Tutorial5_new

April 7, 2022

1 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.

1.0.1 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.

[10]:
```
'''
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
```

[11]:
```
'''
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()
```

```python
#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.

[15]:
```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):
```

```python
    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]
'''
```

[15]: '\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'

2 Task 1

Complete the code cell below

[40]:
```python
#Q-Table: (States x Actions) === (env.ns(48) x total actions(6))
q_values_SMDP = np.zeros((48,6))

#Update_Frequency Data structure? Check TODO 4

update_frequency = np.zeros((48, 6))

seed = 42
```

```python
rg = np.random.RandomState(seed)

# TODO: epsilon-greedy action selection function
def egreedy_policy(q_values,state,epsilon):
  if not q_values[state].any() or rg.rand() < epsilon:
        return rg.choice(q_values.shape[-1])
  else:
      return np.argmax(q_values[state])
```

3 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)

[41]:
```python
#### SMDP Q-Learning

# Add parameters you might need here
gamma = 0.9
alpha = 0.4
q_values = np.zeros((48,6))
update_freq_smdp = np.zeros((48, 6))

# Iterate over 1000 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, state, epsilon=0.1)

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

            state_next, reward, done , _= env.step(action)
            action_next = np.argmax(q_values[state_next])
            q_values[state, action] += alpha*(reward +␣
↪gamma*q_values[state_next, action_next] - q_values[state, action])
            update_freq_smdp[state, action] += 1

        # Checking if action chosen is an option
```

```
        reward_bar = 0
        if action == 4: # action => Away option

            state_t = state  # state at whcih option was selected
            tau = 0          # Time taken for option to terminate

            optdone = False
            while (optdone == False):

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

                tau = tau + 1 # Increase the time

                # Is this formulation right? What is this term?
                # reward_bar = gamma*reward_bar + reward # this is wrong
                reward_bar += (gamma**tau) * reward

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

                state = next_state

            action_next = np.argmax(q_values[state])
            q_values[state_t, action] += alpha*(reward_bar + (gamma**tau)*q_values[state, action_next] - q_values[state_t, action])
            update_freq_smdp[state_t, action] += 1

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

            state_t = state  # state at whcih option was selected
            tau = 0          # Time taken for option to terminate

            optdone = False
            while (optdone == False):

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

                tau = tau + 1 # Increase the time

                # Is this formulation right? What is this term?
                #reward_bar = gamma*reward_bar + reward
                reward_bar += (gamma**tau) * reward
```

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

                state = next_state

            action_next = np.argmax(q_values[state])
            q_values[state_t, action] += alpha*(reward_bar +␣
↪(gamma**tau)*q_values[state, action_next] - q_values[state_t, action])
            update_freq_smdp[state_t, action] += 1

q_values_smdp = q_values
```

```
100%|      | 1000/1000 [00:22<00:00, 43.57it/s]
```

4 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)

[42]:
```
#### Intra-Option Q-Learning
#### SMDP Q-Learning

# Add parameters you might need here
gamma = 0.9
alpha = 0.4
q_values = np.zeros((48,6))
update_freq_intra = np.zeros((48, 6))
# Iterate over 1000 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, state, epsilon=0.1)

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

            state_next, reward, done , _= env.step(action)
            action_next = np.argmax(q_values[state_next])
```

```
            q_values[state, action] += alpha*(reward + gamma*q_values[state_next, action_next] - q_values[state, action])
            update_freq_intra[state, action ] += 1

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

            optdone = False
            while (optdone == False):

                optact,optdone = Away(env,state)
                next_state, reward, done, _ = env.step(optact)


                if ( not optdone ):  # if not terminating
                  q_values[state, action] += alpha*(reward + gamma*q_values[next_state, action] - q_values[state, action])
                  update_freq_intra[state, action] += 1
                else: # if terminating
                  max_action = np.argmax(q_values[state_next])
                  q_values[state, action] += alpha*(reward + gamma*q_values[next_state, max_action] - q_values[state, action])
                  update_freq_intra[state, action] += 1

                state = next_state


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

            optdone = False
            while (optdone == False):

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

                if ( not optdone ):
                  q_values[state, action] += alpha*(reward + gamma*q_values[next_state, action] - q_values[state, action])
                  update_freq_intra[state, action] += 1
                else:
                  max_action = np.argmax(q_values[state_next])
                  q_values[state, action] += alpha*(reward + gamma*q_values[next_state, max_action] - q_values[state, action])
                  update_freq_intra[state, action] += 1

                state = next_state
```

```python
q_values_intra = q_values
```

```
100%|          | 1000/1000 [00:35<00:00, 28.12it/s]
```

5 Task 4

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

[43]:
```python
# Use this cell for Task 4 Code
import pprint
import pandas as pd
pp = pprint.PrettyPrinter(indent=4)
pd.DataFrame(q_values_smdp.T)
```

[43]:
```
          0         1         2          3          4          5          6  \
0 -3.691877 -8.449851 -5.867800  -7.087039  -5.874626  -4.170851  -4.610943
1 -3.143860 -2.248663 -3.657367  -2.892480  -2.313573  -2.379811  -2.318276
2 -4.052660 -9.600215 -1.825237 -27.952182 -27.786626 -12.893277 -44.218248
3 -4.104565 -7.519756 -6.817036  -7.134509  -4.764477  -7.136071  -6.304913
4 -3.463399 -6.346188 -6.311557  -4.991570  -7.083544  -3.876638  -3.771415
5 -3.108303 -4.303297 -7.036353  -5.580064  -3.717082  -3.950565  -6.231592

          7          8          9  …   38   39   40   41   42   43   44  \
0 -6.279650  -6.343141  -6.106495  …  0.0  0.0  0.0  0.0  0.0  0.0  0.0
1 -2.725843  -1.185866  -3.645733  …  0.0  0.0  0.0  0.0  0.0  0.0  0.0
2 -7.759842 -11.936703 -17.410918  …  0.0  0.0  0.0  0.0  0.0  0.0  0.0
3 -6.241036  -3.562310  -4.955121  …  0.0  0.0  0.0  0.0  0.0  0.0  0.0
4 -6.172926  -3.582149  -3.690065  …  0.0  0.0  0.0  0.0  0.0  0.0  0.0
5 -3.451363  -3.720931  -4.491591  …  0.0  0.0  0.0  0.0  0.0  0.0  0.0

    45   46   47
0  0.0  0.0  0.0
1  0.0  0.0  0.0
2  0.0  0.0  0.0
3  0.0  0.0  0.0
4  0.0  0.0  0.0
5  0.0  0.0  0.0

[6 rows x 48 columns]
```

[44]:
```python
pd.DataFrame(q_values_intra.T)
```

[44]:
```
          0          1          2          3         4          5          6  \
0 -8.189464  -8.782674  -8.681404  -8.549958 -8.173937  -8.376694  -8.426642
1 -6.446833  -8.939067 -11.616702 -12.155948 -1.045178 -10.296436  -8.448870
```

```
2 -8.590815 -46.230819  -9.410944 -46.004230 -6.012437 -10.205005 -46.033860
3 -5.989878  -8.577999  -8.599909  -8.528593 -5.616082  -7.038119  -6.722718
4 -8.753571  -8.594196  -8.728136  -8.549959 -8.017029  -9.304435  -8.434722
5 -8.700649  -8.785667  -8.681732  -8.673946 -8.694789  -8.616726  -8.435278

          7         8          9  …   38   39   40   41   42   43   44  \
0  -7.533589 -7.921744  -8.485203  …  0.0  0.0  0.0  0.0  0.0  0.0  0.0
1 -19.982907 -7.893445 -21.553575  …  0.0  0.0  0.0  0.0  0.0  0.0  0.0
2 -44.337327 -9.765647  -9.877353  …  0.0  0.0  0.0  0.0  0.0  0.0  0.0
3  -8.096110 -8.122719  -7.824842  …  0.0  0.0  0.0  0.0  0.0  0.0  0.0
4  -7.569324 -7.923458  -8.458760  …  0.0  0.0  0.0  0.0  0.0  0.0  0.0
5  -8.678885 -8.138686  -8.493774  …  0.0  0.0  0.0  0.0  0.0  0.0  0.0

    45   46   47
0  0.0  0.0  0.0
1  0.0  0.0  0.0
2  0.0  0.0  0.0
3  0.0  0.0  0.0
4  0.0  0.0  0.0
5  0.0  0.0  0.0

[6 rows x 48 columns]
```

[45]:
```
pd.DataFrame(update_freq_smdp.T)
```

[45]:
```
         0        1       2       3       4       5       6       7       8  \
0  17861.0   4791.0  1516.0   951.0  1201.0  1094.0   812.0   794.0   875.0
1  45960.0  14826.0  4976.0  3663.0  3572.0  3130.0  2531.0  2271.0  2050.0
2  94108.0   1781.0   458.0   239.0   251.0   309.0   257.0   190.0   190.0
3  23422.0  18254.0  6920.0  4594.0  3690.0  5584.0  5384.0  3659.0  3415.0
4  37747.0   5546.0  1350.0   930.0   859.0   904.0   708.0   530.0   491.0
5  37398.0   5602.0  1401.0   850.0   787.0   917.0   674.0   547.0   564.0

        9  …   38   39   40   41   42   43   44   45   46   47
0   466.0  …  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
1  2347.0  …  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
2   165.0  …  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
3  2418.0  …  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
4   434.0  …  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
5   395.0  …  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0

[6 rows x 48 columns]
```

[46]:
```
pd.DataFrame(update_freq_intra.T)
```

[46]:
```
         0        1       2       3       4       5       6       7       8  \
0  51213.0   6014.0  2889.0  1958.0  1870.0  1397.0  1533.0  1231.0  1857.0
```

```
1   68500.0  16485.0   6041.0  5389.0  3216.0  4205.0  2738.0  2671.0  2954.0
2  155686.0   1933.0    740.0   526.0   439.0   282.0   385.0   378.0   476.0
3   62065.0  22862.0  11730.0  8141.0  8823.0  6087.0  7757.0  6279.0  7322.0
4   30937.0   3291.0   1269.0   854.0   713.0   677.0   678.0   626.0   632.0
5   33221.0   3233.0   1284.0   879.0   789.0   630.0   698.0   545.0   511.0

       9  …   38   39   40   41   42   43   44   45   46   47
0 1214.0  …  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
1 2596.0  …  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
2  322.0  …  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
3 5767.0  …  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
4  575.0  …  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
5  454.0  …  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0

[6 rows x 48 columns]
```

Use this text cell for your comments - Task 4

It can be clearly seen that Intra Q Learning makes more updtes compared to SMDP to the state action pairs (Infeered from the update frequency table). This leads to effective usage of sample by Intra Q learning.

6 (IGNORE)

[ ]:
```
!sudo apt-get install texlive-xetex texlive-fonts-recommended
 ↪texlive-plain-generic
```

[ ]:
```
# Run this only if you are using Google Colab
from google.colab import drive
import os

drive.mount('/content/drive')
```

[ ]:
```
!jupyter nbconvert --to pdf /content/drive/MyDrive/Documents/Sem6-drive/RL/
 ↪Tutorial/5Tut/Tutorial5_new.ipynb
```

[ ]:
```
```