Final_611-taxi

March 4, 2025

1 *Reinforcement Learning (RL)*

Fatima Sultana

2 `gymnasium` `Taxi-v3` environment with Q-Learning Agent

Objective: Implement a Q-Learning agent that attempts to solve the *Taxi Problem*.

2.0.1 Reference

Details of the `Taxi-v3` environment are available here: https://gymnasium.farama.org/environments/toy_text/tax

2.1 Step 0: Import the dependencies

[ ]:
```
!pip install gymnasium --quiet
```

```
                                        0.0/953.9
kB ? eta -:--:--

92.2/953.9 kB 2.7 MB/s eta 0:00:01

686.1/953.9 kB 9.9 MB/s eta 0:00:01
                                 952.3/953.9
kB 12.3 MB/s eta 0:00:01
                                 953.9/953.9 kB
8.5 MB/s eta 0:00:00
```

[ ]:
```
import numpy as np
import gymnasium as gym
import random
import time
```

[ ]:
```
!pip install renderlab --quiet
```

[ ]:
```
import renderlab as rl
```

2.2 Step 1: Create the environment

- Let's create Taxi-v3 environment
- This environment is part of the Toy Text environments provided by OpenAI Gymnasium.
- The Taxi Problem entails navigating through a grid world to locate passengers, picking them up, and then dropping them off at their destinations.

In this environment, the setup options are much more limited compared to the FrozenLake-v1 environment. The Taxi-v3 environment does not have configuration parameters such as map_name, is_slippery, or render_mode that we can set during initialization. The environment is predefined and has default settings. Hence only gym.make('Taxi-v3') is used to create the Taxi-v3 environment.

[ ]:
```python
#Function to display additional information about the environment
def query_environment(env, name):
    print(f"{name} environment information")
    print(f"Action Space          : {env.action_space}")
    print(f"Action Space Size     : {env.action_space.n}")
    print(f"Observation Space     : {env.observation_space}")
    print(f"Observation Space Size: {env.observation_space.n}")
    print(f"Reward Range          : {env.reward_range}")
    print()

#Create the Taxi-v3 environment
env = gym.make('Taxi-v3')

#Query the environment
query_environment(env, "Taxi-v3")
```

```
Taxi-v3 environment information
Action Space          : Discrete(6)
Action Space Size     : 6
Observation Space     : Discrete(500)
Observation Space Size: 500
Reward Range          : (-inf, inf)
```

Action Space: There are 6 discrete actions that the agent can choose from. These represent following:

0: Move south (down)

1: Move north (up)

2: Move east (right)

3: Move west (left)

4: Pickup passenger

5: Drop off passenger

Observation Space: The environment has 500 discrete states. These states encode various scenarios in which the taxi and the passenger can be, including different locations for the taxi, the passenger, and the destination.

Reward Range: The possible range of rewards is from negative infinity to positive infinity, indicating a very flexible reward system that isn't constrained to specific values.

2.3 Step 2: Create the Q-table and initialize it

Q-table is crucial for implementing the Q-learning algorithm, as the agent will iteratively update this table based on the rewards received from the environment for taking specific actions in specific states.

- Creating the Q-table, by calculating the action_size and the state_size
- OpenAI Gym provides env.action_space.n and env.observation_space.n to do that

[ ]:
```
action_size = env.action_space.n
state_size = env.observation_space.n
Q_value = np.zeros((state_size, action_size))

print(Q_value.shape)
print(Q_value.size)
```

```
(500, 6)
3000
```

Q_value is initialized as a two-dimensional NumPy array with dimensions (state_size, action_size), filled with zeros. This array will be used to store the Q-values, where each element (i, j) represents the value of taking action j in state i. This table will guide the agent to take the most rewarding actions, aiming to maximize cumulative rewards.

2.4 Step 3: Specify the hyperparameters

Specifying various hyperparameters for configuring a Q-learning algorithm in a reinforcement learning environment, likely for the Taxi-v3 task. These parameters allow fine-tuning of the agent's learning behavior. Adjusting these values can help avoid situations where the agent either learns too slowly (undertraining) or memorizes specific paths rather than generalizing from its experience (overtraining)

[ ]:
```
total_episodes = 2000 #1000 #4000 #2000 #500  # Total episodes for training(A␣
 ↪higher number to ensure robust learning)
max_steps = 100 #99 #98        #Max steps per episode (Enough steps to allow␣
 ↪completion of "long" episodes without looping)

learning_rate = 0.1 #0.7    #rate at which the agent adopts new information
discount_factor = 0.9 #0.99 #0.618    #factor by which future rewards are␣
 ↪diminished as they are brought to present value.

#Controlling the exploration-exploitation balance.
exploration_rate = 1.0
min_exploration_rate = 0.01
max_exploration_rate = 1.0
```

```
exploration_decay_rate = 0.01  #Adjusted for a more gradual decay over more␣
↪episodes
```

Proper tuning of total_episodes and max_steps is essential for achieving a balance between training duration and computational efficiency. We have methodically tested various values for these parameters to optimize our training process. The impact of these various values will be thoroughly analyzed in the observation section to illustrate their effects on the learning performance and efficiency of our model. The exploration and exploitation balance is fundamental in ensuring that the agent doesn't get stuck in local optima.

2.5 Step 4: Train via the Q-Learning algorithm

Steps in Q-learning: 1. Initialize the Q-table with zeros (eventually, updating will happen with a reward received for each action taken during learning). 2. Updating of a Q value for a state-action pair, $Q(s, a)$, is given by:

$$Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma \max_{a'} Q(s',a') - Q(s,a)]$$

where * s = current state * a = action taken (choosing new action through *epsilon-greedy approach*) * s' = resulting next state * a' = action taken to get to resulting next state * r = reward received for taking action a * α = the *learning rate*; that is, the rate at which the learning of the agent converges towards minimized error * γ = the *discount factor* (or *reward decay rate*); that is, discounts future reward to get an idea of how important that future reward is with regards to the current reward 3. By updating the Q values using the formula in step 2, the table converges to obtain accurate values for an action to take in a given state.

[ ]:
```
#Initialize lists to store rewards, Q-table snapshots, states, and actions
rewards = []
history = []
all_states = []
all_actions = []
#Loop through total episodes
for episode in range(total_episodes):
    #Reset the environment
    raw_state = env.reset()
    state = raw_state[0] if isinstance(raw_state, tuple) else raw_state  #␣
↪Extract state integer if it's a tuple
    total_rewards = 0

    #Loop through steps within each episode
    for step in range(max_steps):
        #Epsilon-greedy action selection
        if random.uniform(0, 1) < exploration_rate:
            action = env.action_space.sample()  # Explore and select a random␣
↪action
        else:
```

```
            action = np.argmax(Q_value[state, :])  # Exploit and select the␣
↪best action from this state

        #Take the action (a) and observe the reward (r) and resultant state␣
↪(new_state)
        raw_new_state, reward, done, truncated, info = env.step(action)
        new_state = raw_new_state[0] if isinstance(raw_new_state, tuple) else␣
↪raw_new_state

        #Q-Learning algorithm update
        Q_value[state, action] = Q_value[state, action] + learning_rate * (
            reward + discount_factor * np.max(Q_value[new_state, :]) -␣
↪Q_value[state, action])

        #Log all states and actions
        all_states.append(state)
        all_actions.append(action)

        state = new_state  #Update state to new_state
        total_rewards += reward #Add the rewards

        if done:
            break

    #Store Q-table snapshot after each episode (or at another interval)
    if episode % 100 == 0:  # Adjust this interval as needed
        history.append(np.copy(Q_value))

    #Decay the exploration rate
    exploration_rate = min_exploration_rate + (max_exploration_rate -␣
↪min_exploration_rate) * np.exp(-exploration_decay_rate * episode)
    rewards.append(total_rewards)

#Calculate and print the average reward per hundred episodes
for i in range(0, total_episodes, 100):
    print(f"Episode {i+1}-{i+100}: Average Reward: {np.mean(rewards[i:i+100])}")
```

```
Episode 1-100: Average Reward: -134.84
Episode 101-200: Average Reward: -9.79
Episode 201-300: Average Reward: 2.96
Episode 301-400: Average Reward: 5.71
Episode 401-500: Average Reward: 7.32
Episode 501-600: Average Reward: 7.5
Episode 601-700: Average Reward: 7.01
Episode 701-800: Average Reward: 7.44
Episode 801-900: Average Reward: 7.65
Episode 901-1000: Average Reward: 7.72
```

```
Episode 1001-1100: Average Reward: 7.75
Episode 1101-1200: Average Reward: 7.07
Episode 1201-1300: Average Reward: 7.41
Episode 1301-1400: Average Reward: 7.13
Episode 1401-1500: Average Reward: 7.16
Episode 1501-1600: Average Reward: 7.24
Episode 1601-1700: Average Reward: 7.35
Episode 1701-1800: Average Reward: 7.39
Episode 1801-1900: Average Reward: 7.53
Episode 1901-2000: Average Reward: 7.33
```

Average reward for every 100 episodes is displayed. The rewards start quite negative, indicating early struggles of the learning algorithm to find successful strategies. As episodes progress, there is a noticeable improvement in the average reward, transitioning from negative to positive values, which suggests that the learning algorithm starts to find and exploit successful strategies. After the initial improvement, the average rewards per 100 episodes stabilize around values slightly above 7, indicating that the model has likely converged to a stable policy.

[ ]:
```
#Priniting score overall
print ("Score over time: " +  str(sum(rewards)/total_episodes))
```

```
Score over time: -0.898
```

The average score over all episodes of a reinforcement learning task is calculated as -0.898. This indicates that it has learned the key aspects of the task effectively as initially the reward was around -134.

CHECKING OUT THE Q_TABLE

[ ]:
```
#A function to print Q-table in a readable format with action symbols for␣
 ↪clarity
def printQtable(arr):
    #Define action symbols for Taxi-v3
    actions = ["↓", "↑", "→", "←", "Pickup", "Dropoff"]
    print("Q:SxA    ", end=" ")
    for a in actions:
        print("{:8s}".format(a), end=" ")
    print("")

    #Print table contents
    i = 0
    for row in arr:
        print("{:5d}".format(i), end=" ")
        for item in row:
            print("{:8.5f}".format(item), end=" ")
        i += 1
        print("")
```

```python
#Q-table from your Q-learning training for Taxi-v3
printQtable(Q_value)
```

```
Q:SxA     ↓        ↑        →        ←        Pickup   Dropoff
    0  0.00000  0.00000  0.00000  0.00000  0.00000  0.00000
    1 -0.02289 -1.63555 -2.34224  0.30496  9.62207 -3.52448
    2  1.71173  3.09888  0.91133  0.80378 14.11881 -1.06157
    3 -2.05475 -0.80421 -1.68483 -3.24293 10.72936 -3.74663
    4 -4.60963 -7.36049 -7.33975 -7.38703 -9.09589 -7.48130
    5  0.00000  0.00000  0.00000  0.00000  0.00000  0.00000
    6 -7.30450 -7.58185 -6.65007 -7.49874 -9.95038 -10.55235
    7 -5.90475 -5.89151 -4.67912 -5.91525 -6.53500 -6.60040
    8  5.63994 -3.53560 -2.96249 -2.92298 -5.48271 -7.99973
    9 -0.45365 -6.29095 -6.23949 -6.32555 -7.13687 -7.26797
   10  0.00000  0.00000  0.00000  0.00000  0.00000  0.00000
   11  2.62859 -4.54566 -5.07422 -4.90228 -8.59049 -8.82733
   12 -4.06910 -6.76183 -6.68353 -6.72934 -6.72177 -6.76240
   13 -3.34551 -5.72403 -5.76253 -5.71853 -5.93266 -7.58900
   14 -4.11666 -6.41012 -6.40922 -6.46113 -7.84541 -6.56868
   15  0.00000  0.00000  0.00000  0.00000  0.00000  0.00000
   16  9.14889  3.49100  9.45824  9.76019  3.81646 20.00000
   17 10.72936  1.25376 -2.74954  1.24610 -1.42736  2.22399
   18 15.27152  4.96267  0.14549  2.81419  0.47319  3.54195
   19 11.84784  1.00232 -2.31884 -0.03718 -1.58391 -0.16151
   20  0.00000  0.00000  0.00000  0.00000  0.00000  0.00000
   21 -3.44609 -3.47929 -3.89560  5.47110 -5.01542 -4.93792
   22  0.11944 -1.83976 -2.56498 10.94125 -4.19870 -4.49743
   23 -3.76767 -3.72099 -2.87978  8.12660 -3.94265 -4.44437
   24 -2.46998 -7.29556 -7.28682 -7.26416 -7.73323 -7.81669
   25  0.00000  0.00000  0.00000  0.00000  0.00000  0.00000
   26 -2.63034 -7.25214 -7.15803 -7.63003 -11.97234 -9.17831
   27  0.25202 -5.70726 -5.70469 -5.70008 -6.35303 -7.68014
   28  4.05507 -4.16060 -4.58404 -4.19545 -7.51045 -7.90250
   29 -4.59227 -6.37989 -6.38900 -6.38012 -6.80267 -6.62892
   30  0.00000  0.00000  0.00000  0.00000  0.00000  0.00000
   31  1.08020 -5.48967 -5.47393 -4.99347 -8.19746 -7.42158
   32 -2.58218 -6.59667 -6.56582 -6.61166 -6.88080 -7.87807
   33  0.72484 -5.62838 -5.58923 -5.63995 -5.74716 -6.57560
   34 -0.94766 -6.38095 -6.37691 -6.39591 -6.51641 -6.83025
   35  0.00000  0.00000  0.00000  0.00000  0.00000  0.00000
   36  0.12411  2.42839  3.22731 17.83306 -1.00000  0.58666
   37  2.64014 -2.48770 -2.48064 -1.37377 -2.83785 -2.99700
   38 -1.23892 -1.18754 -1.25666  6.80832 -3.96975 -4.43958
   39  0.38889 -2.19411 -2.13605 -2.08748 -3.85073 -5.01812
   40  0.00000  0.00000  0.00000  0.00000  0.00000  0.00000
   41  0.27676 -7.55345 -7.41977 -7.49351 -9.90701 -9.56048
   42  4.03690 -4.44517 -4.99164 -4.64232 -5.84502 -6.17640
```

43	-4.39591	-6.53829	-6.48672	-6.46540	-6.73023	-6.62274
44	-4.87850	-4.41499	5.85164	-4.87869	-6.53258	-5.40061
45	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
46	-4.64894	-4.21148	5.37904	-4.57457	-4.55177	-4.84481
47	-2.45470	-3.21978	7.30657	-2.73762	-6.07470	-4.08734
48	1.20479	-5.69223	-5.92608	-5.82499	-6.22862	-7.81559
49	-4.03721	-7.83366	-7.89492	-7.87658	-7.93537	-7.78665
50	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
51	2.84132	-6.65882	-6.66974	-6.36794	-7.35170	-8.26662
52	0.00694	-5.54203	-5.02480	-5.46571	-9.91010	-6.66705
53	-3.83041	-4.59350	2.53806	-4.45247	-6.11524	-6.42865
54	-4.35075	-5.54857	0.39758	-5.31242	-7.81670	-6.67067
55	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
56	1.11021	-2.02457	-2.10973	-1.97139	-2.94197	-1.99900
57	-0.29970	-0.29970	0.23214	-0.29970	-1.00000	-1.00000
58	-1.45998	-1.89006	-1.93263	-1.96403	-4.30641	-3.50241
59	-0.28676	-0.97684	1.87577	-0.94493	-2.76637	-3.51773
60	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
61	-2.47939	-7.65809	-7.86381	-7.60611	-10.25289	-8.48875
62	-2.11713	-5.16223	-5.21816	-4.87503	-5.28308	-5.47762
63	-5.55522	-6.69241	-6.77539	-6.66240	-6.70901	-6.73852
64	-1.52016	-3.19561	8.52585	-2.02099	-4.39202	-4.59210
65	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
66	-4.23683	-0.16903	8.52585	-3.07508	-4.00816	-4.57530
67	0.52542	-0.27440	11.84784	-2.41302	-4.19578	-3.07464
68	1.56750	-6.09020	-6.06150	-5.93257	-8.48013	-6.97117
69	-1.01398	-8.06072	-8.02392	-8.00168	-9.54475	-9.49446
70	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
71	-6.60721	-7.02305	-7.03980	-2.30713	-7.76963	-11.28838
72	0.21944	-5.26799	-5.40244	-5.60601	-6.43252	-6.26050
73	8.13738	-2.39462	-4.28637	-4.12001	-6.08169	-5.99130
74	5.50278	-3.01311	-5.24510	-5.09818	-6.26216	-6.14012
75	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
76	-0.86279	-2.43333	-2.44012	-2.33905	-5.70474	-3.52055
77	1.46368	-0.19990	13.88944	-0.28583	-1.90990	-1.00000
78	11.84784	1.25634	-2.28287	-2.25292	-3.19803	-2.09899
79	15.27152	1.87621	1.59635	-0.93785	-2.42913	0.08659
80	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
81	-7.81047	-7.77599	-7.77522	-6.49737	-9.72330	-8.75886
82	-3.70952	-5.43456	-5.41391	-5.35051	-5.40745	-6.10469
83	-2.94527	-6.75463	-6.79432	-6.83005	-8.14764	-7.94510
84	-0.09452	-0.40914	0.42807	-0.51176	9.62207	-3.00708
85	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
86	-1.68713	-1.29853	1.35209	0.20592	9.62207	-2.54779
87	0.99415	1.56003	1.55308	0.20581	12.97762	-2.07143
88	-5.43668	-6.25108	-6.27751	-6.27954	-6.96231	-6.42384
89	-8.16991	-8.15174	-8.15035	-6.83062	-8.58479	-8.44926
90	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000

91	-6.29326	-7.31452	-7.31271	-7.26065	-7.97206	-7.90891
92	-1.88744	-5.39184	-5.35330	-5.33566	-5.71616	-5.44086
93	-3.96394	-4.36748	-3.67600	3.34727	-6.97315	-5.63021
94	-1.39062	-5.24318	-5.25562	-4.06589	-5.82168	-5.76442
95	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
96	10.72936	-0.17213	0.45797	-2.87230	-3.06128	-2.96775
97	7.09343	14.90866	6.39971	4.84505	4.77786	20.00000
98	-2.23873	1.43681	2.26300	10.72936	-1.69775	1.06346
99	0.42021	5.25937	1.19298	14.11881	-0.93011	0.89173
100	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
101	-1.49511	8.52585	1.08511	-0.45278	-3.90628	-4.03918
102	1.96739	12.97762	1.23288	2.32645	-1.14351	-0.58296
103	-2.86865	9.62207	-0.75647	-1.46809	-3.26634	-2.55260
104	-7.10988	-7.11427	-0.04767	-7.13022	-7.84075	-7.51962
105	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
106	-3.20739	-7.15755	-6.03988	-7.01335	-10.40625	-9.74141
107	-5.75819	-5.76811	-2.26373	-5.80371	-6.29851	-5.85376
108	10.16269	-2.43930	-0.59513	-2.12859	-3.95584	-4.05533
109	5.13789	-6.06751	-5.71811	-6.08095	-7.00136	-7.91257
110	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
111	7.42220	-3.25026	-1.61622	-1.51321	-4.56969	-4.58097
112	1.05026	-6.66302	-5.02968	-6.14255	-7.73684	-8.13081
113	3.91115	-5.41035	-5.44846	-5.44005	-6.36461	-5.90600
114	1.93411	-6.28915	-5.49236	-6.25340	-7.00120	-6.62839
115	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
116	7.51316	18.80000	2.64016	7.04116	4.88811	4.08332
117	1.39091	3.18933	11.84784	2.01059	-2.47874	-1.01556
118	16.43588	1.86872	1.83487	6.67974	1.24976	-2.32635
119	12.97762	1.45244	-0.30039	2.83963	-1.35090	-0.31249
120	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
121	-1.53669	-3.63325	-2.93269	7.44059	-5.14172	-4.02870
122	-0.64411	-2.21184	-3.02085	9.51869	-3.45781	-6.62226
123	-2.16449	-3.22943	-4.20142	8.52585	-4.85180	-3.50509
124	2.98570	-6.88643	-6.51542	-6.94689	-7.22080	-7.70244
125	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
126	2.24884	-7.06355	-5.46306	-6.85339	-9.58320	-9.51831
127	5.84972	-5.37560	-5.43085	-5.42638	-5.62164	-5.81326
128	9.42622	-3.85186	-1.14241	0.00632	-5.47809	-7.05330
129	2.92878	-6.26885	-6.28341	-5.67930	-7.23101	-7.70097
130	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
131	0.06585	-5.21104	-3.66443	5.77445	-7.75262	-7.83599
132	3.42013	-6.33520	-5.35735	-5.99038	-6.73043	-7.27436
133	7.03355	-5.28954	-4.74223	-5.25296	-5.85651	-5.48460
134	4.48262	-6.16451	-5.13921	-6.18438	-6.89426	-6.45757
135	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
136	3.94102	10.96651	0.54999	1.56525	-1.00000	-2.51735
137	12.97762	-1.93780	0.57620	3.11823	-0.21104	-0.93449
138	9.02015	-0.53473	-0.43292	2.09044	-1.90000	-2.73860

139	10.05543	-1.92038	-0.42059	1.99645	-3.44194	-3.65067
140	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
141	4.15035	-7.14729	-6.42313	-5.63633	-8.63016	-7.75931
142	8.46819	-3.20420	-3.40403	-1.68871	-5.36666	-5.08793
143	3.97631	-6.09068	-6.07779	-6.07848	-6.63957	-6.88495
144	-5.26601	-5.17771	6.36617	-4.20530	-6.07544	-5.55070
145	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
146	-3.28636	-4.63112	6.36616	-4.94755	-5.51225	-5.07986
147	-2.61116	-2.63732	7.99956	-3.72530	-4.30032	-4.36543
148	7.39287	-5.51906	-5.12770	-4.85549	-7.74104	-5.88504
149	2.32843	-7.40532	-7.54820	-6.48796	-7.86711	-7.95341
150	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
151	5.28523	-6.15571	-6.13156	-6.35918	-6.72690	-6.41582
152	5.09447	-5.33194	-2.70052	-3.82491	-7.31659	-7.31188
153	-3.08238	-4.34027	5.22858	-4.39880	-5.86139	-5.66031
154	-4.02467	-5.18049	3.07023	-4.54108	-5.90755	-5.79688
155	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
156	12.28276	-1.69176	-1.36630	-1.68647	-1.99900	-1.60578
157	0.90398	-0.54632	6.52320	-0.49900	-1.61901	-1.48440
158	9.14276	-1.64954	-1.49672	-1.14750	-3.53265	-1.90990
159	8.35045	-0.89328	2.22651	-0.81859	-2.76692	-1.92869
160	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
161	-6.39331	-7.47027	-7.39662	1.52856	-8.76028	-7.77183
162	-2.80323	-5.01318	-4.99394	6.49713	-5.67317	-5.72189
163	1.03562	-6.43850	-6.35044	-6.40840	-6.85328	-7.30087
164	-3.87855	7.44059	-0.92268	-4.83368	-4.79988	-4.69867
165	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
166	-3.14026	7.44059	-4.56832	-2.69512	-4.89939	-4.37694
167	1.55004	10.72936	0.41401	-1.91845	-2.40737	-3.74949
168	6.58596	-5.83901	-5.72660	-5.07995	-7.40754	-8.12493
169	2.68582	-7.84631	-7.76964	-7.14365	-8.97254	-8.91855
170	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
171	-3.41059	-6.82401	-6.77490	3.28757	-8.25884	-7.49199
172	6.96704	-4.94519	-5.14586	-4.66121	-6.84491	-5.85451
173	9.61803	-1.53299	-4.10049	-4.14359	-3.61716	-4.89389
174	7.43670	-2.89010	-3.74802	-4.30211	-4.09395	-4.72990
175	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
176	-1.94346	-2.03360	-0.73624	5.39590	-1.97165	-4.78844
177	6.98130	2.69097	17.61200	0.35761	2.95271	1.43841
178	12.97762	0.49389	-0.60469	0.12693	-1.78227	-2.24639
179	16.43588	4.39334	0.94377	1.21165	-0.35463	-0.12525
180	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
181	-3.66358	-7.60787	-7.59708	-7.38889	-9.48155	-7.84514
182	-5.14556	-5.20750	-5.03549	1.37378	-5.90505	-5.92329
183	2.14990	-6.70074	-6.09760	-6.37231	-7.80577	-7.49317
184	-3.12426	8.45016	-2.26082	-3.30094	-4.56026	-4.22520
185	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
186	-3.24629	8.50930	-4.29038	-4.30174	-4.22305	-4.27912

187	-2.01892	11.76415	-1.59635	-2.89831	-4.30300	-5.43283
188	3.01039	-6.08035	-6.09538	-6.07693	-6.31903	-6.75011
189	-8.02841	-8.10656	-8.04748	-4.00927	-9.05966	-10.49416
190	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
191	-6.68399	-6.99890	-7.07155	-1.51374	-7.41503	-7.92048
192	3.70836	-5.09886	-5.14401	-5.11529	-5.89628	-5.33264
193	-4.03050	-4.09362	-3.74390	3.74182	-4.27985	-4.98984
194	-4.31642	-4.94680	-3.82888	4.81546	-5.52682	-6.09336
195	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
196	11.84784	0.90248	0.11755	-2.14108	-1.30689	-2.04211
197	6.72321	18.80000	9.67057	4.31561	4.47435	4.08693
198	-2.27333	-1.15430	-2.27488	5.26161	-4.20569	-5.85047
199	-0.93786	-0.93966	-0.59603	9.25504	-2.32781	-2.02408
200	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
201	-4.97738	7.18440	-3.33825	-2.50547	-5.67146	-5.29424
202	1.10708	11.84784	-1.86502	2.51536	-4.70720	-0.82089
203	-3.89308	8.15711	-4.36268	-3.34078	-4.92340	-4.65734
204	-6.90973	-6.88323	2.47399	-6.88600	-8.11174	-7.74586
205	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
206	-6.44716	-6.88484	3.01682	-5.14416	-9.94251	-9.53624
207	-5.68228	-5.64211	5.94823	-5.59861	-5.61098	-6.02790
208	11.84784	2.03625	5.42418	3.81824	-0.86617	-2.68803
209	7.44059	-3.85565	-2.03980	-2.37283	-5.04054	-4.23459
210	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
211	8.52585	3.06403	2.02412	0.91125	-4.39239	-3.68694
212	-6.04243	-6.00663	5.13358	-5.47553	-7.14694	-6.87188
213	-4.71354	-5.32797	7.34279	-1.86709	-5.41017	-5.35447
214	-6.07030	-6.03144	5.25622	-6.08193	-6.83743	-7.33960
215	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
216	11.18616	17.61200	9.11837	13.86567	5.47885	5.87892
217	3.07420	1.25018	12.97762	1.48617	-2.34939	-2.99700
218	17.61200	11.26042	10.81164	11.65827	0.60048	5.05334
219	6.58462	5.37173	14.11881	9.96131	0.76192	1.17140
220	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
221	-3.67502	6.36618	-4.90024	-1.57837	-5.21170	-5.36050
222	-3.73931	-3.62875	-2.42689	10.72936	-3.59725	-3.34914
223	-3.13214	7.44059	-4.59611	-3.80967	-4.89484	-4.92895
224	-6.25434	-6.10425	4.24609	-5.51224	-7.31848	-6.98119
225	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
226	-5.62642	-4.79449	4.24582	-4.77994	-6.72009	-7.04918
227	-4.47418	-5.01868	7.43933	-5.02131	-5.85964	-5.00416
228	1.17874	1.66612	3.99911	10.72936	-2.11450	-4.18996
229	-6.41708	-6.35049	-5.63695	6.36618	-6.07302	-6.33640
230	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
231	-0.10673	-1.58940	-1.13198	7.44059	-4.12986	-4.24792
232	-4.91265	-4.90862	6.36573	-4.18803	-5.47590	-5.77964
233	-4.23571	-3.00203	8.52567	-3.91625	-5.24533	-4.59051
234	-4.57770	-4.19056	6.36605	-4.86294	-6.44247	-5.66505

235	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
236	0.95822	2.87071	9.91040	16.43588	2.74585	0.91019
237	-0.43527	3.87594	14.11881	4.66132	1.16637	-1.56200
238	2.65911	0.85594	8.31984	16.43588	1.50068	-0.12877
239	1.19574	2.93657	15.27152	5.44787	1.70758	3.14539
240	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
241	-6.26086	-4.04993	-3.71267	5.30245	-6.09379	-6.45668
242	-4.26226	0.17776	-0.58750	9.62206	-3.41613	-4.42698
243	-5.55184	-5.55235	-4.64136	6.36618	-6.46660	-6.31532
244	-5.35943	5.30225	-5.72170	-5.77593	-6.06799	-5.65715
245	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
246	-5.41505	5.30223	-3.97927	-1.26344	-5.86886	-5.90933
247	-4.32494	-4.28028	8.52584	-3.44188	-4.30440	-4.92089
248	-4.29903	-4.44048	-0.14038	9.62207	-4.84432	-5.03177
249	-7.03950	-6.09362	-7.00541	5.30245	-7.67186	-7.85216
250	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
251	-5.30457	-3.61511	-2.96703	6.36617	-5.55304	-5.85644
252	-3.06538	-4.24259	7.44059	-4.24582	-4.14837	-4.30928
253	-4.21381	-4.01416	9.62207	-4.34920	-3.97856	-5.60891
254	-4.57491	-3.35773	7.44059	-2.38450	-5.00290	-4.45869
255	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
256	2.68892	4.01427	3.61395	15.27152	2.09318	-1.25573
257	-0.64093	-0.10053	15.27152	5.51192	2.19220	0.65820
258	2.16566	1.22032	0.18973	15.27152	1.74414	3.10923
259	-0.25499	-0.00865	16.43588	3.27384	1.96521	2.08720
260	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
261	-6.50525	-6.93493	-7.16855	4.23222	-7.80213	-7.61157
262	-2.78292	-4.05905	-4.17023	8.51993	-4.34772	-4.41199
263	-6.19717	-6.04576	-6.25237	5.30235	-6.29446	-6.40068
264	-4.88492	5.47679	-5.19207	-5.25664	-5.79857	-6.96174
265	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
266	-4.78742	6.05090	-4.21577	-4.22804	-5.23710	-6.20498
267	0.67308	9.62207	-1.85174	-0.67418	-3.31759	-3.77715
268	-4.12771	-4.70647	-3.90546	8.52575	-5.96241	-5.44307
269	-6.13536	-5.61632	-7.55601	4.24459	-7.93950	-7.87611
270	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
271	-6.16450	-5.53521	-6.40628	5.28325	-7.04634	-6.75850
272	8.52585	-1.28447	-3.92058	-1.37306	-4.60179	-4.61849
273	10.72936	-0.19059	-1.33355	0.43799	-3.92241	-3.35222
274	8.52585	-1.71177	-3.72109	-0.30912	-4.02734	-4.30959
275	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
276	0.80850	-0.70614	0.87767	14.11881	0.09535	-0.61272
277	3.32926	16.43588	4.74852	4.40330	3.14137	-1.00000
278	4.60954	5.37019	-1.12705	14.11881	0.83923	-1.33092
279	17.61200	7.77026	0.92430	9.20135	3.41283	3.15611
280	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
281	-7.51904	-7.47151	-7.47309	2.68278	-7.79289	-8.12915
282	-5.02275	-4.92654	-4.58197	4.49149	-6.22044	-7.07246

283	-6.33091	-6.41893	-5.27443	4.22768	-7.46551	-6.78415
284	-4.82995	6.45676	-4.84628	-4.86273	-5.65807	-5.84888
285	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
286	-4.80793	7.01706	-3.17122	-4.71938	-4.71378	-5.71580
287	-3.12251	9.08429	-3.33491	-2.21751	-3.63986	-4.80340
288	-5.17791	-5.79013	-3.22638	7.34071	-6.08744	-6.49802
289	-7.87900	-7.98732	-7.86328	-1.82522	-9.43413	-8.71026
290	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
291	-6.85866	-6.93085	-6.72010	-0.70195	-7.68434	-6.85598
292	-4.67729	-4.73304	-4.68654	7.05740	-6.13007	-4.99001
293	-3.71843	-3.68552	-3.70623	7.46926	-6.48119	-5.00958
294	-4.27611	-3.60504	-4.68891	4.29105	-7.17651	-6.32939
295	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
296	-1.87255	-0.72310	-0.59822	12.97762	-3.45781	-4.15435
297	-0.26915	15.95253	-0.49900	1.08061	-3.43732	-0.03315
298	-2.10589	-1.92472	-2.04306	3.29709	-4.93881	-2.75562
299	-0.04751	-0.22789	-0.49900	6.64678	-3.41501	-2.56885
300	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
301	-5.50355	2.53325	-4.93932	-5.44229	-7.22929	-6.02896
302	-2.88978	10.02863	-0.91528	-2.37347	-3.72221	-3.48751
303	-4.59945	4.79915	-4.59327	-4.08293	-6.29786	-6.35356
304	-7.20636	-1.96584	-7.26953	-7.21682	-7.87046	-8.39512
305	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
306	-7.36378	0.48230	-6.84586	-6.85488	-8.15014	-10.36249
307	-6.13517	2.53695	-6.09814	-6.10164	-6.73185	-7.65682
308	12.97762	4.12884	1.03883	3.99381	-2.54103	-0.57739
309	8.52585	-1.89236	-1.54189	-0.63936	-3.16495	-3.30446
310	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
311	9.62207	0.60464	1.06245	0.23768	-3.63537	-3.14079
312	-6.56753	1.64969	-6.41241	-6.54972	-8.32373	-7.12345
313	-5.93269	4.17003	-5.11390	-4.61806	-7.87909	-6.71360
314	-6.38841	2.69557	-6.38214	-5.77456	-6.98830	-7.08847
315	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
316	7.57039	16.43588	3.35428	6.61904	-2.49298	-0.35543
317	-0.42097	11.84784	3.55587	0.89377	-1.71616	-0.72987
318	18.80000	9.27089	11.40137	8.14603	3.02023	3.30584
319	1.31593	12.97762	6.36266	5.18670	1.09221	-1.85605
320	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
321	-5.20954	5.22544	-5.98964	-2.80163	-6.81398	-8.08654
322	-4.16027	8.65212	-4.22408	-3.43489	-6.34849	-5.00326
323	-5.25837	5.62151	-5.29022	-5.25079	-5.46233	-5.68518
324	-6.53875	1.21176	-6.51934	-6.01608	-6.75418	-7.87474
325	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
326	-6.35165	1.77392	-6.11981	-6.15098	-6.69650	-7.84004
327	-5.11225	3.06244	-5.07600	-5.08149	-5.29019	-5.95391
328	-1.52908	9.56105	-3.02665	0.23951	-5.13650	-4.35282
329	-6.91428	4.08139	-6.89177	-6.84311	-7.73199	-8.08455
330	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000

331	-5.06984	6.25679	-5.43082	-1.11702	-6.23531	-6.42813
332	-5.96506	4.51376	-4.13386	-5.15323	-5.94117	-6.40986
333	-4.79217	6.30766	-4.76300	-4.78201	-5.67855	-4.89394
334	-5.88003	4.26717	-4.86612	-5.95359	-6.67034	-6.37015
335	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
336	-1.14150	7.27115	0.17911	-0.69127	-2.72193	-2.97678
337	-1.64007	5.29997	-1.24266	-1.14937	-2.47677	-3.16339
338	-1.02287	9.57011	-0.87502	-0.98759	-4.74206	-1.85540
339	-1.26995	6.85209	-0.12265	-0.50328	-4.19993	-4.27693
340	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
341	-6.74798	-3.64318	-5.95721	2.27801	-8.88711	-7.84296
342	-4.34465	5.17558	-4.37782	-3.32932	-6.26728	-4.79992
343	-5.47360	-0.87085	-5.53392	-5.49907	-5.73830	-5.66008
344	-6.24424	1.80670	-5.94998	-6.21212	-6.86203	-6.68069
345	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
346	-6.05006	3.10515	-5.56433	-6.15042	-6.19011	-8.88556
347	-4.86650	6.52152	-4.05850	-4.85408	-4.89282	-4.96999
348	-5.20414	-4.99692	-3.52252	5.86831	-7.65112	-6.92516
349	-7.12997	1.68539	-7.13627	-7.09452	-7.44494	-7.77296
350	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
351	-6.18357	-3.98012	-5.92220	1.43034	-8.46045	-8.06751
352	-5.38241	6.05133	-2.73467	-5.00805	-7.09460	-6.42493
353	-4.63979	6.48431	-4.33941	-4.07210	-4.87576	-5.89932
354	-5.59516	5.89867	-5.54604	-5.62387	-6.67493	-5.81422
355	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
356	-1.22765	9.34884	0.04342	-1.07222	-3.41897	-4.34044
357	-1.26690	4.05366	-1.29223	-1.22991	-1.99900	-1.99900
358	-1.26442	8.79585	-1.15499	-0.86070	-3.75156	-5.12784
359	-0.97007	5.73784	0.06167	0.09325	-5.92554	-4.52810
360	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
361	-7.68521	1.27157	-7.50213	-6.75251	-8.75774	-7.74458
362	-4.80850	7.19968	-5.13080	-4.37009	-6.43594	-5.93297
363	-6.61888	3.32132	-6.58211	-6.15116	-7.27141	-7.49415
364	-5.45133	0.03295	-5.42765	-5.29173	-6.73943	-6.28473
365	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
366	-5.37074	2.76073	-5.21194	-5.12660	-5.64842	-5.80111
367	-3.10397	8.02858	-3.60231	-1.17964	-4.60398	-3.90016
368	-5.94388	6.39581	-5.20585	-5.90224	-7.20135	-6.54052
369	-8.01211	2.58632	-7.86320	-6.87138	-10.34958	-10.98108
370	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
371	-6.69028	3.93248	-6.80761	-6.32686	-8.10152	-7.67479
372	9.62207	-2.58992	-1.66422	-1.63554	-4.84000	-3.82755
373	11.84784	-0.65967	0.71654	1.64071	-2.55654	-1.70765
374	9.62207	0.71353	-0.32763	0.31026	-3.46424	-3.84355
375	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
376	2.21877	12.97762	-1.84719	0.33318	-0.21743	-1.70927
377	-1.05652	15.27152	0.01217	-1.05721	-0.02683	-1.51047
378	2.21773	12.97762	-1.79280	3.57007	-1.82820	-0.90059

379	18.80000	10.43639	2.19584	4.69995	4.01461	4.46271
380	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
381	-7.93180	-3.02037	-7.70262	-6.62941	-11.33445	-11.20539
382	-5.36976	-4.86257	-5.22710	-1.87135	-5.68471	-6.29593
383	-6.79354	1.55890	-6.73704	-6.73824	-7.49624	-6.76401
384	-5.25849	1.49248	-5.24055	-5.19824	-7.94830	-5.86841
385	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
386	-5.14950	3.40988	-5.11405	-5.03995	-7.28867	-5.98098
387	-3.66189	1.81473	-3.28574	-2.03842	-6.17711	-4.66548
388	-6.24533	4.58898	-5.78145	-5.21154	-8.25332	-7.09305
389	-8.29651	-8.00963	-8.03331	-3.93479	-10.43494	-10.93239
390	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
391	-6.97640	-6.77248	-6.94239	-4.19787	-7.12657	-7.44313
392	-4.26116	-4.29153	-4.29170	6.36366	-4.89131	-4.90215
393	-2.33331	-3.30286	-3.32262	9.55487	-3.78676	-4.19930
394	-2.03526	-3.90287	-4.28643	5.95150	-4.98563	-4.99861
395	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
396	-2.36337	-2.27011	-2.01021	4.35463	-4.60950	-5.49329
397	-0.98691	6.89100	-0.11978	-0.93209	-1.00000	-3.15910
398	-1.85205	-2.18570	-2.35506	2.57900	-7.00898	-5.60935
399	0.85704	-0.19990	0.18861	9.38995	-2.47577	-4.64384
400	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
401	-5.91601	-3.48206	-5.91926	-5.91702	-6.55367	-6.62466
402	-3.26544	3.49916	-3.13590	-3.32766	-5.07094	-4.27239
403	-4.86150	-1.06826	-4.84694	-4.77809	-6.32817	-4.91075
404	-7.63656	-6.16309	-7.65990	-7.58554	-7.86895	-7.80162
405	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
406	-7.59537	-4.86699	-7.27223	-7.74211	-12.20131	-12.31172
407	-6.54129	-3.34204	-6.47864	-6.47701	-7.00434	-6.55001
408	0.67045	4.32610	2.97991	2.97097	14.11881	-1.04155
409	-1.74088	-2.00411	0.17683	0.17894	9.62207	-5.31932
410	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
411	4.01149	-0.14341	4.56377	4.01408	10.72936	-2.00438
412	-6.87801	-4.87327	-6.83338	-6.85712	-7.69634	-6.95359
413	-6.29462	-1.78015	-6.24559	-6.16346	-7.82604	-8.59501
414	-6.61865	-3.08564	-6.62031	-6.63066	-6.89385	-9.10713
415	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
416	4.83660	15.27152	2.62219	5.77154	-0.52260	5.19021
417	-2.16736	10.72936	-3.42693	-0.95707	-2.89881	-1.21236
418	9.71714	10.87943	10.66417	11.47775	4.59188	20.00000
419	-0.19111	11.84784	4.92542	4.93052	-1.18815	3.04429
420	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
421	-5.96519	1.82055	-6.68044	-6.02032	-7.88868	-10.04609
422	-4.45534	1.89341	-4.40341	-4.38188	-5.57311	-4.74271
423	-5.51441	0.89897	-5.62289	-5.44661	-7.20291	-5.70112
424	-6.89412	-6.71572	-6.59652	-6.74288	-7.87071	-10.12870
425	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
426	-6.66981	-3.80379	-6.64559	-6.66509	-7.23024	-7.48462

427	-5.23816	-2.68151	-5.24515	-5.30537	-5.43126	-6.11398
428	-5.01087	6.74892	-5.03483	-4.56060	-7.25104	-8.19810
429	-7.12751	-1.56757	-7.11378	-7.12008	-8.31561	-7.69323
430	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
431	-5.85860	2.98142	-6.26343	-5.31080	-7.96905	-8.44042
432	-6.08045	-0.50262	-5.77793	-6.08517	-6.18741	-6.80388
433	-4.90282	0.92803	-4.90362	-4.85872	-5.24771	-5.36812
434	-5.71478	-0.65049	-5.97714	-6.01714	-6.86133	-6.14162
435	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
436	-1.18605	-0.39286	-1.08066	-1.14809	-1.00000	-4.42514
437	-1.57189	-1.46534	-1.65649	-1.57189	-2.93343	-1.90000
438	-1.18446	-1.09724	-1.21156	-1.18693	-1.99900	-1.99900
439	-1.26372	-0.73459	-1.03422	-1.44219	-4.82182	-3.57057
440	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
441	-7.02817	-3.01932	-6.90345	-6.38428	-9.82478	-10.57570
442	-4.57007	-1.65470	-4.63935	-4.56348	-5.34557	-4.59928
443	-5.63117	-5.62317	-5.68447	-3.76991	-5.87239	-6.58405
444	-6.52249	-3.46836	-6.53660	-6.62913	-6.72273	-7.39985
445	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
446	-6.45089	-1.23110	-6.47946	-6.50625	-6.83041	-7.35403
447	-5.14064	1.58135	-5.03098	-5.11550	-5.89628	-5.71763
448	-5.32379	-3.50506	-5.26355	0.66875	-7.61456	-9.16839
449	-7.30065	-4.32168	-7.22913	-7.29968	-7.93276	-7.84206
450	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
451	-6.45162	-6.03787	-6.29545	-2.86050	-6.92968	-7.51064
452	-5.60373	2.39751	-5.86804	-5.88037	-6.53899	-6.34154
453	-4.85741	0.12706	-4.86448	-4.81922	-5.13756	-5.88820
454	-5.93884	-0.90252	-5.86181	-5.97610	-5.97875	-6.72068
455	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
456	-1.09452	-1.13110	-1.08760	-1.15260	-1.90000	-1.93954
457	-1.39094	-1.05278	-1.39094	-1.50478	-2.92234	-1.93073
458	-1.09452	-0.57587	-1.08471	-1.27376	-1.99900	-1.99900
459	-0.99551	-1.02225	-0.99551	-0.98874	-1.00000	-4.19197
460	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
461	-7.91910	-3.96299	-8.08430	-7.83531	-10.41574	-7.78231
462	-5.41905	4.46366	-5.44789	-5.42764	-5.90153	-6.29617
463	-6.96896	-1.83727	-6.96769	-6.93506	-7.58524	-7.66899
464	-5.72275	-4.75229	-5.70416	-5.72604	-6.55953	-5.76344
465	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
466	-5.72237	-2.95668	-5.59710	-5.63169	-6.60947	-5.80136
467	-3.77501	3.67788	-4.07502	-3.43873	-5.20430	-6.67978
468	-6.12666	1.88263	-6.43292	-6.08578	-9.19234	-6.21088
469	-8.14904	-1.75753	-8.30848	-8.10956	-8.93007	-8.95293
470	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
471	-7.20422	0.87071	-7.32989	-7.11412	-7.97206	-7.71504
472	0.12966	-2.27578	-2.94302	-0.97144	10.72936	-4.54131
473	4.23953	2.91992	0.97149	3.39667	12.97762	-1.31109
474	2.68186	3.37212	-0.95032	2.71306	10.72936	-0.65125

```
475  0.00000  0.00000  0.00000  0.00000  0.00000  0.00000
476 -0.17787 11.84784 -2.98088  2.76155 -1.88255 -0.43021
477  3.35646 14.11881 -1.69052 -0.20537 -2.73039 -1.72409
478  4.27632 11.84784 -1.37503  1.91460 -3.76259  3.70875
479 10.66417 11.82199  7.78735 12.89118  6.38295 20.00000
480  0.00000  0.00000  0.00000  0.00000  0.00000  0.00000
481 -8.12743 -6.98576 -8.20678 -8.04346 -8.59025 -10.41400
482 -5.63774 -5.58364 -5.72640 -1.36060 -5.61268 -5.91526
483 -7.07874 -3.93936 -7.02780 -7.03211 -7.23236 -7.06380
484 -5.58101 -4.55223 -5.58035 -5.60905 -7.39758 -6.37904
485  0.00000  0.00000  0.00000  0.00000  0.00000  0.00000
486 -5.53347 -2.02334 -5.50849 -5.56283 -5.86986 -5.98502
487 -4.00639 -3.80369 -4.09974 -1.36658 -4.89503 -6.51161
488 -6.48778 -0.88615 -6.43334 -6.19279 -7.66942 -7.77423
489 -8.47890 -8.30601 -8.48330 -6.48235 -10.42848 -8.67645
490  0.00000  0.00000  0.00000  0.00000  0.00000  0.00000
491 -7.31271 -7.28256 -7.31846 -4.00839 -7.66952 -7.94136
492 -3.90486 -3.97669 -3.92068  5.45418 -3.92387 -4.80575
493 -1.12633 -0.31986 -1.93178 10.39410 -4.19481 -2.89810
494 -3.07199 -2.77049 -0.74639  8.67047 -4.77336 -5.03012
495  0.00000  0.00000  0.00000  0.00000  0.00000  0.00000
496 -2.54843 -1.83645 -2.80767 -2.56428 -4.37665 -3.69641
497 -1.32717 -0.32512 -1.30736 -1.32593 -4.22104 -4.31561
498 -2.19103 -2.20179 -1.59714  3.65542 -4.35063 -5.31645
499  3.28253 -0.19990  1.95389 15.28111 -1.75885 -0.83199
```

Recall that there are 500 discrete states since there are 25 taxi positions, 5 possible locations of the passenger (including the case when the passenger is in the taxi), and 4 destination locations.

The Q-values in the table show how useful each action is expected to be when the agent is in a certain state. Higher values suggest that an action is likely to lead to better outcomes. You can see that some actions have particularly high or low values, which tells us what the agent has learned to prefer or avoid.

2.5.1 Analysis and Visualizations

ANALYSIS

These configuration for a reinforcement learning Taxi-v3 environment play a critical role in determining the efficiency and effectiveness of the learning process.

Total Episodes: More episodes give the agent ample opportunity to explore different strategies, ensuring it learns the most effective paths and actions needed to navigate the environment successfully. This depth of experience is invaluable for the agent to understand the full scope of the state-action space.

Max Steps per Episode: Setting a reasonable limit on steps per episode is essential to avoid infinite loops, ensuring that each session has enough room for the agent to complete its mission of picking up and dropping off passengers efficiently.

Learning Rate: It controls how quickly the agent integrates new information impacting how rapidly the Q-values in the Q-table are updated.

Discount Factor: Discount factor helps the agent to evaluate the long-term outcomes of its actions, by weighting the importance of future rewards, fostering a strategy that looks beyond immediate gains for greater eventual success.

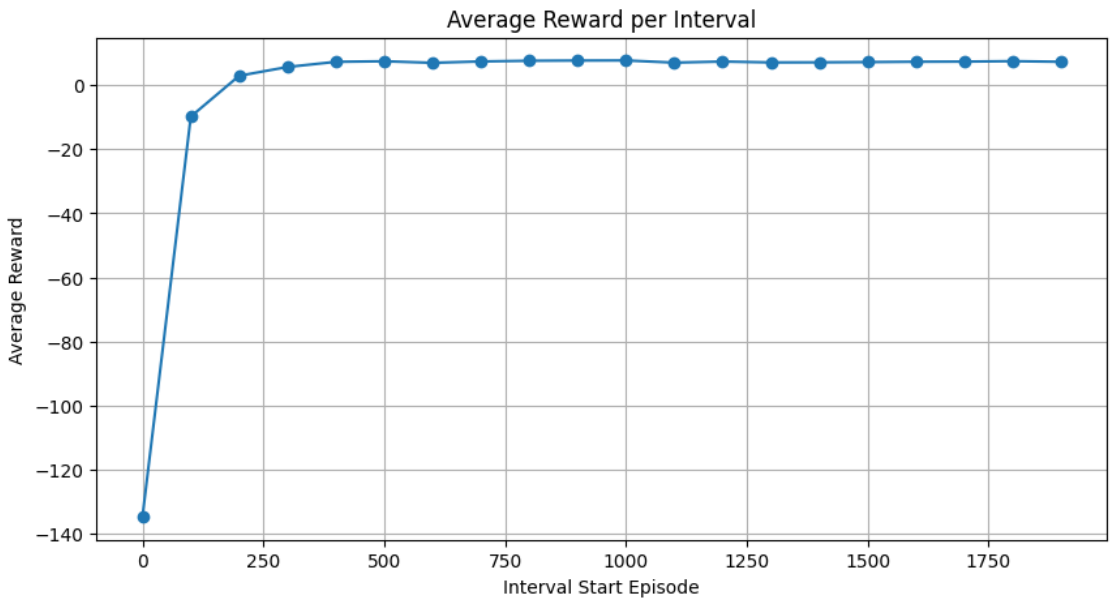
Exploration Parameters: Initially high exploration rates prompt the agent to experiment with different approaches, a critical phase for learning in Taxi-v3's complex and varied environment. Over time, fine-tuning the balance between exploration and exploitation is necessary to refine the agent's strategies towards optimal performance.

These hyperparameters are not just settings but are instrumental tools that guide the learning trajectory of the Taxi-v3 agent. Through strategic adjustment and careful monitoring, they enable the agent to master an effective balance of exploration and exploitation. Ensuring these parameters are optimally set helps the agent to learn effectively as observed above.

VISUALIZATIONS

[ ]:
```python
import matplotlib.pyplot as plt
import pandas as pd

def process_rewards(rewards, total_episodes, interval=100):
    #Process rewards to calculate the average over specified intervals
    avg_rewards = [np.mean(rewards[i:i + interval]) for i in range(0,␣
 ↪total_episodes, interval)]

    #Create a DataFrame for easy plotting
    intervals = range(0, total_episodes, interval)
    df = pd.DataFrame({
        'Interval Start': intervals,
        'Average Reward': avg_rewards
    })

    return df

def plot_rewards(df):
    # Plot the average rewards over intervals
    plt.figure(figsize=(10, 5))
    plt.plot(df['Interval Start'], df['Average Reward'], marker='o')
    plt.title('Average Reward per Interval')
    plt.xlabel('Interval Start Episode')
    plt.ylabel('Average Reward')
    plt.grid(True)
    plt.show()

#Define the interval
interval = 100

#Process rewards and plot
```

```
df_rewards = process_rewards(rewards, total_episodes, interval)
plot_rewards(df_rewards)
```



The training regimen appears to be highly effective, with the agent achieving and maintaining a high level of performance after initial learning. The graph implies that the exploration decay rate is set appropriately to reduce exploration as the agent becomes more competent. This ensures that by the time the agent has learned effective strategies, it does not deviate too much into less optimal actions, maintaining a high average reward.

```
[ ]: def qtable_directions_map_taxi(qtable, num_locations=25):
         """Get the best learned action & map it to arrows for Taxi-v3 environment.
      ↪"""
         directions = {0: "↓", 1: "↑", 2: "→", 3: "←", 4: "Pickup", 5: "Dropoff"}
         qtable_val_max = qtable.max(axis=1)
         qtable_best_action = np.argmax(qtable, axis=1)
         qtable_directions = np.empty(qtable_best_action.shape, dtype=object)

         #Mapping the best actions to directions
         for idx, action in enumerate(qtable_best_action):
             if qtable_val_max[idx] > np.finfo(float).eps:  #Checking for meaningful␣
      ↪Q-values
                 qtable_directions[idx] = directions[action]
             else:
                 qtable_directions[idx] = ""  #Empty string for unlearned actions

         #Reshape for visualization if possible
```

```
    qtable_directions = qtable_directions.reshape((-1, 5))  #Grid layout for the taxi positions
    return qtable_directions


qtable_directions = qtable_directions_map_taxi(Q_value)
print(qtable_directions)
```

```
[['' 'Pickup' 'Pickup' 'Pickup' '']
 ['' '' '' '↓' '']
 ['' '↓' '' '' '']
 ['' 'Dropoff' '↓' '↓' '↓']
 ['' '←' '←' '←' '']
 ['' '' '↓' '↓' '']
 ['' '↓' '' '↓' '']
 ['' '←' '↓' '←' '↓']
 ['' '↓' '↓' '' '→']
 ['' '→' '→' '↓' '']
 ['' '↓' '↓' '→' '→']
 ['' '↓' '→' '' '→']
 ['' '' '' '' '→']
 ['' '→' '→' '↓' '']
 ['' '' '↓' '↓' '↓']
 ['' '' '→' '↓' '↓']
 ['' '' '' '' 'Pickup']
 ['' 'Pickup' 'Pickup' '' '']
 ['' '' '' '←' '']
 ['' '↓' 'Dropoff' '←' '←']
 ['' '↑' '↑' '↑' '']
 ['' '' '' '↓' '↓']
 ['' '↓' '↓' '↓' '↓']
 ['' '↑' '→' '↓' '↓']
 ['' '←' '←' '←' '↓']
 ['' '↓' '↓' '↓' '↓']
 ['' '←' '↓' '↓' '↓']
 ['' '↑' '↓' '↓' '↓']
 ['' '↓' '↓' '↓' '→']
 ['' '→' '→' '↓' '↓']
 ['' '↓' '↓' '→' '→']
 ['' '↓' '→' '↓' '↓']
 ['' '←' '←' '↓' '↑']
 ['' '↑' '↑' '↓' '↓']
 ['' '←' '↓' '↓' '↓']
 ['' '←' '→' '↓' '↓']
 ['' '' '←' '↓' '↑']
 ['' '↑' '↑' '↓' '']
 ['' '' '↓' '←' '←']
```

```
['' '↓' '↑' '←' '←']
['' '↑' '↑' '↑' '→']
['' '→' '→' '↓' '↓']
['' '↓' '→' '→' '→']
['' '↑' '→' '↓' '→']
['' '↑' '←' '↑' '→']
['' '→' '→' '←' '←']
['' '←' '→' '→' '→']
['' '←' '→' '←' '→']
['' '←' '←' '←' '↑']
['' '↑' '→' '←' '←']
['' '←' '→' '→' '→']
['' '←' '→' '←' '→']
['' '←' '←' '←' '↑']
['' '↑' '↑' '←' '←']
['' '←' '↓' '↓' '↓']
['' '←' '↑' '←' '↓']
['' '←' '←' '←' '↑']
['' '↑' '↑' '←' '']
['' '' '←' '←' '←']
['' '←' '↑' '←' '←']
['' '↑' '↑' '↑' '']
['' '↑' '↑' '↓' '↓']
['' '↓' '↑' '↑' '↑']
['' '↑' '↑' '↓' '↑']
['' '↑' '↑' '↑' '↑']
['' '↑' '↑' '↑' '↑']
['' '↑' '↑' '↑' '↑']
['' '↑' '↑' '↑' '↑']
['' '←' '↑' '' '↑']
['' '↑' '↑' '←' '↑']
['' '←' '↑' '↑' '↑']
['' '↑' '↑' '↑' '↑']
['' '↑' '↑' '↑' '↑']
['' '↑' '↑' '↑' '↑']
['' '↑' '↓' '↓' '↓']
['' '↑' '↑' '↑' '↓']
['' '' '' '↑' '↑']
['' '↑' '↑' '↑' '']
['' '' '←' '←' '←']
['' '←' '↑' '←' '←']
['' '' '↑' '' '']
['' '' '' 'Pickup' 'Pickup']
['' 'Pickup' '' '' '']
['' '↑' '↑' 'Dropoff' '↑']
['' '↑' '↑' '↑' '']
['' '' '' '↑' '']
['' '↑' '' '↑' '']
```

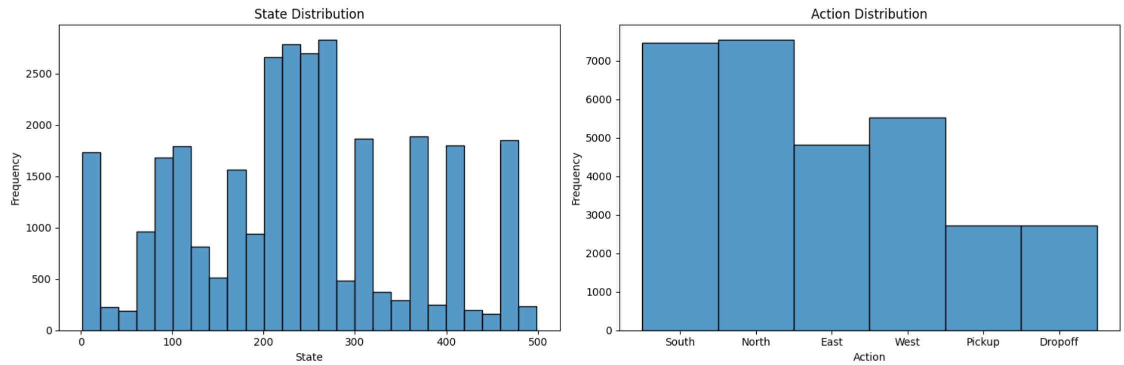
```
['' '' '' '' '']
['' '' '' '' '']
['' '' '↑' '←' '']
['' '' '↑' '↑' '']
['' '' '' '' '']
['' '' '↑' '' '']
['' '' '↑' '↑' '']
['' '↑' 'Pickup' 'Pickup' 'Pickup']
['' '↑' '↑' '↑' 'Dropoff']
['' '' '' '' '']
['' '' '' '' '']
['' '' '←' '←' '←']
['' '' '' '←' '←']]
```

It checks if the maximum Q-value for each state is significant to determine if an action has been effectively learned. The function then maps these actions to human-readable directions like arrows for movement and labels for pickup and dropoff, reshaping the output to a 2D grid that matches the taxi environment's layout, facilitating easier visualization of the learned policy.

Plot to understand distribution of states and actions for Taxi-v3 environment

[ ]:
```
import seaborn as sns
def plot_states_actions_distribution_taxi_v3(states, actions):
    #Labelling actions
    action_labels = {0: "South", 1: "North", 2: "East", 3: "West", 4: "Pickup",␣
 ↪5: "Dropoff"}
    #Creating a subplots for action and state
    fig, ax = plt.subplots(nrows=1, ncols=2, figsize=(15, 5))

    #Plotting state distributions, assuming states are integers representing␣
 ↪taxi positions
    sns.histplot(data=states, ax=ax[0], kde=False, bins=25)
    ax[0].set_title("State Distribution")
    ax[0].set_xlabel("State")
    ax[0].set_ylabel("Frequency")

    #Plotting action distributions
    sns.histplot(data=actions, ax=ax[1], discrete=True)
    ax[1].set_xticks(list(action_labels.keys()))
    ax[1].set_xticklabels(action_labels.values())
    ax[1].set_title("Action Distribution")
    ax[1].set_xlabel("Action")
    ax[1].set_ylabel("Frequency")

    fig.tight_layout()
    plt.show()
```

```
plot_states_actions_distribution_taxi_v3(all_states, all_actions)
```



The plot displays two histograms: one showing the distribution of states and the other showing the distribution of actions within a Taxi-v3 environment.

State Distribution Histogram: The uneven distribution implies that certain states are encountered more often than others, suggesting there are specific areas in the environment where the taxi either frequently travels or tends to get caught.

Action Distribution Histogram: The graph shows a clear preference for certain directions, with 'South' and 'North' actions being the most frequent. The lesser frequency of 'Pickup' and 'Dropoff' actions reflects their specific use cases, only applicable when the taxi is in the correct position to perform these tasks.

2.6 Step 5: Q-Learning Agent uses our Q-table to drive the *Taxi*

Defining a policy function to select actions using a Q-table and simulate a series of episodes in the Taxi-v3 environment to assess the policy's effectiveness. It tracks successful passenger drop-offs and displays each episode's outcome, tallying the success rate over 5 episodes, showcasing the trained agent's ability to execute the task within the environment.

[ ]:
```python
#Define the policy using the Q-table
def policy(state):
    return np.argmax(Q_value[state, :])

n_episodes = 5
max_steps = 99
success = 0  #Number of successful attempts

for episode in range(n_episodes):
    step = 0
    done = False
    state = env.reset()[0]

    print(f"EPISODE #{episode+1}\n")
```

```python
    for step in range(max_steps):
        #Use the current policy to determine the next action
        action = policy(state)

        #Update the environment based on the action taken
        new_state, reward, done,truncated, info = env.step(action)

        env.render()  #Optional: to visually see the taxi environment in action

        if done:
            if reward == 20:  #In Taxi-v3, a reward of 20 indicates successful␣
 ↪drop-off
                success += 1
                print("Successfully dropped off the passenger   ", end="")
            else:
                print("Failed to drop off the passenger   ", end="")

            #We print the number of steps it took.
            print(f"after {step+1} steps.\n")
            break
        #Update state
        state = new_state
#Print the success or failure
print(f"\nSuccess rate: {success} out of {n_episodes} episodes.\n")
env.close() #close the environment
```

```
WARNING:py.warnings:/usr/local/lib/python3.10/dist-
packages/gymnasium/envs/toy_text/taxi.py:314: UserWarning: WARN: You are
calling render method without specifying any render mode. You can specify the
render_mode at initialization, e.g. gym.make("Taxi-v3",
render_mode="rgb_array")
  gym.logger.warn(


EPISODE #1

Successfully dropped off the passenger   after 17 steps.

EPISODE #2

Successfully dropped off the passenger   after 13 steps.

EPISODE #3

Successfully dropped off the passenger   after 15 steps.
```

```
EPISODE #4

Successfully dropped off the passenger  after 13 steps.

EPISODE #5

Successfully dropped off the passenger  after 15 steps.


Success rate: 5 out of 5 episodes.
```

The performance of a reinforcement learning agent in the Taxi-v3 environment demonstrates a perfect success rate in a test of five episodes. These results validate the chosen hyperparameters setup.

3 Notes, observations, and conclusions

OBERVATIONS

I have observed that Q-learning process in the Taxi-v3 environment exhibits stochastic behavior which means that the results can vary with each training session due to the randomness inherent in the agent's actions and the environment. Therefore, outcomes across different hyperparameter combinations may not be consistent in every run, but they tend to cluster around certain performance levels. This variability underscores the critical importance of understanding how hyperparameters influence learning dynamics. A thorough grasp of these settings ensures that the training process can be fine-tuned to achieve reliably near-optimal results, despite the inherent randomness in the learning process.

HYPERPARAMETERS TUNING - Try different hyperparameter settings to determine what you observed are the best or better settings.

Let's analyze how different combination of the hyperparameters used in in the Taxi-v3 environment in detail to understand how each parameter setting contributes to the agent's average reward score, success rate, learning efficiency and overall performance.

Combination 1:

Settings:

- Total Episodes: 500
- Maximum steps:100
- Learning Rate: 0.1
- Discount Factor: 0.9

Results:

- Average Reward Score: -100.78
- Success Rate: 0/5

Observations: Due to the lower number of total episodes (500) and a moderate discount factor (0.9) the environment was not able to sufficiently explore and learn about the optimal strategies. The high negative score and zero success rate indicate that the agent was unable to effectively learn the task within the given episodes, suggesting undertraining.

Combination 2:

Settings:

- Total Episodes: 1000
- Maximum steps:99
- Learning Rate: 0.1
- Discount Factor: 0.99

Results:

- Average Reward Score: -37.34
- Success Rate: 2/5

Observations: Doubling the number of episodes to 1000 with a higher discount factor of 0.9 and maximum steps nearly same shows improvement in both the average reward score and success rate.The higher discount factor promotes consideration of future rewards, slightly improving performance, but still indicates insufficient training or exploration.

Combination 3:

Settings:

- Total Episodes: 2000
- Maximum steps: 100
- Learning Rate: 0.1
- Discount Factor: 0.9

Results:

- Average Reward Score: -0.898
- Success Rate: 5/5

Observations: This setting strikes an optimal balance with enough episodes and a moderate learning rate, combined with a discount factor that appreciates future rewards sufficiently. The notable improvement to a near-zero average reward and a perfect success rate indicates that the agent has effectively learned to navigate the environment and perform its tasks efficiently.

Combination 4:

Settings:

- Total Episodes: 4000
- Maximum steps: 100
- Learning Rate: 0.7
- Discount Factor: 0.618

Results:

- Average Reward Score: -6.34
- Success Rate: 4/5

Observations: Although the combination 4 has double the number of episodes compared to Combination 3 (4000 vs. 2000), it still doesn't perform better. Typically, more episodes would provide more learning opportunities. However, combination 4 could have been exploring less optimally or settling into suboptimal policies due to other factors such as the high learning rate and low discount factor which indicates their significance and how much those factors matter as well than compared to just total episodes. High learning rate (0.7), speeds up the policy updates significantly. This can be advantageous in dynamic environments but risky as it may cause the policy to converge prematurely or oscillate without stabilizing. This suggests that a more aggressive learning rate does not necessarily translate to better performance.

Combination 5:

Settings:

- Total Episodes: 5000
- Maximum steps: 99
- Learning Rate: 0.1
- Discount Factor: 0.9

Results:

- Average Reward Score: 1.23
- Success Rate: 5/5

Observations: The increase from 2000 to 5000 total episodes provides the agent more opportunities to explore and exploit the environment. This can lead to a more refined understanding and optimization of the Q-values, potentially allowing the agent to discover even more efficient strategies. Both settings use the same learning rate (0.1) and discount factor (0.9), which indicates that the improvements in training outcomes are not due to these parameters but rather the adjustments in the episodes and steps. The success rate remains perfect in both settings (5/5), demonstrating that the agent reliably learns the necessary task completion strategies under both configurations.

The current combination (3) strikes an optimal balance by carefully managing the number of episodes, which minimizes the risk of both overtraining and undertraining. This setup allows for efficient and rapid training of the agent, utilizing fewer computational resources while still maintaining high reliability and achieving near-optimal performance. This makes it a preferable choice for effectively navigating the complexities of the training environment.

The initial phase of high exploration rates is essential for the agent to explore various strategies in the complex and varied Taxi-v3 environment. This exploration is crucial for the agent to learn and adapt effectively. Over time, we have fine-tuned the exploration-exploitation balance to ensure that the agent optimally refines its strategies. Because this balance has proven effective in helping the agent discover efficient paths and leverage learned behaviors, we have decided not to alter the exploration-exploitation parameters further. Changing these parameters without a clear necessity risks upsetting this carefully established balance, potentially causing the agent to engage in overly risky behavior or to cease learning new strategies effectively.

SUMMARIZING MY THOUGHTS

The Taxi-v3 environment provides an interesting playground for developing and testing Reinforcement Learning algorithms. Some insights or thoughts on the experience of Q-learning from these environments include:

Learning Environment: This environment is challenging due to its discrete state and action spaces, requiring the agent to learn from a large combination of scenarios effectively.

Hyperparameter tuning: Key hyperparameters like total episodes, steps per episode, learning rate, discount factor, and exploration rates significantly impact the training dynamics and must be properly tuned.It is essential for balancing exploration and exploitation during training to achieve efficient learning without underfitting or overfitting.

Reward System: A reward system is employed that significantly influences the agent's learning behavior. The rewards for actions like pickups, dropoffs, illegal moves, and step counts are adjusted to steer the training towards minimizing illegal actions and optimizing path efficiency.

Performance Evaluation: The success of training is assessed through average reward scores and success rates. Watching these metrics improve over time was highly satisfying as it provides tangible evidence of the agent's growing competence.

These insights provide an overall view of how Q-Learning is applied in the Taxi-v3 environment to develop an autonomous agent capable of navigating and solving the taxi passenger pickup and dropoff problem.

CONCLUSION

Consider guiding a novice taxi driver through the busy streets of a Virtual city environment. Each exploration the driver undertakes is an episode in their learning journey, with you tweaking their lessons (hyperparameters) to help them become more adept at navigating the city's challenges. Through trial and error, guided by the rewards of customer satisfaction(and perhaps a few fines for traffic violations), our virtual driver gradually transforms from a hapless newbie to a seasoned pro.

Let's buckle up and enjoy the ride, cheering for our AI-driven taxi as it learns to navigate the complexities of its digital world, delivering passengers safely to their destinations with increasing skill and fewer detours. Here's to hoping our virtual taxi driver doesn't just find the fastest route but also enjoys the journey—after all, even a digital driver should have some fun!