import gym

import random

#Red — 0 , Green — 1, Yellow — 2, and Blue — 3 for pick up

streets = gym.make("Taxi-v3").env #New versions keep getting released; if -v3 doesn't work, try -v2 or -v4

streets.render()

--------------------------------------------------------------------------------

##Red — 0 , Green — 1, Yellow — 2, and Blue — 3 for pick up

#Each state is defined by a 4 entries tuple: （taxi\_row, taxi\_col, passenger\_location, destination)

initial\_state = streets.encode(2, 1, 3, 1)

streets.s = initial\_state

streets.render()

#State Space: 25 possible taxi positions, 5 possible locations of the passenger

# 25\*5\*4 = 500

--------------------------------------------------------------------------------

import numpy as np

#Action space:6 --> N,S,E,W, DROP-OFF, PICKUP

#Rewards: CORRECT FINAL DEST. +20, STEP -1, INCORRECT PICK/DROP -10

q\_table = np.zeros([streets.observation\_space.n, streets.action\_space.n]) # 500 , 6

# a 2D array that represent every possible state and action in the virtual space and initialize all of them to 0

G = 0

learning\_rate = 0.1

discount\_factor = 0.5

exploration = 0.1

epochs = 500

for taxi\_run in range(epochs): #Start training (the agent plays the number of epochs)

state = streets.reset()

done = False

G = 0

steps=0

while not done:#each epoch/play contains this number of actions, starting from pickup a passenger until drop-off

steps +=1

random\_value = random.uniform(0, 1)

if (random\_value < exploration):

action = streets.action\_space.sample() # Explore a random action

else:

action = np.argmax(q\_table[state]) # Return the action with the highest q-value

next\_state, reward, done, info = streets.step(action) # Do the above action

prev\_q = q\_table[state, action]

next\_max\_q = np.max(q\_table[next\_state])

# see RL-2 PPT file --- slide# 5

new\_q = (1 - learning\_rate) \* prev\_q + learning\_rate \* (reward + discount\_factor \* next\_max\_q)

G += reward

q\_table[state, action] = new\_q

#streets.render()

state = next\_state

--------------------------------------------------------------------------------

from IPython.display import clear\_output

from time import sleep

lengths=[]

for tripnum in range(1, 11):

state = streets.reset()

done = False

trip\_length = 0

while not done and trip\_length < 25:

action = np.argmax(q\_table[state])

next\_state, reward, done, info = streets.step(action)

clear\_output(wait=True)

print("Trip number " + str(tripnum) + " Step " + str(trip\_length))

print(streets.render(mode='ansi'))

sleep(.2)

state = next\_state

trip\_length += 1

lengths.append(trip\_length)

sleep(.2)

avg\_len=sum(lengths)/10

print(avg\_len)

--------------------------------------------------------------------------------