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.

```
!pip install numpy==1.23
```

```
Requirement already satisfied: numpy==1.23 in
/usr/local/lib/python3.10/dist-packages (1.23.0)
```

```
!pip install gym==0.22
```

```
Requirement already satisfied: gym==0.22 in
/usr/local/lib/python3.10/dist-packages (0.22.0)
Requirement already satisfied: numpy>=1.18.0 in
/usr/local/lib/python3.10/dist-packages (from gym==0.22) (1.23.0)
Requirement already satisfied: cloudpickle>=1.2.0 in
/usr/local/lib/python3.10/dist-packages (from gym==0.22) (2.2.1)
Requirement already satisfied: gym-notices>=0.0.4 in
/usr/local/lib/python3.10/dist-packages (from gym==0.22) (0.0.8)
```

```
'''
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
from tqdm import tqdm



'''
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: right
Transition probability: {'prob': 1.0}
Next state: 36
Reward recieved: -100
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

    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

```
# epsilon-greedy action selection function
def egreedy_policy(q_values, state, epsilon):
    if np.random.rand() < epsilon:
        return np.random.randint(6)
    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
q_values_smdp = np.zeros((48,6))
update_frequency_smdp = np.zeros((48, 6))

# Add parameters you might need here
gamma = 0.9
alpha = 0.1

# Iterate over 5000 episodes
for _ in tqdm(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:
            next_state, reward, done, _ = env.step(action)
            best_next_action = np.argmax(q_values_smdp[next_state])
            q_values_smdp[state, action] += alpha * (reward + gamma * 
q_values_smdp[next_state, best_next_action] - q_values_smdp[state, 
action])
            update_frequency_smdp[state, action] += 1
            state = next_state

        # Checking if action chosen is an option
        reward_bar = 0
        if action == 4: # action => Away option
            optdone = False
            state_1 = state
            steps = 0

            while (optdone == False):
                # Think about what this function might do?
                optact, optdone = Away(env,state)
                next_state, reward, done, _= env.step(optact)
                # Is this formulation right? What is this term?
                reward_bar = gamma*reward_bar + reward
                # Complete SMDP Q-Learning Update
                # Remember SMDP Updates. When & What do you update?
                state = next_state
                steps += 1

            q_values_smdp[state_1, action] += alpha*(reward_bar +
(gamma**steps)*np.max(q_values_smdp[state,:]) -
q_values_smdp[state_1,action])
            update_frequency_smdp[state_1, action] += 1

        if action == 5: # action => Close option
            state_1 = state
            steps = 0
            optdone = False
            while (optdone == False):
                optact, optdone = Away(env,state)
                next_state, reward, done, _= env.step(optact)
                reward_bar = gamma*reward_bar + reward
                state = next_state
                steps += 1

            q_values_smdp[state_1, action] += alpha*(reward_bar +
(gamma**steps)*np.max(q_values_smdp[state,:]) -
q_values_smdp[state_1,action])
            update_frequency_smdp[state_1, action] += 1

100%|██████████| 1000/1000 [00:01<00:00, 623.85it/s]
```

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)

```
q_values_intra = np.zeros((48,6))
update_frequency_intra = np.zeros((48, 6))
gamma = 0.9
alpha = 0.1
epsilon = 0.1
for _ in tqdm(range(1000)):
    state = env.reset()
    done = False

    while not done:
        action = egreedy_policy(q_values_intra, state, epsilon)
        if action < 4:
            next_state, reward, done, _ = env.step(action)

            valid_options = [action]
            if action == Away(env, state)[0]:
                valid_options.append(4)
            if action == Close(env, state)[0]:
                valid_options.append(5)

            for o in valid_options:
                if o == 4:
                    _, optdone = Away(env, state)
                elif o == 5:
                    _, optdone = Close(env, state)
                else:
                    optdone = True

                beta = 1 if optdone else 0
                Q_tilde = (1 - beta) * q_values_intra[next_state, o] +
beta * max(q_values_intra[next_state, :])
                q_values_intra[state, o] += alpha * (reward + gamma *
Q_tilde - q_values_intra[state, o])
                update_frequency_intra[state, o] += 1

            state = next_state

        reward_bar = 0
        if action == 4:
            steps = 0
            optdone = False
            while (optdone == False):
                # Think about what this function might do?
                optact, optdone = Away(env,state)
                next_state, reward, done, _= env.step(optact)
                # Is this formulation right? What is this term?
                reward_bar = gamma*reward_bar + reward
                # Complete SMDP Q-Learning Update
                # Remember SMDP Updates. When & What do you update?
                state = next_state
                steps += 1

            q_values_intra[state, action] += alpha*(reward_bar +
(gamma**steps)*np.max(q_values_intra[state,:]) -
q_values_intra[state,action])
            update_frequency_intra[state, action] += 1

        if action == 5:
            steps = 0
            optdone = False
            while (optdone == False):
                optact, optdone = Away(env,state)
                next_state, reward, done, _= env.step(optact)
                reward_bar = gamma*reward_bar + reward
                state = next_state
                steps += 1

            q_values_intra[state, action] += alpha*(reward_bar +
(gamma**steps)*np.max(q_values_intra[state,:]) -
q_values_intra[state,action])
            update_frequency_intra[state, action] += 1

100%|██████████| 1000/1000 [00:01<00:00, 611.92it/s]
```

Task 4

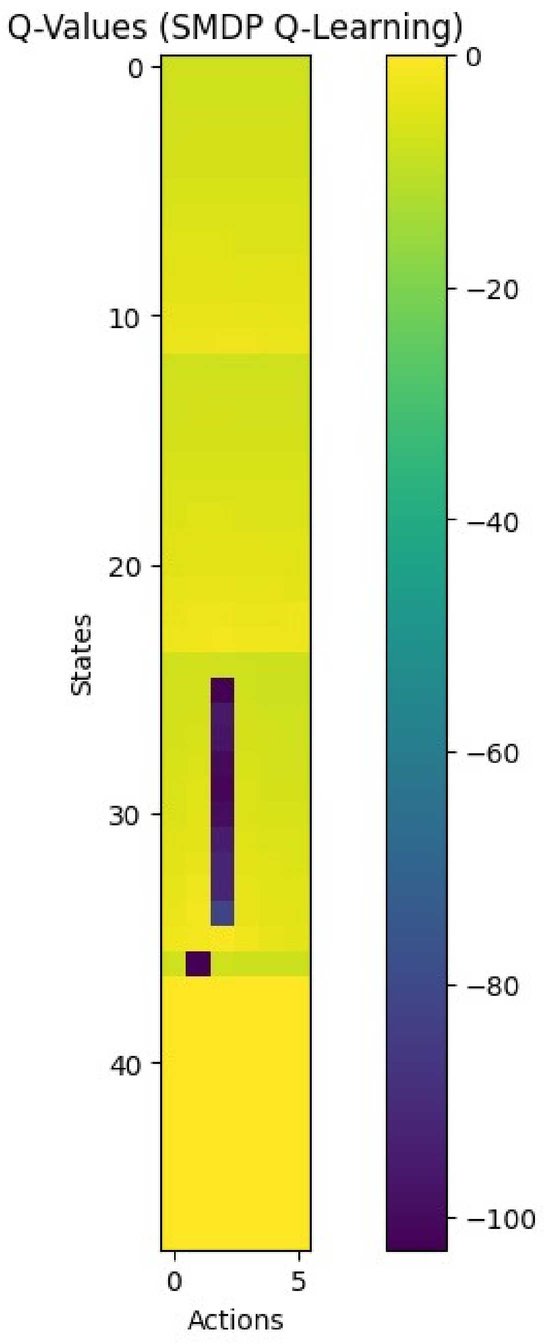
Compare the two Q-Tables and Update Frequencies and provide comments.

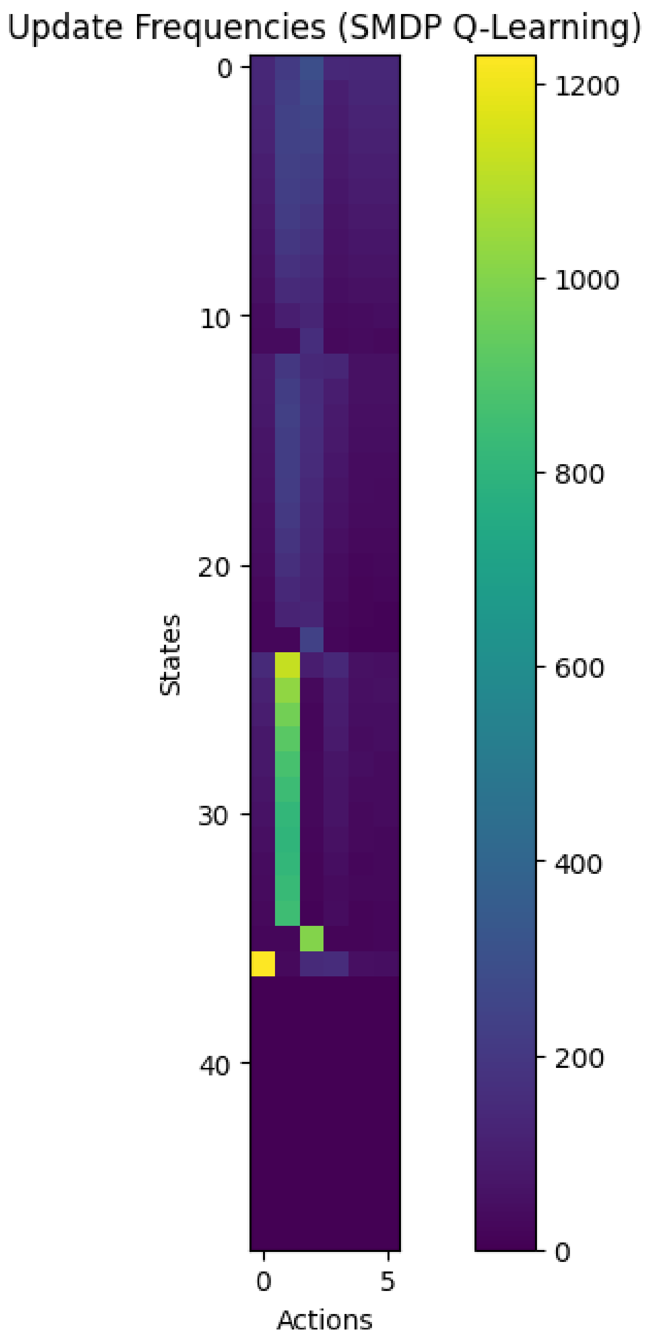
```
# Use this cell for Task 4 Code# Task 4
# Compare the two Q-Tables and Update Frequencies and provide
comments.

# Compare final Q-tables
print("Comparison of final Q-tables:")
print("SMDP Q-Learning Q-table:")
print(q_values_smdp)
print("\nIntra-Option Q-Learning Q-table:")
print(q_values_intra)

# Compare update frequencies
print("\nComparison of update frequencies:")
print("SMDP Q-Learning Update Frequency:")
print(update_frequency_smdp)
print("\nIntra-Option Q-Learning Update Frequency:")
print(update_frequency_intra)

Comparison of final Q-tables:
SMDP Q-Learning Q-table:
[[  -7.50877774   -7.50954513   -7.51442257   -7.52002679   -
7.51193566
    -7.51997296]
 [  -7.32936137   -7.3105195    -7.31482208   -7.34798965   -7.3439365
    -7.32990203]
 [  -7.09673422   -7.07188656   -7.0772615    -7.0834586    -
7.09997476
    -7.09532597]
 [  -6.84054652   -6.787103     -6.78640821   -6.7850336    -
6.79731326
    -6.82385045]
 [  -6.49970169   -6.45532361   -6.45621093   -6.57527387   -
6.46828304
    -6.48743223]
 [  -6.08685753   -6.0865382    -6.08843547   -6.1638715    -
6.13503919
    -6.10839513]
 [  -5.71178006   -5.66646538   -5.66762344   -5.69893591   -
5.67528755
    -5.73098255]
 [  -5.2096934    -5.1983116    -5.19844184   -5.27566419   -
5.26439787
    -5.22411082]
 [  -4.77796163   -4.6755327    -4.67636459   -4.96691502   -4.7163492
    -4.6828786 ]
 [  -4.20009785   -4.09010502   -4.0901687    -4.43810666   -
4.11368439
    -4.10847639]
 [  -3.439389     -3.43714149   -3.43716368   -3.6613163    -
3.43754405
    -3.64521381]
 [  -2.85807982   -3.02931396   -2.7099948    -2.80138667   -
2.93185672
    -2.81494641]
 [  -7.32771506   -7.30796401   -7.32247985   -7.33473997   -
7.36970242
    -7.39017597]
 [  -7.11389779   -7.09794043   -7.1032606    -7.11075476   -
7.12986954
    -7.11427587]
 [  -6.8541261    -6.81356452   -6.8226979    -6.87690789   -
6.98189471
    -6.99191587]
 [  -6.58332287   -6.48257401   -6.48411031   -6.62684682   -
6.64588909
    -6.67294102]
 [  -6.1305482    -6.10856313   -6.10910955   -6.17743435   -
6.18392517
    -6.21407417]
 [  -5.93695309   -5.68546257   -5.68502006   -5.74558627   -
5.93611348
    -5.92673909]
 [  -5.39721577   -5.21154724   -5.21154766   -5.32482837   -
5.50064202
    -5.66894344]
 [  -4.809273     -4.68296927   -4.6831529    -4.80322861   -
4.95579174
    -5.13381159]
 [  -4.47524331   -4.0941282    -4.09415447   -4.16155597   -
4.34294927
    -4.56056478]
 [  -3.70137237   -3.43867498   -3.43868452   -3.86166433   -
3.44357471
    -4.02504172]
 [  -2.87621332   -2.70995373   -2.70995643   -2.81925758   -
3.15231769
    -2.77377334]
 [  -2.55840422   -2.11789204   -1.9          -2.34378922   -
2.56286243
    -2.63233742]
 [  -7.41029097   -7.17570464   -7.51560832   -7.39249229   -7.8828451
    -7.79693381]
 [  -7.19303846   -6.86189404 -102.05013445   -7.30792586   -
7.72586812
    -7.84780344]
 [  -6.92330456   -6.5132156   -95.97357498   -7.03150796   -
7.48813091
    -7.49364516]
 [  -6.44211074   -6.12579511  -96.932233     -6.72059051   -
7.16244146
    -7.3246566 ]
 [  -6.25454578   -5.6953279  -100.3015843    -6.14540026   -
7.16498863
    -6.84245055]
 [  -5.79729775   -5.217031   -100.96393936   -5.82583212   -
6.73260179
    -6.59125372]
 [  -5.28841003   -4.68559     -99.5761268    -5.44465241   -
6.32199218
    -6.30457437]
 [  -4.80580847   -4.0951      -94.76464298   -4.93926289   -
5.89902391
    -5.71262501]
 [  -4.22702594   -3.439       -92.14068719   -4.29797979   -
4.98887402
    -5.34795367]
 [  -3.70417474   -2.71        -92.099285     -3.71909121   -
5.07688158
    -5.12540536]
 [  -3.13344153   -1.9         -82.12921399   -3.27631824   -
4.08937028
    -4.56658303]
 [  -2.39397175   -1.65818279   -1.           -2.03162921   -
3.97803645
    -4.26268139]
 [  -7.45813417 -102.80813884   -7.61362082   -7.67430645   -
7.96189369
    -7.90170509]
 [   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.        ]]

Intra-Option Q-Learning Q-table:
[[  -7.4880433    -7.48174397   -7.48284535   -7.48897755   -
7.97156307
    -8.01549091]
 [  -7.25590989   -7.25918464   -7.2558419    -7.28889025   -
7.58431895
   -50.03341047]
 [  -7.03771096   -7.01560682   -7.01113935   -7.02869809   -
7.42074467
   -42.65321011]
 [  -6.76726858   -6.73420453   -6.73254339   -6.75504848   -
7.12751838
   -42.53596252]
 [  -6.44262176   -6.41826943   -6.41463593   -6.41985286   -
6.76156391
   -34.91929102]
 [  -6.10071924   -6.06391356   -6.06273096   -6.13271203   -6.2946244
   -37.44999044]
 [  -5.73081679   -5.65617242   -5.65563743   -5.90759998   -
6.50143982
   -37.36174907]
 [  -5.23577439   -5.18976019   -5.19050587   -5.30315881   -6.0206989
   -40.85084393]
 [  -4.76922914   -4.67029784   -4.67024196   -4.77535052   -
5.66479114
   -42.94027296]
 [  -4.121201     -4.08879454   -4.08932074   -4.30596266   -
5.23531999
```

```
   -42.94505127]
 [  -3.49383633   -3.43779282   -3.43777566   -3.5311038    -
4.86606734
   -51.91100055]
 [  -2.80276134   -2.81811452   -2.7099878    -2.98828916   -
3.86170029
    -3.02643102]
 [  -7.3409209    -7.28762393   -7.29127182   -7.28441045   -
7.67225962
    -7.51043277]
 [  -7.1017214    -7.08703211   -7.08579788   -7.08417771   -
7.30925552
   -75.28577982]
 [  -6.82212586   -6.8067621    -6.8059429    -6.86250867   -
7.03310079
   -65.89115711]
 [  -6.48695933   -6.48014166   -6.47922176   -6.54696139   -
6.68398956
   -78.96224136]
 [  -6.11070069   -6.10585394   -6.10640509   -6.15587199   -
6.32591205
   -70.97762529]
 [  -5.75191847   -5.68413083   -5.68437088   -5.68757034   -
5.96679847
   -72.88861005]
 [  -5.3092105    -5.21146329   -5.21201455   -5.26675231   -
5.57796512
   -80.77059505]
 [  -4.74659183   -4.6828342    -4.68294188   -4.81486537   -
5.07199346
   -65.95869318]
 [  -4.19115327   -4.09412474   -4.0940867    -4.34471778   -
4.46288289
   -69.29170423]
 [  -3.44493295   -3.43879446   -3.43878709   -3.59844493   -
3.84098056
   -69.07232025]
 [  -3.13979482   -2.70996517   -2.70996819   -3.29634748   -
3.55052134
   -73.7849474 ]
 [  -2.59911046   -2.17672698   -1.9          -2.05813061   -
2.99375193
    -1.9        ]
 [  -7.36807664   -7.17570464   -7.62563897   -7.40440858   -
7.60805228
    -7.62563897]
 [  -7.23474326   -6.86189404  -95.91972416   -7.21757938   -
7.41059079
   -95.91972416]
 [  -6.88578582   -6.5132156   -84.05228256   -6.93864666   -
7.07647347
   -84.05228256]
 [  -6.61350844   -6.12579511 -100.32226671   -6.63280146   -
6.79921251
  -100.32226671]
 [  -6.22066966   -5.6953279   -81.64373427   -6.27208663   -
6.41099012
   -81.64373427]
 [  -5.81828506   -5.217031    -92.10876009   -5.94601286   -6.04751
   -92.10876009]
 [  -5.42862678   -4.68559     -95.91705421   -5.41609332   -
5.71276469
   -95.91705421]
 [  -4.73320671   -4.0951      -79.26378706   -4.9858518    -
5.02443602
   -79.26378706]
 [  -4.36803303   -3.439       -81.95734563   -4.55690529   -
4.65326965
   -81.95734563]
 [  -3.86994607   -2.71        -84.28499166   -3.83049416   -
4.21738154
   -84.28499166]
 [  -3.22904253   -1.9         -88.79899311   -3.03390363   -
3.64678187
   -88.79899311]
 [  -2.36565361   -1.70475575   -1.           -2.57697717   -
2.91434468
    -1.        ]
 [  -7.45813417  -91.70307061   -7.66019164   -7.65079119   -
7.84724704
    -7.82955103]
 [   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.
```

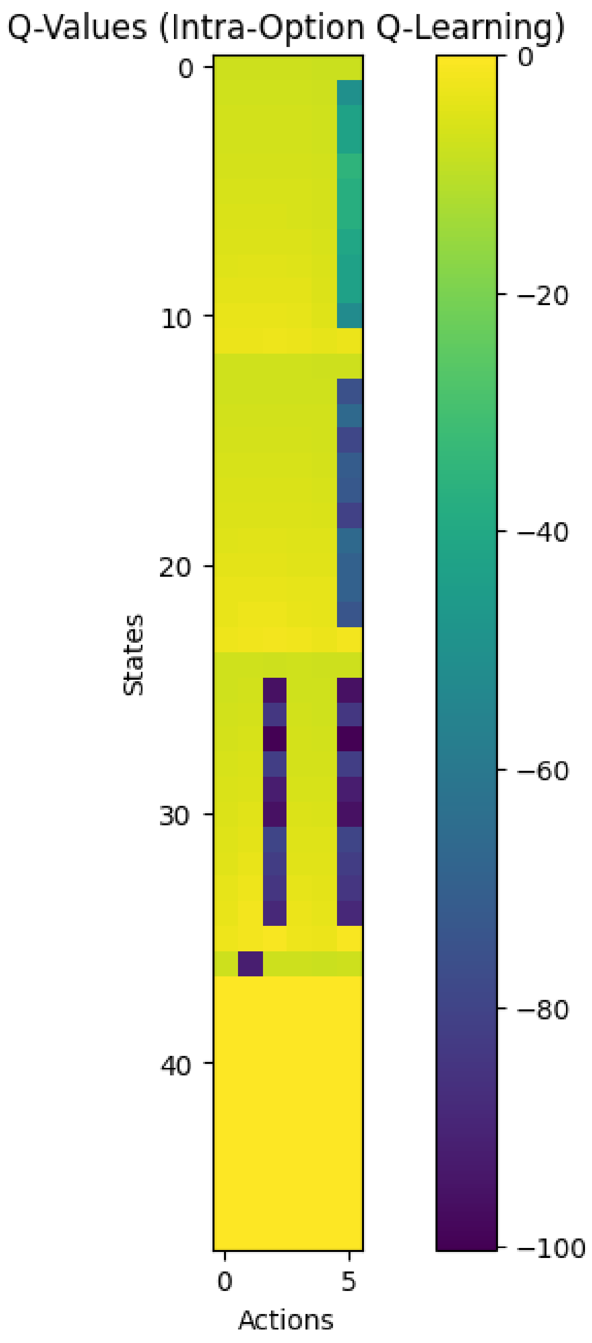
```
     0.         ]
 [   0.            0.            0.            0.            0.
     0.         ]
 [   0.            0.            0.            0.            0.
     0.         ]]

Comparison of update frequencies:
SMDP Q-Learning Update Frequency:
[[ 139.  211.  297.  140.  139.  139.]
 [ 132.  230.  270.  105.  133.  132.]
 [ 124.  237.  257.   96.  124.  124.]
 [ 116.  239.  243.   88.  114.  115.]
 [ 105.  235.  225.   82.  104.  105.]
 [  94.  227.  210.   74.   95.   94.]
 [  85.  218.  191.   65.   84.   85.]
 [  74.  201.  173.   58.   75.   74.]
 [  65.  174.  156.   53.   64.   63.]
 [  55.  146.  141.   46.   53.   53.]
 [  42.  101.  129.   37.   42.   46.]
 [  34.   37.  161.   29.   35.   33.]
 [  88.  198.  141.  132.   57.   57.]
 [  84.  223.  157.   99.   53.   53.]
 [  78.  231.  160.   89.   51.   51.]
 [  72.  225.  155.   82.   47.   47.]
 [  64.  225.  150.   73.   41.   41.]
 [  61.  217.  142.   65.   39.   38.]
 [  52.  202.  134.   58.   34.   36.]
 [  44.  186.  128.   50.   29.   31.]
 [  40.  168.  123.   42.   24.   26.]
 [  31.  144.  120.   38.   18.   22.]
 [  23.  121.  126.   27.   16.   14.]
 [  20.   24.  238.   22.   13.   13.]
 [ 145. 1121.   97.  140.   54.   51.]
 [ 112. 1024.   30.   97.   51.   57.]
 [  98.  967.   22.   93.   46.   46.]
 [  81.  917.   23.   88.   40.   45.]
 [  79.  873.   27.   72.   46.   37.]
 [  69.  841.   28.   67.   38.   35.]
 [  58.  816.   26.   63.   33.   34.]
 [  51.  805.   21.   55.   30.   27.]
 [  43.  813.   19.   45.   21.   25.]
 [  37.  827.   19.   38.   25.   26.]
 [  32.  847.   14.   39.   17.   22.]
 [  24.   23. 1000.   18.   19.   24.]
 [1230.   32.  146.  155.   49.   47.]
 [   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.]]

Intra-Option Q-Learning Update Frequency:
[[ 138.  209.  151.  138.  207.  227.]
 [ 129.  220.  142.  106.  167.  174.]
 [ 122.  225.  140.   94.  163.  170.]
 [ 113.  220.  135.   86.  149.  159.]
 [ 103.  216.  131.   78.  135.  169.]
 [  94.  197.  125.   71.  119.  152.]
 [  85.  201.  120.   67.  132.  156.]
 [  74.  186.  111.   56.  111.  138.]
 [  65.  162.  105.   49.   94.  130.]
 [  53.  132.  104.   43.   88.  132.]
 [  43.  103.  107.   34.   73.  139.]
 [  33.   33.  136.   30.   54.  167.]
 [ 131.  196.  105.  130.  131.  105.]
 [ 117.  225.  120.   98.  117.  120.]
 [ 104.  227.  124.   88.  104.  124.]
 [  93.  229.  124.   80.   93.  124.]
 [  82.  219.  120.   72.   82.  120.]
 [  74.  211.  116.   64.   74.  116.]
 [  65.  199.  114.   57.   65.  114.]
 [  56.  181.  110.   50.   56.  110.]
 [  47.  167.  109.   43.   47.  109.]
 [  36.  151.  114.   34.   36.  114.]
 [  31.  124.  122.   31.   31.  122.]
 [  24.   25.  232.   19.   24.  232.]
 [ 201. 1067.  100.  142.  201.  100.]
 [ 145.  980.   22.   90.  145.   22.]
 [ 115.  937.   15.   87.  115.   15.]
 [ 104.  891.   27.   82.  104.   27.]
 [  91.  872.   14.   76.   91.   14.]
 [  81.  858.   19.   72.   81.   19.]
 [  72.  822.   22.   61.   72.   22.]
 [  57.  826.   13.   57.   57.   13.]
 [  52.  821.   14.   55.   52.   14.]
 [  46.  829.   15.   41.   46.   15.]
 [  38.  855.   17.   30.   38.   17.]
 [  24.   25. 1000.   33.   24. 1000.]
 [1228.   19.  152.  150. 1228.  152.]
 [   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.]]
```

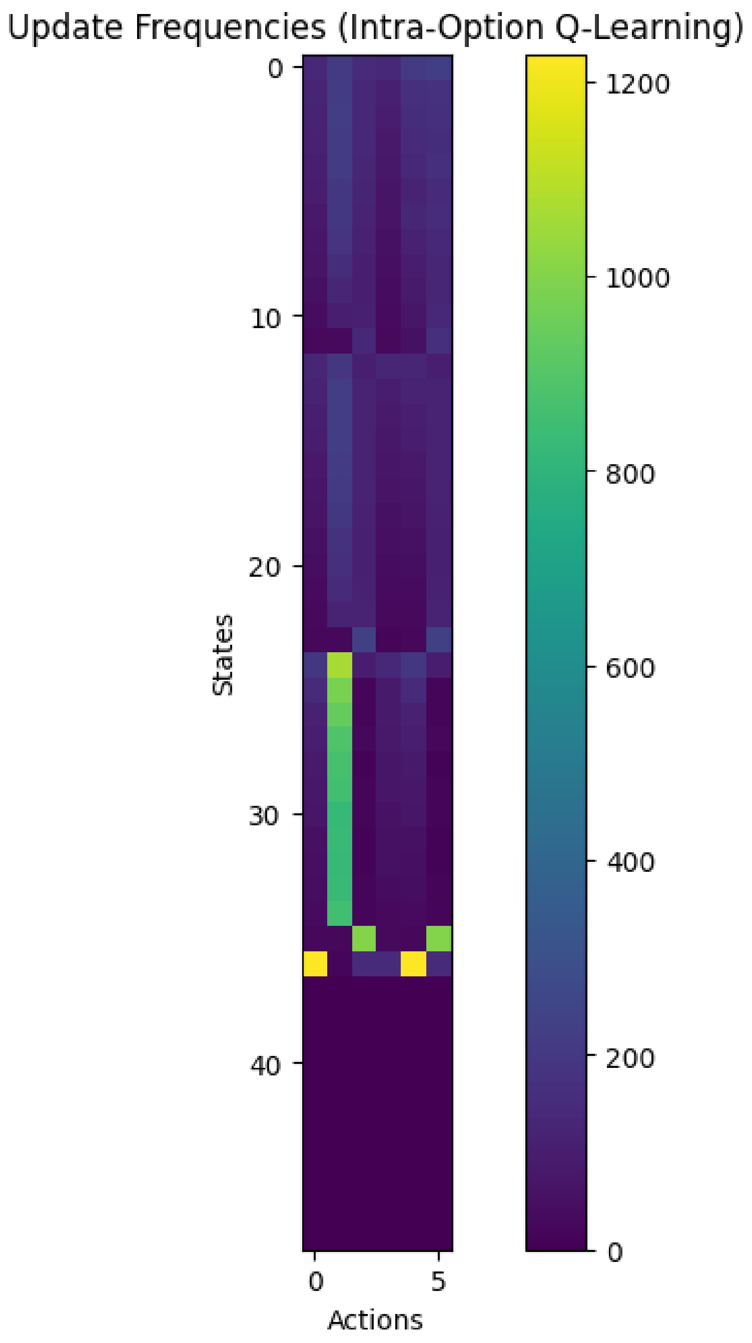
```python
import matplotlib.pyplot as plt

# Function to plot heatmap
def plot_heatmap(data, title):
    plt.figure(figsize=(10, 8))
    plt.imshow(data, cmap='viridis', interpolation='nearest')
    plt.title(title)
    plt.colorbar()
    plt.xlabel('Actions')
    plt.ylabel('States')
    plt.show()

# Plot heatmaps for Q-values and update frequencies for SMDP Q-
Learning
plot_heatmap(q_values_smdp, 'Q-Values (SMDP Q-Learning)')
print('\n')
plot_heatmap(update_frequency_smdp, 'Update Frequencies (SMDP Q-
Learning)')
```


Q-Values (SMDP Q-Learning)
States
Actions
0
10
20
30
40
0
5
0
−20
−40
−60
−80
−100



```
# Plot heatmaps for Q-values and update frequencies for Intra-Option
Q-Learning
plot_heatmap(q_values_intra, 'Q-Values (Intra-Option Q-Learning)')
print('\n')
plot_heatmap(update_frequency_intra, 'Update Frequencies (Intra-Option
Q-Learning)')
```


Q-Values (Intra-Option Q-Learning)
States
Actions
0
10
20
30
40
0
5
0
−20
−40
−60
−80
−100



Use this text cell for your comments - Task 4

Inference

1. **Q-Value Comparison**:
   - The Q-tables obtained from SMDP Q-Learning and Intra-Option Q-Learning differ significantly.

    - In SMDP Q-Learning, the Q-values are generally higher, with more negative values compared to Intra-Option Q-Learning.
    - In Intra-Option Q-Learning, there are extremely negative values (e.g., -50.03, -42.65) indicating potentially catastrophic actions or terminal states.
2. **Update Frequency Comparison**:
    - The update frequencies for both approaches also differ notably.
    - In SMDP Q-Learning, there is a higher update frequency for many state-action pairs compared to Intra-Option Q-Learning.
    - In Intra-Option Q-Learning, the update frequencies are generally lower, possibly indicating slower convergence or exploration in certain regions of the state-action space.
3. **Overall Comparison**:
    - SMDP Q-Learning tends to update Q-values more frequently, potentially leading to faster convergence, but it may also be prone to overfitting or instability.
    - Intra-Option Q-Learning appears to have less aggressive updates, which might result in smoother learning but could also lead to slower convergence or suboptimal policies if exploration is insufficient.