CZ3005 Artificial Intelligence

Assignment 2: Reinforcement Learning

Kannan Shivani

U1822998H

Description of algorithm

Q-learning algorithm was used for the MDP.

Brief explanation of the agent

Agent named QAgent was created, with 4 main functions:

- Initialize
  - Initialization of agent with appropriate variables and values
- Take_action
  - Epsilon greedy function is used to determine what action to take
- Train
  - Updates the Q-table values based on the current state, action and reward.
- Display Q-table
  - Saves the resulting Q-table values to a csv file

1) Initialize

```python
def __init__(self):
    self.action_space = ['left','right','forward','backward','up','down'] # in TreasureCube
    row_names = [str(z)+str(x)+str(y) for z in range(4) for x in range(4) for y in range(4)]
    self.Q = pd.DataFrame([[0]*len(self.action_space) for i in range(4*4*4)], index=row_names, columns=self.action_space)
    #Parameters of the Q-learning model
    self.epsilon = 0.01
    self.alpha = 0.5
    self.gamma = 0.9
```

The initialize function first initialized the Q-table as a 2D list of 0s with

rows equal to the number of states (4*4*4) and

columns corresponding to each action (len(self.action_space)).

Additionally, the parameters required for Q-learning are also initialized:

- Discount factor (γ), gamma=0.9
- Learning rate (α), alpha=0.5
- Exploration rate (ϵ), epsilon=0.01

2)Take_action

```
def take_action(self, state):
    #using epsilon greedy function to decide action
    optimal_a_index = self.action_space.index(self.Q.loc[state].idxmax())
    optimal_prob = 1-self.epsilon + (self.epsilon/len(self.action_space))
    non_optimal_prob = self.epsilon/len(self.action_space)
    action_prob_list = len(self.action_space)*[non_optimal_prob]
    action_prob_list[optimal_a_index] = optimal_prob
    action = np.random.choice(self.action_space,p=action_prob_list)
    return action
```

Action to take is decided with an epsilon-greedy choice.

$$a^* \leftarrow \arg\max_a Q(s,a)$$

For all $a \in \mathcal{A}(s)$:

$$\pi(s,a) \leftarrow \begin{cases} 1-\varepsilon+\varepsilon/|\mathcal{A}(s)| & \text{if } a = a^* \\ \varepsilon/|\mathcal{A}(s)| & \text{if } a \neq a^* \end{cases}$$

The first step is to compute the optimal action from the Q-table, which is the action with the maximum Q-value at the current state.

The optimal action will then be chosen with a probability of $1-\varepsilon+\varepsilon/|A(s)|$ or a non-optimal action chosen with a probability of $\varepsilon/|A(s)|$, where $|A(s)|$ refers to the number of actions to take, which in this case is always 6.

However, the actual action taken may not be the one described by the probability above, as within the environment there is a probability of 0.4 of taking a different action instead.

3)Train function

```python
def train(self, state, action, next_state, reward):
    Q_old = self.Q.loc[state,action]
    Q_max = max(self.Q.loc[next_state])
    #Update Q-table!
    self.Q.loc[state,action] = Q_old+self.alpha*(reward+(self.gamma*Q_max)-Q_old)
```

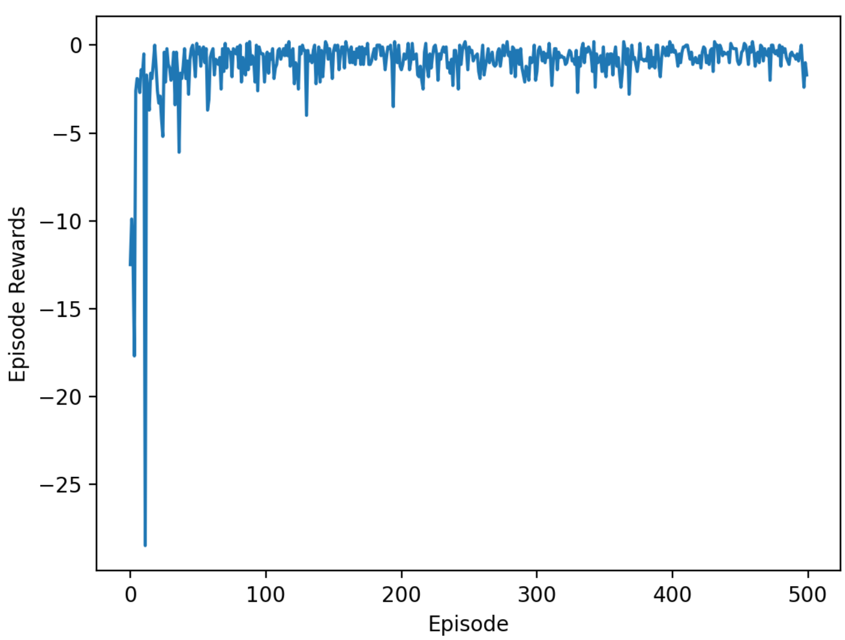
After an action is taken, the agent will arrive at a new state. The agent will update the Q-table based on this state and the action taken to get to the state. The following formula is used to perform this update:

$$Q_{new}(S_t, A_t) \leftarrow Q_{old}(S_t, A_t) + \alpha(R_{t+1} + \gamma \max_a Q_{old}(S_{t+1}, a) - Q_{old}(S_t, A_t))$$

where $\alpha = 0.5$, $\gamma = 0.9$, and R_{t+1} is the reward for each step.

Learning progress

The learning progress is plotted and saved into ‘Episode Rewards.png’



```python
plt.plot(episode_rewards)
plt.ylabel("Episode Rewards")
plt.xlabel("Episode")
plt.savefig('Episode Rewards.png')
plt.show()
```

As seen in the graph, 500 episodes was more than enough for the values to stabilize, with the rewards staying about the same since the 100th episode.

The noise in the rewards afterwards is likely due to the environment making the agent choose a random option sometimes.

4)Display Q table

The following code below was used to save the Q-table results into a .csv file named Q_table.csv. The Q-table index must be re-converted into the state numbers.

```python
def display_Qtable(self):
    pd.set_option('display.max_rows',65)
    print(self.Q)
    self.Q.to_csv('Q_table.csv')
```

The Q-table is as follows:

	left	**right**	**forward**	**backward**	**up**	**down**
0	-0.5646952447260390	-0.5034621633189430	-0.5949249996992070	-0.5919564953542800	-0.5688939919765390	-0.6061271840452880
1	-0.5613618629414810	-0.5473238218194700	-0.54451513754766	-0.5522169579833570	-0.3938169618013810	-0.5691535569171420
2	-0.47937237665289400	-0.49608456944175400	-0.4611570761911920	-0.4903865402231290	-0.357839596954082	-0.47062179671739000

3	-0.4256159168940070	-0.42631186711436800	-0.23883399490933300	-0.40434222846381600	-0.40126306076162100	-0.4203829858700660
10	-0.539535328627335	-0.4131316956580260	-0.5352639486289130	-0.5486178157089130	-0.531405613569027	-0.5492500087347300
11	-0.4759670388779690	-0.4710979266844470	-0.4747379291400810	-0.4810137342598390	-0.36656194350838400	-0.47954417963422100
12	-0.4215053540250740	-0.42577536616645100	-0.4352170852025000	-0.4263180209279350	-0.17056073799813200	-0.45888926286469900
13	-0.3505743304335890	-0.30871778118994800	-0.16367884207938200	-0.33253311451033700	-0.3207928536770200	-0.3370697824826730
20	-0.4788317520307530	-0.4510828361110000	-0.45937232961716400	-0.47225717486715000	-0.26386019366984500	-0.47665871958877500
21	-0.422127711367946	-0.41244387307318300	-0.17301162909398900	-0.38773252251986400	-0.4025504145681240	-0.40271024514944500
22	-0.2682389316678360	-0.2630422497243550	-0.3015317406997540	-0.2766079854726740	-0.06945218234038190	-0.26295115282920200
23	-0.2401787411257990	0.06741951253743060	-0.26971790511783200	-0.23284881813341100	-0.2583035851695630	-0.2422431110665670
30	-0.44172171014683100	-0.44122245421621100	-0.45187941246859900	-0.4457438932250580	-0.3613180205774540	-0.4646944954244180
31	-0.3423154582817790	-0.32258419282522600	-0.26529200162736500	-0.344114604133151	-0.3269898108205320	-0.34033874148369300
32	-0.2594743402624220	-0.27795585188110200	-0.07960644397507250	-0.2486691819454580	-0.27323282517081700	-0.27893373189708500
33	-0.22379502157757400	-0.23209516809370100	0.12354446629184200	-0.2262190625	-0.22068069892479800	-0.23815442213822200
100	-0.5895032399356770	-0.5765186026818300	-0.5073639793810160	-0.5814740734834390	-0.5676641527002200	-0.5694650890775210
101	-0.4833030900250030	-0.4517326118929570	-0.4929181677266380	-0.4837103203234430	-0.4672166495281830	-0.4677605742484270
102	-0.4160564477065690	-0.4138026560068960	-0.10929556148611700	-0.407962313768681	-0.41229909561130500	-0.41474186176747000
103	-0.3269099862890630	-0.017701271785637500	-0.3375885321734240	-0.3362707830134110	-0.30641666548720800	-0.33198051287477600
110	-0.47471646662401400	-0.474484264607495	-0.49405007259177700	-0.47285960907584200	-0.3984873781098080	-0.46387917259973700
111	-0.3977692657810160	-0.42404543508899600	-0.40523429863818900	-0.4301058440507140	-0.16124734426018200	-0.4069723584906270
112	-0.2703334954287900	-0.2706719444159110	0.048776644050625200	-0.2827493263542370	-0.2817939293116650	-0.31480356146684300
113	-0.23668085937500000	0.19160863473908300	-0.24514265079860200	-0.2765124154200890	-0.26643412109375000	-0.24569709453125000
120	-0.4005816288772970	-0.30448305481893000	-0.4092014427123630	-0.33250925549621800	-0.40457486804286	-0.38329061155671200
121	-0.3538519216147580	-0.3145220323381860	-0.10958554384586200	-0.3322498856855070	-0.33796047157245	-0.35306326719622200
122	-0.21319944623930100	-0.18989093330256400	0.19727076166919200	-0.1927473361229180	-0.1997226520772700	-0.2668894756429330

123	-0.07960981908434160	0.17205748474275400	-0.038495165107711000	-0.0975	-0.12482620152141400	-0.11887500000000000
130	-0.3315067496407780	-0.358351193125258	-0.38131070904098500	-0.3604881472958700	-0.24879844544394400	-0.3317782687109380
131	-0.2467709545396760	-0.2824889413085940	-0.24687026104185300	-0.28194139609535500	-0.15484513177975300	-0.24985046093750000
132	-0.13229842826730600	-0.142625	-0.12498287001505800	0.005885965038176610	-0.1352807864075510	-0.13918125000000000
133	0.012460135987360600	-0.0975	0.39567729339081400	-0.05	-0.07250000000000000	-0.05
200	-0.45870385741014600	-0.505784499921099	-0.45582672653867400	-0.4859363527974940	-0.47791751881335900	-0.45138019658301100
201	-0.37305850378320300	-0.37877107911330600	-0.3584988261726030	-0.361101816811914	-0.21325644198966300	-0.37206135924674700
202	-0.3116379094055880	-0.06663578750656700	-0.31554456972571800	-0.2928159138713580	-0.2964064750735930	-0.29687825120544300
203	-0.27819474257812500	-0.0034722742203117300	-0.28283277647237800	-0.28926029028511100	-0.2790745390625	-0.2675038455343070
210	-0.423678376297431	-0.4149654988045270	-0.41864412973332000	-0.44541816797720100	-0.35273693895168000	-0.4083842136264880
211	-0.34718903715019200	-0.28003568851642900	-0.33442529881757500	-0.3106651102102970	-0.3201226843287150	-0.35027520639905300
212	-0.21093272707031300	-0.1863055609832820	-0.2179315854026470	-0.198178125	-0.06275747382645380	-0.1809404652126160
213	-0.09495636579558700	-0.1692490862270160	0.1485239372751710	-0.20007909130872700	-0.18549375	-0.1695487930761700
220	-0.3494962327610120	-0.37470401469738200	-0.2235134015467180	-0.3270318510112200	-0.3438093502281530	-0.3398056421699660
221	-0.2388664184748510	-0.202245813627547	-0.20649014576962100	-0.18457181392407900	0.07657517122354800	-0.20939076426861100
222	-0.13137500000000000	-0.11000000000000000	0.01329342181525910	-0.142625	0.3126433218695160	-0.12493718209937400
223	-0.04551864608184310	-0.07250000000000000	0.37966102988942200	-0.05	-0.07250000000000000	0.1186767477360480
230	-0.3178078490046330	-0.32004000117187500	-0.2976386408013460	-0.30100153831390400	-0.2227367069282850	-0.3185224862902490
231	-0.14429703125	-0.15331250000000000	-0.15729101518630200	-0.14768750000000000	0.14750272839473700	-0.13056811823753900
232	-0.107625	-0.0975	0.4059118486639530	-0.05	-0.05	-0.05
233	-0.07500000000000000	-0.07500000000000000	0.4052117457814650	-0.05	-0.05	-0.05
300	-0.4130540839890690	-0.41165114948659100	-0.40674300154165100	-0.42081799490421500	-0.39032541130191300	-0.44682342555676500
301	-0.3409600601024330	-0.34050054260860900	-0.3686715579239850	-0.33265945789576200	-0.16950071622428200	-0.3307439744489760
302	-0.26490810937500000	-0.28004074439109800	-0.267446390625	-0.2601329557024510	0.007558040633873690	-0.27738489609552100

303	-0.2403639176628140	0.1261310624910280	-0.2262190625	-0.22509740858102200	-0.2262190625	-0.22578319364218500
310	-0.38703330079922800	-0.3547925349551700	-0.3941923956386720	-0.3803364652595330	-0.3934666178846320	-0.3991970566362480
311	-0.30522354114344700	-0.27923050171297300	-0.2800504557430080	-0.2961816556292200	0.043789237387022300	-0.31198930914466600
312	-0.0975	0.11552459026395400	-0.1164795747446900	-0.11289857260771300	-0.0975	-0.07182889742990600
313	-0.13345078125000000	0.4416198808646170	-0.0803128397291211	0.09619720555415460	-0.10194892841367400	-0.10165156250000000
320	-0.2970772984993130	0.0036366235347431200	-0.29810993991499900	-0.31551480650634400	-0.30571344024346100	-0.3080536489052870
321	-0.1736490490526410	0.2786963042630330	-0.1591830808735220	-0.13459720511135700	-0.14692026959065100	-0.13137500000000000
322	-0.0975	0.4194791048390920	-0.08166468750000000	-0.07500000000000000	0.1531355171098290	-0.0174315188177877
323	-0.05	0.9310526935751910	-0.05	0.0	0.21438867875422200	0.2503250723278560
330	-0.23601335894140500	-0.27713920034279500	-0.23856890420075900	-0.2344882996764000	0.36628822541144600	-0.2601961932957860
331	0.01145550864046750	-0.142625	-0.142625	-0.15440680261241200	0.703843971717652	-0.11887500000000000
332	-0.05	-0.05	-0.05	0.12071107468303600	0.8128006845358200	0.0
333	0.0	0.0	0.0	0.0	0.0	0.0