q-Learning update for state-action pair
            if action != 0 and action != 2:

              next_state, reward, done,_,_ = env.step(action)
              update = reward + (gamma* return_max_state_Intra_Q(next_state)) -␣
↪q_values_intra_q_learn[state][action]
              q_values_intra_q_learn[state][action] +=  alpha*(update)
              Update_Frequency_Intra_option_q_learning[state][action] += 1
              state = next_state
            if action == 0: #update for up and away as well
              #update for primitive action up
              next_state, reward, done,_,_ = env.step(action)
              update_primitive = reward + (gamma*␣
↪return_max_state_Intra_Q(next_state)) - q_values_intra_q_learn[state][action]
              Update_Frequency_Intra_option_q_learning[state][action] += 1
              q_values_intra_q_learn[state][action] +=  alpha*(update_primitive)

              #check if the option Away is available in the currrent state
              _,beta = Away(env,state)
              if beta == False:
                #update for option Away
                q_values_intra_q_learn[state][4] += ␣
↪alpha*(update_for_Away(env,state,next_state,reward,gamma))
                Update_Frequency_Intra_option_q_learning[state][4] += 1
```

```
            state = next_state
          if action == 2: #update for down and Close as well
            #update for primitive action down
            next_state, reward, done,_,_ = env.step(action)
            update_primitive = reward + (gamma*␣
↪return_max_state_Intra_Q(next_state)) - q_values_intra_q_learn[state][action]
            Update_Frequency_Intra_option_q_learning[state][action] += 1
            q_values_intra_q_learn[state][action] +=  alpha*(update_primitive)

            #check if the option Close is available in the currrent state
            _,beta = Close(env,state)
            if beta == False:
              #update for option Close
              q_values_intra_q_learn[state][5] += ␣
↪alpha*(update_for_Close(env,state,next_state,reward,gamma))
              Update_Frequency_Intra_option_q_learning[state][5] += 1

            state = next_state


      # Checking if action chosen is an option

      if action == 4: # action => Away option
          reward_bar = 0
          curr_state = state
          optdone = False
          tau = 0    # variable defined to keep track of number of time steps␣
↪in the option
          while (optdone == False):

              optact,optdone = Away(env,state)
              if optdone == False:
                next_state, reward, done,_,_ = env.step(optact)
                tau += 1
                # Is this formulation right? What is this term? -> Given␣
↪formulation is incorrect
                #reward_bar = gamma*reward_bar + reward
                #update for primitive actions seen in trajectories
                update_primitive = reward + (gamma*␣
↪return_max_state_Intra_Q(next_state)) - q_values_intra_q_learn[state][optact]
                q_values_intra_q_learn[state][optact] +=␣
↪alpha*update_primitive
                Update_Frequency_Intra_option_q_learning[state][optact] += 1
                #update for option for states seen in trajectories
                if optdone == False:
```

```
                    q_values_intra_q_learn[state][action] += ␣
↪alpha*(update_for_Away(env,state,next_state,reward,gamma))
                    Update_Frequency_Intra_option_q_learning[state][action] += 1

                  state = next_state

        if action == 5: # action => Close option
            reward_bar = 0
            curr_state = state
            optdone = False
            tau = 0     # variable defined to keep track of number of time steps␣
↪in the option
            while (optdone == False):

                optact,optdone = Close(env,state)
                if optdone == False:
                  next_state, reward, done,_,_ = env.step(optact)
                  tau += 1
                  # Is this formulation right? What is this term? -> Given␣
↪formulation is incorrect
                  #reward_bar = gamma*reward_bar + reward
                  #update for primitive actions seen in trajectories
                  update_primitive = reward + (gamma*␣
↪return_max_state_Intra_Q(next_state)) - q_values_intra_q_learn[state][optact]
                  q_values_intra_q_learn[state][optact] +=␣
↪alpha*update_primitive
                  Update_Frequency_Intra_option_q_learning[state][optact] += 1
                  #update for option for states seen in trajectories
                  if optdone == False:
                    q_values_intra_q_learn[state][action] += ␣
↪alpha*(update_for_Close(env,state,next_state,reward,gamma))
                    Update_Frequency_Intra_option_q_learning[state][action] += 1

                  state = next_state


print(q_values_intra_q_learn)

print(Update_Frequency_Intra_option_q_learning)
```

```
/usr/local/lib/python3.9/dist-packages/ipykernel/ipkernel.py:283:
DeprecationWarning: `should_run_async` will not call `transform_cell`
automatically in the future. Please pass the result to `transformed_cell`
argument and any exception that happen during thetransform in
`preprocessing_exc_tuple` in IPython 7.17 and above.
  and should_run_async(code)
```

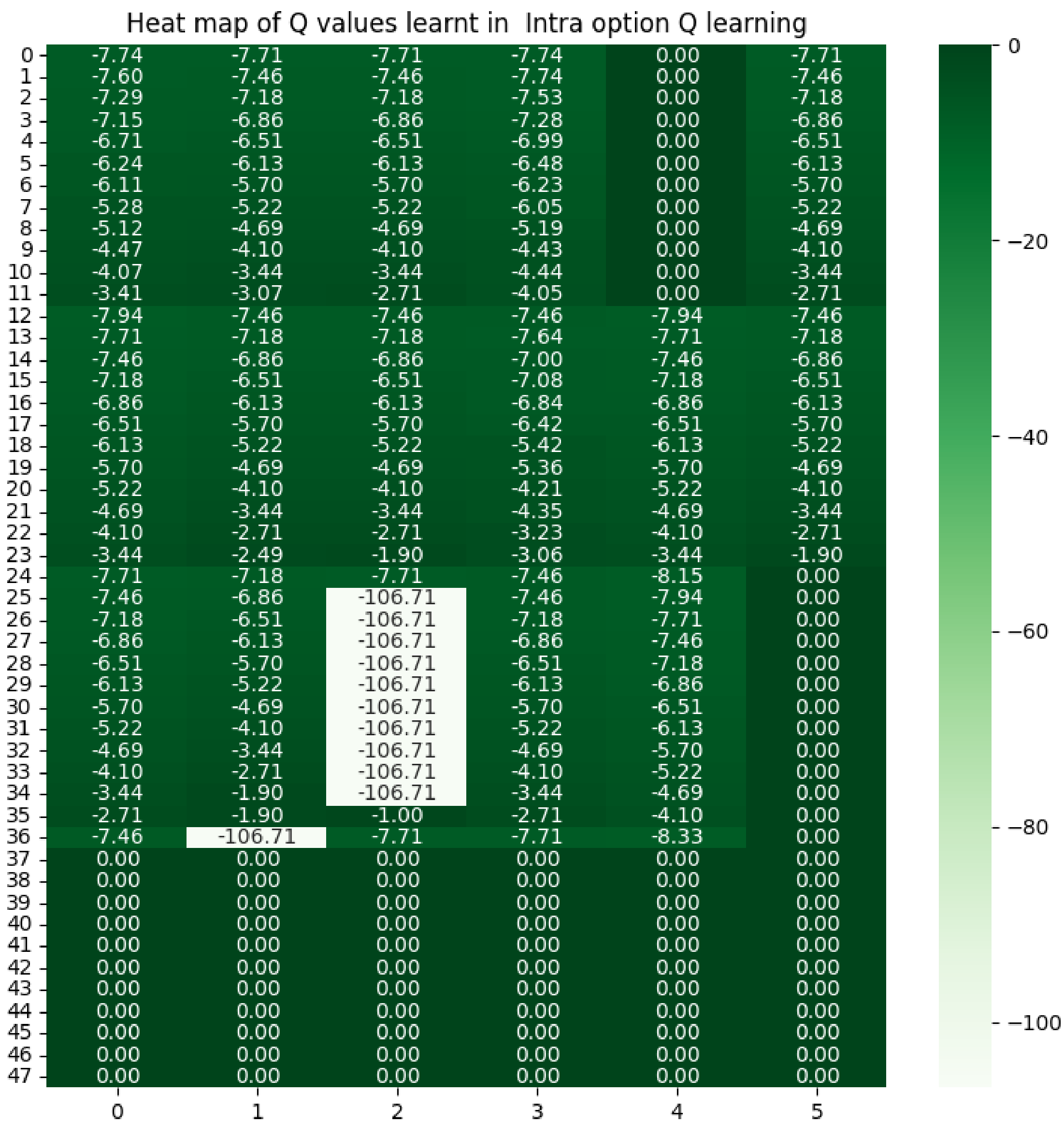
```
[[ -7.74064459   -7.71230177   -7.71229215   -7.74064459    0.
   -7.71229716]
 [ -7.60442368   -7.45812853   -7.45812883   -7.73826221    0.
   -7.45813029]
 [ -7.2855876    -7.17570264   -7.17570335   -7.53069195    0.
   -7.17570368]
 [ -7.14784767   -6.8618934    -6.8618937    -7.28398174    0.
   -6.86189377]
 [ -6.71450063   -6.51321549   -6.51321547   -6.99353046    0.
   -6.51321551]
 [ -6.2360258    -6.12579508   -6.12579508   -6.48047886    0.
   -6.12579509]
 [ -6.10974684   -5.69532789   -5.69532789   -6.22650253    0.
   -5.69532789]
 [ -5.28386757   -5.217031     -5.217031     -6.05204005    0.
   -5.217031  ]
 [ -5.11935912   -4.68559      -4.68559      -5.18937249    0.
   -4.68559   ]
 [ -4.47257388   -4.0951       -4.0951       -4.42598064    0.
   -4.0951    ]
 [ -4.07024168   -3.439        -3.439        -4.43961748    0.
   -3.439     ]
 [ -3.41258392   -3.06979468   -2.71         -4.04622755    0.
   -2.71      ]
 [ -7.94105023   -7.45813345   -7.45813312   -7.46256708   -7.94105023
   -7.45813312]
 [ -7.71214987   -7.17570463   -7.17570463   -7.63737846   -7.71214987
   -7.17570463]
 [ -7.45812807   -6.86189404   -6.86189404   -6.99654626   -7.45812807
   -6.86189404]
 [ -7.17570148   -6.5132156    -6.5132156    -7.08204485   -7.17570148
   -6.5132156 ]
 [ -6.86188619   -6.12579511   -6.12579511   -6.8396106    -6.86188619
   -6.12579511]
 [ -6.51321457   -5.6953279    -5.6953279    -6.42133696   -6.51321457
   -5.6953279 ]
 [ -6.12579353   -5.217031     -5.217031     -5.41533693   -6.12579353
   -5.217031  ]
 [ -5.69532717   -4.68559      -4.68559      -5.36374813   -5.69532717
   -4.68559   ]
 [ -5.217031     -4.0951       -4.0951       -4.21245643   -5.217031
   -4.0951    ]
 [ -4.68559      -3.439        -3.439        -4.34956143   -4.68559
   -3.439     ]
 [ -4.0950996    -2.71         -2.71         -3.23310226   -4.0950996
   -2.71      ]
 [ -3.439        -2.48609531   -1.9          -3.05985937   -3.439
   -1.9       ]
```

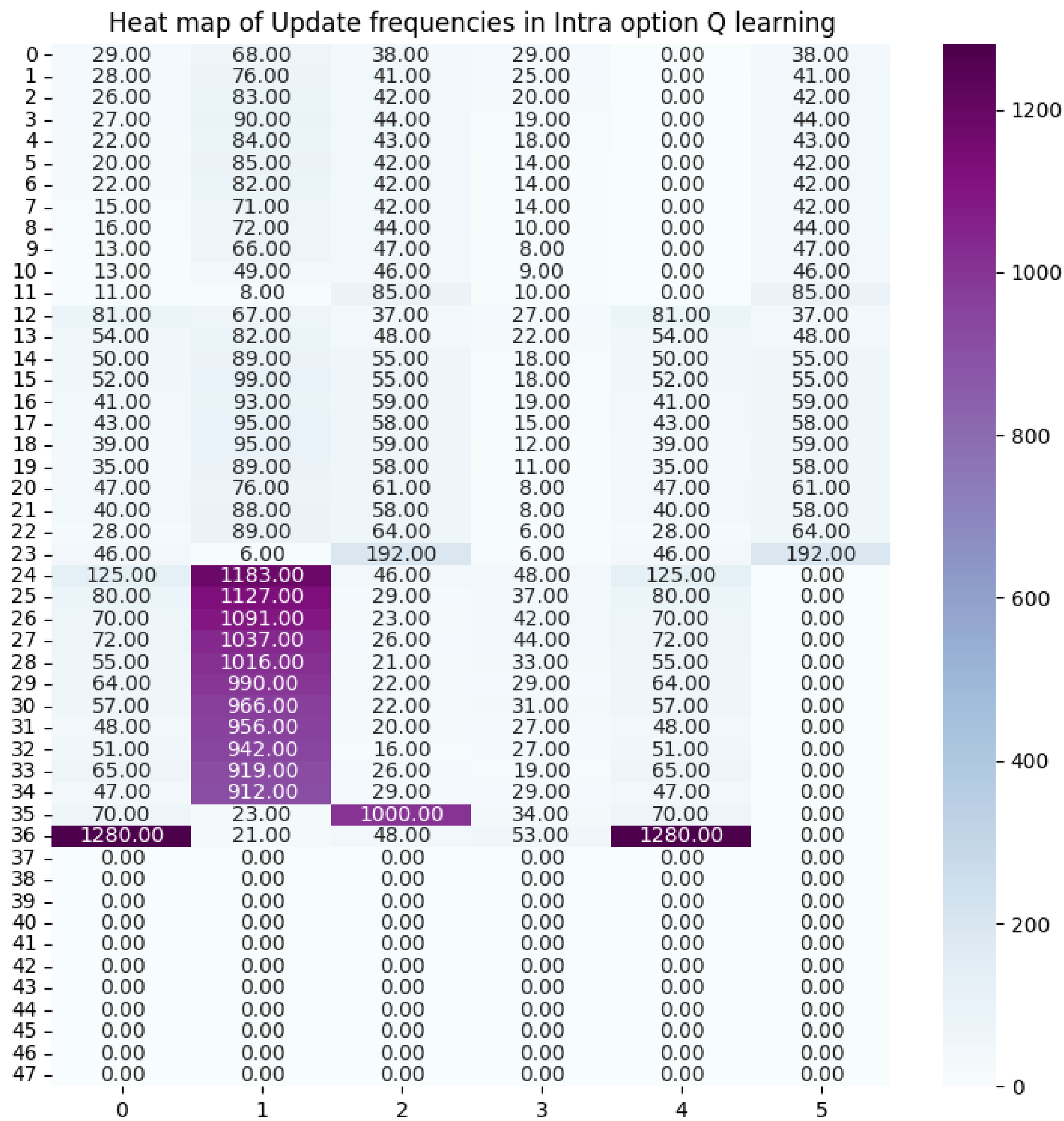
```
[  -7.7123197    -7.17570464   -7.71232075   -7.4581341    -8.14691763
    0.        ]
[  -7.45813417   -6.86189404 -106.71232053   -7.45813407   -7.94089798
    0.        ]
[  -7.17570464   -6.5132156  -106.71230609   -7.17570463   -7.71231047
    0.        ]
[  -6.86189404   -6.12579511 -106.71231893   -6.86189404   -7.45812768
    0.        ]
[  -6.5132156    -5.6953279  -106.71226758   -6.5132154    -7.17565981
    0.        ]
[  -6.12579511   -5.217031   -106.71229247   -6.12579361   -6.86189277
    0.        ]
[  -5.6953279    -4.68559    -106.71229177   -5.69532765   -6.51321206
    0.        ]
[  -5.217031     -4.0951     -106.71221517   -5.21702935   -6.12579404
    0.        ]
[  -4.68559      -3.439      -106.7105084    -4.68558952   -5.6953279
    0.        ]
[  -4.0951       -2.71       -106.71231898   -4.09504185   -5.217031
    0.        ]
[  -3.439        -1.9        -106.71232054   -3.43899997   -4.68558889
    0.        ]
[  -2.71         -1.89999955   -1.           -2.71         -4.0951
    0.        ]
[  -7.45813417 -106.7122357    -7.7123206    -7.71232075   -8.332222
    0.        ]
[   0.            0.            0.            0.            0.
    0.        ]
[   0.            0.            0.            0.            0.
    0.        ]
[   0.            0.            0.            0.            0.
    0.        ]
[   0.            0.            0.            0.            0.
    0.        ]
[   0.            0.            0.            0.            0.
    0.        ]
[   0.            0.            0.            0.            0.
    0.        ]
[   0.            0.            0.            0.            0.
    0.        ]
[   0.            0.            0.            0.            0.
    0.        ]
[   0.            0.            0.            0.            0.
    0.        ]
[   0.            0.            0.            0.            0.
    0.        ]
[   0.            0.            0.            0.            0.
    0.        ]]
```

```
[[  29.   68.   38.   29.    0.   38.]
 [  28.   76.   41.   25.    0.   41.]
 [  26.   83.   42.   20.    0.   42.]
 [  27.   90.   44.   19.    0.   44.]
 [  22.   84.   43.   18.    0.   43.]
 [  20.   85.   42.   14.    0.   42.]
 [  22.   82.   42.   14.    0.   42.]
 [  15.   71.   42.   14.    0.   42.]
 [  16.   72.   44.   10.    0.   44.]
 [  13.   66.   47.    8.    0.   47.]
 [  13.   49.   46.    9.    0.   46.]
 [  11.    8.   85.   10.    0.   85.]
 [  81.   67.   37.   27.   81.   37.]
 [  54.   82.   48.   22.   54.   48.]
 [  50.   89.   55.   18.   50.   55.]
 [  52.   99.   55.   18.   52.   55.]
 [  41.   93.   59.   19.   41.   59.]
 [  43.   95.   58.   15.   43.   58.]
 [  39.   95.   59.   12.   39.   59.]
 [  35.   89.   58.   11.   35.   58.]
 [  47.   76.   61.    8.   47.   61.]
 [  40.   88.   58.    8.   40.   58.]
 [  28.   89.   64.    6.   28.   64.]
 [  46.    6.  192.    6.   46.  192.]
 [ 125. 1183.   46.   48.  125.    0.]
 [  80. 1127.   29.   37.   80.    0.]
 [  70. 1091.   23.   42.   70.    0.]
 [  72. 1037.   26.   44.   72.    0.]
 [  55. 1016.   21.   33.   55.    0.]
 [  64.  990.   22.   29.   64.    0.]
 [  57.  966.   22.   31.   57.    0.]
 [  48.  956.   20.   27.   48.    0.]
 [  51.  942.   16.   27.   51.    0.]
 [  65.  919.   26.   19.   65.    0.]
 [  47.  912.   29.   29.   47.    0.]
 [  70.   23. 1000.   34.   70.    0.]
 [1280.   21.   48.   53. 1280.    0.]
 [   0.    0.    0.    0.    0.    0.]
 [   0.    0.    0.    0.    0.    0.]
 [   0.    0.    0.    0.    0.    0.]
 [   0.    0.    0.    0.    0.    0.]
 [   0.    0.    0.    0.    0.    0.]
 [   0.    0.    0.    0.    0.    0.]
 [   0.    0.    0.    0.    0.    0.]
 [   0.    0.    0.    0.    0.    0.]
 [   0.    0.    0.    0.    0.    0.]
 [   0.    0.    0.    0.    0.    0.]
 [   0.    0.    0.    0.    0.    0.]]
```

[8]:
```python
import seaborn as sns
plt.figure(figsize = (9,9))
plt.title("Heat map of Q values learnt in  Intra option Q learning")
sns.heatmap(q_values_intra_q_learn, annot = True, cmap = 'Greens',fmt = '0.2f')
plt.show()

plt.figure(figsize = (9,9))
plt.title("Heat map of Update frequencies in Intra option Q learning")
sns.heatmap(Update_Frequency_Intra_option_q_learning,annot = True, cmap =
 ↪"BuPu",fmt = '0.2f')
plt.show()
```



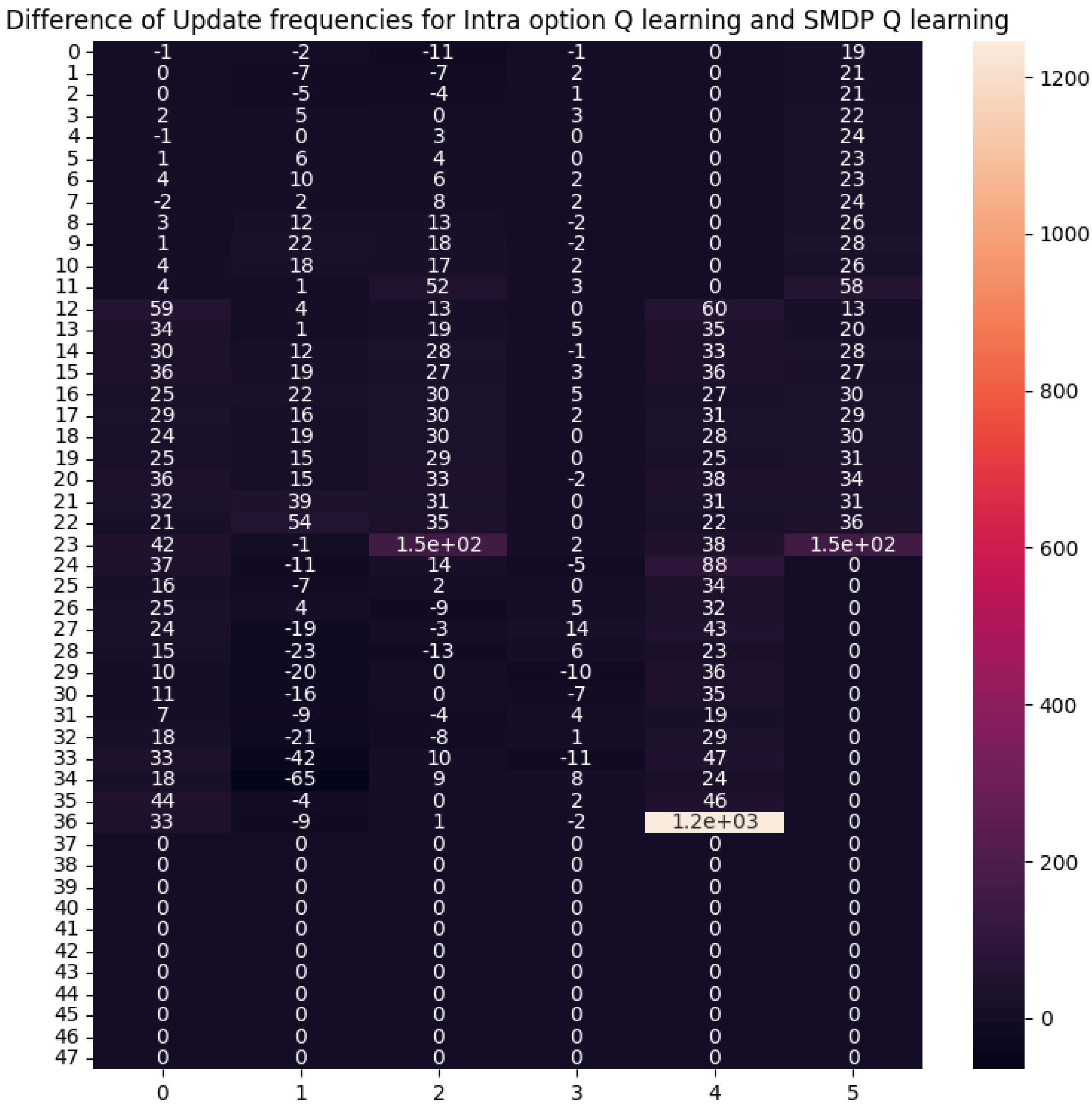


4 Task 4

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

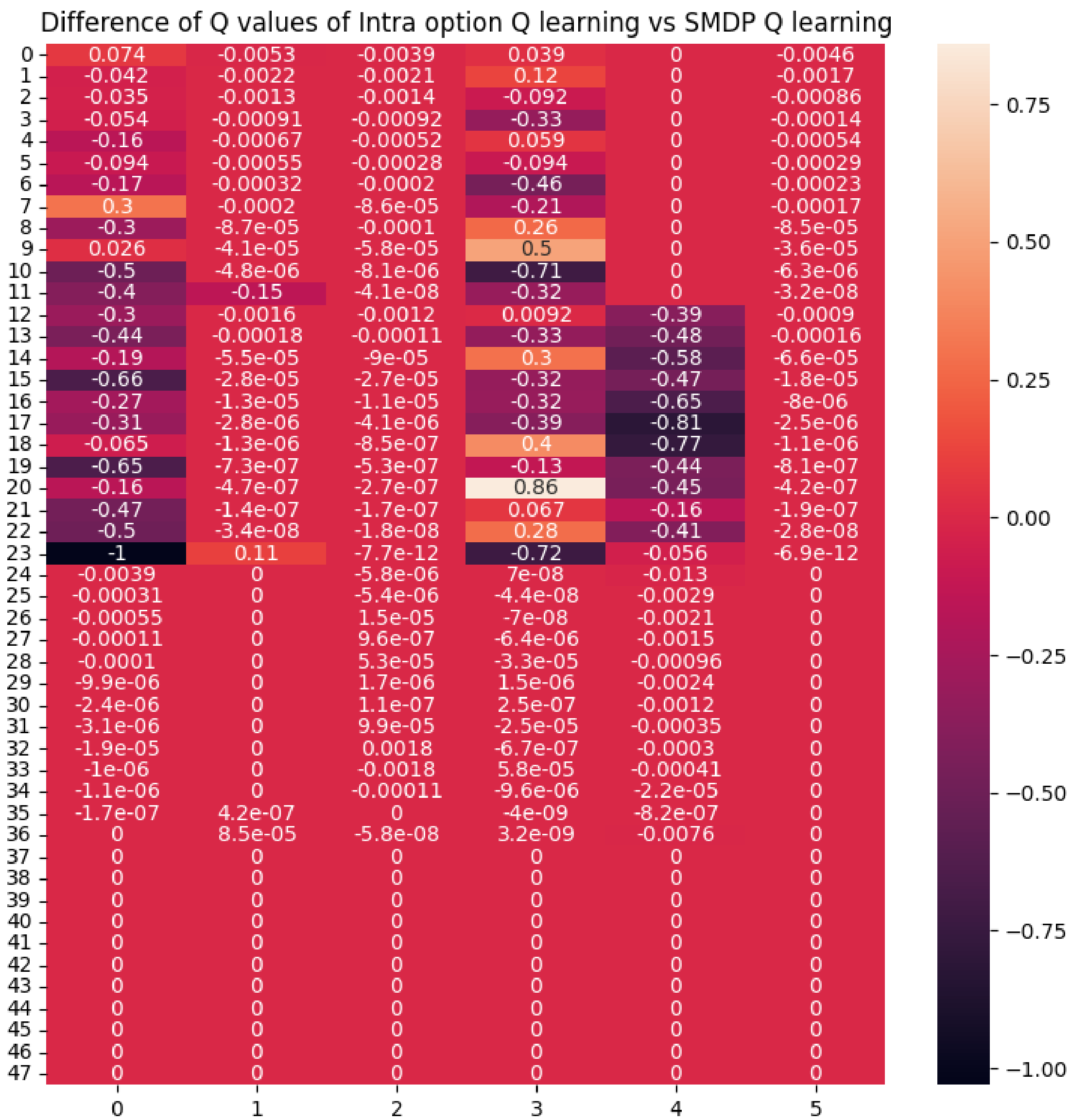
[9]:
```
plt.figure(figsize = (9,9))
plt.title("Difference of Update frequencies for Intra option Q learning and␣
↪SMDP Q learning")
sns.heatmap(Update_Frequency_Intra_option_q_learning -␣
↪Update_Frequency_SMDP_q_learning, annot = True)
```

[9]: <Axes: title={'center': 'Difference of Update frequencies for Intra option Q learning and SMDP Q learning'}>



[10]:
```python
# Use this cell for Task 4 Code
plt.figure(figsize = (9,9))
plt.title("Difference of Q values of Intra option Q learning vs SMDP Q␣
↪learning")
sns.heatmap(q_values_intra_q_learn - q_values_SMDP, annot = True)
```

[10]: <Axes: title={'center': 'Difference of Q values of Intra option Q learning vs SMDP Q learning'}>



link textUse this text cell for your comments - Task 4

Comment on Q values: The Q values learnt by SMDP Q-learning and Intra option Q-learning are different. This is because the reward in SMDP Q-learning is caluclated using the complete discounted reward for an option by taking into account of the time taken by the option to complete. Whereas, Intra Option Q- learning focuses on the one step return to update the Q values of the options and primitive actions. It can interpreted as an off policy algorithm, where the updates are given for the states observed in the trajectories that are obtained by the policy defined for the option. Since, it is an off policy algorithm, the values are also updated for the options that are consistent with the observed state transitions. For majority of states the difference in Q values is very less. Thus, we can conclude that if the number of episodes increases both SMDP Q learning and Intra option Q learning converge to the Optimal value functions.

Comments on Update Frequencies: From the above heat map of "Difference of Update frequencies", it is evident that the frequency of updates in majority of states for Intra option Q learning is more than in SMDP Q learning. In SMDP Q learning we receive updates after the termination of an option. Thus, the update frequencies are relatively low. In contrast, Intra option Q learning updates for every transition observed and does not wait until the option terminates. Thus, intra option Q learning methods allow learning useful information even before the option ends and can be used for multiple options simultaneously that are consistent with the observed trajectories. Thus, the update frequencies for Intra Option Q learning are much higher.