CS 559 - Fall 2021
Homework #8: [Q-learning]

Krishna Garg
kgarg8@uic.edu

1 Environment, States and Actions

```
#### Environment ####

# HO 21 22 23 24
# 15 16 17 18 19
# 10 11 12 13 14
#  5  6  7  8  9
#  I  1  2  3 GM

# GM: Gold Mine (4)
# HO: Home (20)
#  I: Initial Pos (0)

######################
```

GRID is of size 5*5. In addition to these 25 locations, we store even the gold collected at each location as a state property. We consider the STATE in a 3-tuple form: (x, y, gold), otherwise Q-learning was not converging and also, it is justified because every time we reach a location (x, y), the possible actions will depend upon the gold collected at that point of time. In that sense, we have 5*5*4=100 states in the Q-table.

We have 4 ACTIONS: {'R', 'L', 'U', 'D'} - 4 directions.

2 Pure Exploitation

We keep "exploration prob" as 0 for pure exploitation.

Other hyperparameters:

$\alpha = 0.1$

$\gamma = 0.9$

num_episodes = 100

train_timesteps = 5000

test_timesteps = 500

(i) The policy learned by the algorithm is incorrect, it tries to go 'D' from the initial state. In fact, the action remains the same for any number of timesteps.

(ii) The cumulative reward of the policy is always 0 for all the episodes.

(iii) It is definitely not the optimal policy because the miner never learns to collect the gold and reach the home. It just tries some random action which has practically no meaning and worth.

3 Exploration prob=0.1, γ=0.9

(i) The miner learns the policy to collect 3 golds one after the another and then reach the home and come back again to collect the gold and so on.

In short, the first 40 actions can be summarized as: **URRRDRRRLLUULUURRRDDDRRRL-LUULUURRRDDDRRR**.

We describe the 40 timesteps (after the training completes) in more detail below:

```
Timestep: 1, Cur state: 0, Action: U, New state: 5, Reward: 0, Gold: 0
Timestep: 2, Cur state: 5, Action: R, New state: 6, Reward: 0, Gold: 0
Timestep: 3, Cur state: 6, Action: R, New state: 7, Reward: 0, Gold: 0
Timestep: 4, Cur state: 7, Action: R, New state: 8, Reward: 0, Gold: 0
Timestep: 5, Cur state: 8, Action: D, New state: 3, Reward: 0, Gold: 0
Timestep: 6, Cur state: 3, Action: R, New state: 3, Reward: 0, Gold: 1
Timestep: 7, Cur state: 3, Action: R, New state: 3, Reward: 0, Gold: 2
Timestep: 8, Cur state: 3, Action: R, New state: 3, Reward: 0, Gold: 3
Timestep: 9, Cur state: 3, Action: L, New state: 2, Reward: 0, Gold: 3
Timestep: 10, Cur state: 2, Action: L, New state: 1, Reward: 0, Gold: 3
Timestep: 11, Cur state: 1, Action: U, New state: 6, Reward: 0, Gold: 3
Timestep: 12, Cur state: 6, Action: U, New state: 11, Reward: 0, Gold: 3
Timestep: 13, Cur state: 11, Action: L, New state: 10, Reward: 0, Gold: 3
Timestep: 14, Cur state: 10, Action: U, New state: 15, Reward: 0, Gold: 3
Timestep: 15, Cur state: 15, Action: U, New state: 15, Reward: 3, Gold: 0
Timestep: 16, Cur state: 15, Action: R, New state: 16, Reward: 0, Gold: 0
Timestep: 17, Cur state: 16, Action: R, New state: 17, Reward: 0, Gold: 0
Timestep: 18, Cur state: 17, Action: R, New state: 18, Reward: 0, Gold: 0
Timestep: 19, Cur state: 18, Action: D, New state: 13, Reward: 0, Gold: 0
Timestep: 20, Cur state: 13, Action: D, New state: 8, Reward: 0, Gold: 0
Timestep: 21, Cur state: 8, Action: D, New state: 3, Reward: 0, Gold: 0
Timestep: 22, Cur state: 3, Action: R, New state: 3, Reward: 0, Gold: 1
Timestep: 23, Cur state: 3, Action: R, New state: 3, Reward: 0, Gold: 2
Timestep: 24, Cur state: 3, Action: R, New state: 3, Reward: 0, Gold: 3
Timestep: 25, Cur state: 3, Action: L, New state: 2, Reward: 0, Gold: 3
Timestep: 26, Cur state: 2, Action: L, New state: 1, Reward: 0, Gold: 3
Timestep: 27, Cur state: 1, Action: U, New state: 6, Reward: 0, Gold: 3
Timestep: 28, Cur state: 6, Action: U, New state: 11, Reward: 0, Gold: 3
Timestep: 29, Cur state: 11, Action: L, New state: 10, Reward: 0, Gold: 3
Timestep: 30, Cur state: 10, Action: U, New state: 15, Reward: 0, Gold: 3
Timestep: 31, Cur state: 15, Action: U, New state: 15, Reward: 3, Gold: 0
Timestep: 32, Cur state: 15, Action: R, New state: 16, Reward: 0, Gold: 0
Timestep: 33, Cur state: 16, Action: R, New state: 17, Reward: 0, Gold: 0
Timestep: 34, Cur state: 17, Action: R, New state: 18, Reward: 0, Gold: 0
Timestep: 35, Cur state: 18, Action: D, New state: 13, Reward: 0, Gold: 0
Timestep: 36, Cur state: 13, Action: D, New state: 8, Reward: 0, Gold: 0
Timestep: 37, Cur state: 8, Action: D, New state: 3, Reward: 0, Gold: 0
Timestep: 38, Cur state: 3, Action: R, New state: 3, Reward: 0, Gold: 1
Timestep: 39, Cur state: 3, Action: R, New state: 3, Reward: 0, Gold: 2
Timestep: 40, Cur state: 3, Action: R, New state: 3, Reward: 0, Gold: 3
```

(ii) The maximum cumulative reward of the policy of the optimal policy = **0.842**.

Initially, the model starts to learn the optimal policy and the cumulative reward starts increasing from 0 and reaches its maximum value in 30 episodes, after which it becomes steady.

(iii) Yes, this is the optimal policy since this is the desired objective of the game. This is what maximizes the reward for the given gamma.

4 Exploration prob=0.1, γ=0.6

(i) The miner again learns the policy to first collect 3 golds from the gold mine, then go to home straight and then come back to collect golds and so on.

The first 40 actions after the training can be summarized as **URRRDRRRLLUULUURDRDRDR-RRLLUULUURDRDRDRRR**.

```
Timestep: 1, Cur state: 0, Action: U, New state: 5, Reward: 0, Gold: 0
Timestep: 2, Cur state: 5, Action: R, New state: 6, Reward: 0, Gold: 0
Timestep: 3, Cur state: 6, Action: R, New state: 7, Reward: 0, Gold: 0
Timestep: 4, Cur state: 7, Action: R, New state: 8, Reward: 0, Gold: 0
Timestep: 5, Cur state: 8, Action: D, New state: 3, Reward: 0, Gold: 0
Timestep: 6, Cur state: 3, Action: R, New state: 3, Reward: 0, Gold: 1
Timestep: 7, Cur state: 3, Action: R, New state: 3, Reward: 0, Gold: 2
Timestep: 8, Cur state: 3, Action: R, New state: 3, Reward: 0, Gold: 3
Timestep: 9, Cur state: 3, Action: L, New state: 2, Reward: 0, Gold: 3
Timestep: 10, Cur state: 2, Action: L, New state: 1, Reward: 0, Gold: 3
Timestep: 11, Cur state: 1, Action: U, New state: 6, Reward: 0, Gold: 3
Timestep: 12, Cur state: 6, Action: U, New state: 11, Reward: 0, Gold: 3
Timestep: 13, Cur state: 11, Action: L, New state: 10, Reward: 0, Gold: 3
Timestep: 14, Cur state: 10, Action: U, New state: 15, Reward: 0, Gold: 3
Timestep: 15, Cur state: 15, Action: U, New state: 15, Reward: 3, Gold: 0
Timestep: 16, Cur state: 15, Action: R, New state: 16, Reward: 0, Gold: 0
Timestep: 17, Cur state: 16, Action: D, New state: 11, Reward: 0, Gold: 0
Timestep: 18, Cur state: 11, Action: R, New state: 12, Reward: 0, Gold: 0
Timestep: 19, Cur state: 12, Action: D, New state: 7, Reward: 0, Gold: 0
Timestep: 20, Cur state: 7, Action: R, New state: 8, Reward: 0, Gold: 0
Timestep: 21, Cur state: 8, Action: D, New state: 3, Reward: 0, Gold: 0
Timestep: 22, Cur state: 3, Action: R, New state: 3, Reward: 0, Gold: 1
Timestep: 23, Cur state: 3, Action: R, New state: 3, Reward: 0, Gold: 2
Timestep: 24, Cur state: 3, Action: R, New state: 3, Reward: 0, Gold: 3
Timestep: 25, Cur state: 3, Action: L, New state: 2, Reward: 0, Gold: 3
Timestep: 26, Cur state: 2, Action: L, New state: 1, Reward: 0, Gold: 3
Timestep: 27, Cur state: 1, Action: U, New state: 6, Reward: 0, Gold: 3
Timestep: 28, Cur state: 6, Action: U, New state: 11, Reward: 0, Gold: 3
Timestep: 29, Cur state: 11, Action: L, New state: 10, Reward: 0, Gold: 3
Timestep: 30, Cur state: 10, Action: U, New state: 15, Reward: 0, Gold: 3
Timestep: 31, Cur state: 15, Action: U, New state: 15, Reward: 3, Gold: 0
Timestep: 32, Cur state: 15, Action: R, New state: 16, Reward: 0, Gold: 0
Timestep: 33, Cur state: 16, Action: D, New state: 11, Reward: 0, Gold: 0
Timestep: 34, Cur state: 11, Action: R, New state: 12, Reward: 0, Gold: 0
Timestep: 35, Cur state: 12, Action: D, New state: 7, Reward: 0, Gold: 0
Timestep: 36, Cur state: 7, Action: R, New state: 8, Reward: 0, Gold: 0
Timestep: 37, Cur state: 8, Action: D, New state: 3, Reward: 0, Gold: 0
Timestep: 38, Cur state: 3, Action: R, New state: 3, Reward: 0, Gold: 1
Timestep: 39, Cur state: 3, Action: R, New state: 3, Reward: 0, Gold: 2
Timestep: 40, Cur state: 3, Action: R, New state: 3, Reward: 0, Gold: 3
```

(ii) The maximum cumulative reward of the policy for an episode is **0.00235** after which the model becomes steady.

(iii) The model using both gamma values 0.6 & 0.9 learn the optimal policy. There is slight difference only in the way they traverse the grids to reach the gold mine from the home i.e. $\gamma = 0.6$ takes actions **RRRDDD** to reach from home to gold mine whereas $\gamma = 0.9$ takes actions **RDRDRD** to do it. Both are using the shortest possible paths. The slight difference comes due to the exploration probability factor. With $\gamma = 0.6$, the miner tends to explore more rather than always choosing the best action and ends up learning a different path trajectory. See the plots for the two gamma values to visualize the difference in terms of cumulative reward vs. episodes & timesteps curves. We observe that with $\gamma = 0.9$, the model learns the optimal policy in about 30 episodes whereas with $\gamma = 0.6$, it takes around 45 episodes.

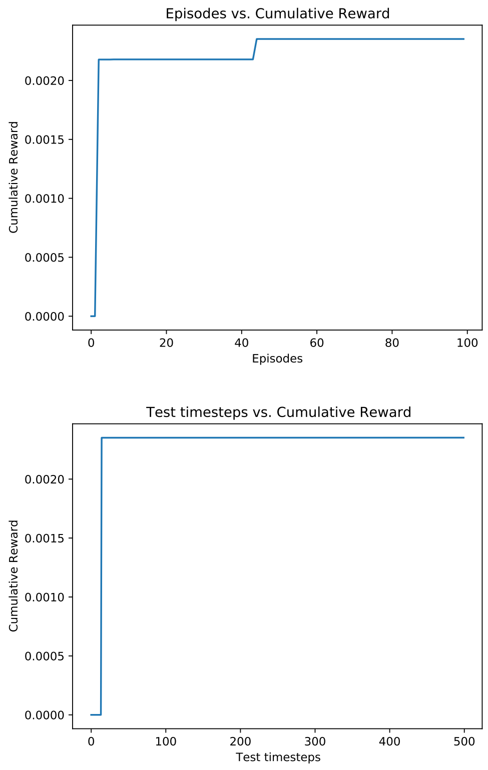

Figure 1: Plots: $\gamma = 0.6$, (a) Episodes vs. Cumulative Reward, (b) Test timesteps vs. Cumulative reward

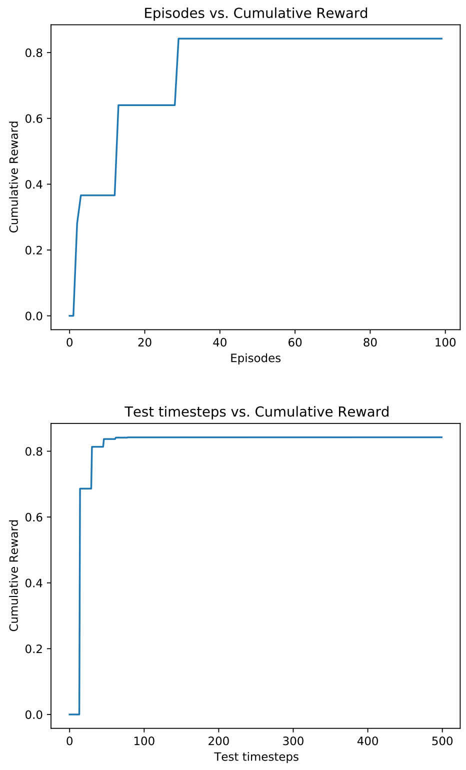

Figure 2: Plots: $\gamma = 0.9$, Episodes vs. Cumulative Reward, (b) Test timesteps vs. Cumulative reward

Code - 08-656377418-Garg.py

```
# Q-learning implementation for 5*5 grid

import numpy as np, matplotlib.pyplot as plt, random, pdb
from tqdm import tqdm

#### Environment ####

# HO 21 22 23 24
# 15 16 17 18 19
# 10 11 12 13 14
#  5  6  7  8  9
#  I  1  2  3 GM

# GM: Gold Mine (4)
# HO: Home    (20)
#  I: Initial Pos (0)

######################

# environment settings
GRID_SIZE = 5
ACTIONS = 4                 # R, L, U, D
GOLD_MAX = 3                # maximum gold value
STATES  = GRID_SIZE*GRID_SIZE
GOLD_MINE = GRID_SIZE-1     # location of gold mine
HOME    = GRID_SIZE*(GRID_SIZE-1) # location of home

# hyperparameters
num_episodes = 100
train_timesteps = 5000
test_timesteps = 500
alpha       = 0.10 # learning rate
gamma       = 0.9 # discount factor
prob        = 0.1 # probability for exploration

# seed
seed = 112
np.random.seed(seed)
random.seed(seed)

# global parameters
Q_table = np.random.normal(size=(STATES, GOLD_MAX+1, ACTIONS)) #
    Initialize with Gaussian Distribution values
action_map = {0:'R', 1:'L', 2:'U', 3:'D'}        # Actions


# returns (reward, next_state, new_gold_val)
def reward(state, action, gold):

   # next step is GOLD MINE
   if (state==GOLD_MINE-1 and action==0) or (state==GOLD_MINE+GRID_SIZE
       and action==3):
      if gold < GOLD_MAX:
         gold = gold + 1
      return (0, state, gold) # zero reward, stay in the same state

   # next step is HOME
   elif (state==HOME+1 and action==1) or (state==HOME-GRID_SIZE and
       action==2):
      ret = (gold, state, 0) # reward equals gold and gold gets unloaded
      gold = 0
      return ret # non-zero reward, stay in the same state
   # RIGHT
   elif action == 0:
      if state%GRID_SIZE == GRID_SIZE-1:
         return (0, state, gold) # no way to go
      else:
         return (0, state+1, gold)

   # LEFT
   elif action == 1:
      if state%GRID_SIZE == 0:
         return (0, state, gold) # no way to go
      else:
         return (0, state-1, gold)

   # UP
   elif action == 2:
      if state//GRID_SIZE == GRID_SIZE-1:
         return (0, state, gold) # no way to go
      else:
         return (0, state+GRID_SIZE, gold)

   # DOWN
   else:
      if state//GRID_SIZE == 0:
         return (0, state, gold) # no way to go
      else:
         return (0, state-GRID_SIZE, gold)


def plot(rewards, label):
   if label == 'Test timesteps':
      arr = [i for i in range(test_timesteps)]
   else: # 'Episodes'
      arr = [i for i in range(num_episodes)]
   plt.figure()
   plt.xlabel(label)
   plt.ylabel('Cumulative Reward')
   plt.plot(arr, rewards)
   plt.title('{} vs. Cumulative Reward'.format(label))
   plt.savefig('{} vs. Cumulative Reward_gamma_{}.pdf'.format(label,
       gamma))


def episodic_test(episode):
   x = 0; gold = 0; cum_reward = 0 # initial state
   for t in range(test_timesteps):
      a = np.argmax(Q_table[(x, gold)])
      r, y, gold = reward(x, a, gold)
      x = y
      cum_reward += r*(gamma**t)
   print('Episode: {}, Cumulative Reward: {}'.format(episode,
       cum_reward))
   return cum_reward


def train():
   rewards = []
   for e in range(num_episodes):
      x = 0; gold = 0 # initial state
      for t in range(train_timesteps):
         a = random.random()
         if a < prob:
            a = random.randint(0, 3)  # exploration
         else:
            a = np.argmax(Q_table[(x, gold)]) # exploitation
```

```
            prev_state = (x, gold)
            r, y, gold = reward(x, a, gold)

            a_Qmax = np.argmax(Q_table[(y, gold)])
            Q_table[prev_state][a] = (1-alpha)*Q_table[prev_state][a] +
                alpha*(r + gamma*Q_table[(y, gold)][a_Qmax])

            # logging
            # print('Cur state: {}, Action: {}, New state: {}, Reward: {},
                Gold: {}'.format(x, action_map[a], y, r, gold))
            # if r != 0: print(t)

            x = y # save new state

        episodic_reward = episodic_test(e)
        rewards.append(episodic_reward)
    plot(rewards, 'Episodes')


def test():
    x = 0; gold = 0; cum_reward = 0 # initial state
    rewards = []
    for t in range(test_timesteps):
        a = np.argmax(Q_table[(x, gold)])
        r, y, gold = reward(x, a, gold)

        # logging
        print('Timestep: {}, Cur state: {}, Action: {}, New state: {},
            Reward: {}, Gold: {}'.format(t+1, x, action_map[a], y, r,
            gold))
        # if r != 0: print(t)

        x = y
        cum_reward += r*(gamma**t)
        rewards.append(cum_reward)

    plot(rewards, 'Test timesteps')


train()
test()
```