

```
In [141]: import gymnasium as gym
import random
import numpy as np
import matplotlib.pyplot as plt
env = gym.make("CartPole-v1", render_mode="human")
```

```

In [143]: class QLearnerSolver:
    def __init__(self, env):
        self.q_table = {}
        self.alpha = 0.1
        self.gamma = 0.95
        self.epsilon = 0.1
        self.env = env
        self.action_space = list(range(env.action_space.n))
        self.state_space_size = [3, 3, 6, 6]
        self.q_table = np.zeros(self.state_space_size + [env.action_spa

    def convert_cont_to_discrete_space(self, state):
        if isinstance(state, tuple):
            state = state[0]
            bucket_edgesv = np.linspace(-4.8, 4.8, 3)
            bucket_edgesvdot = np.linspace(-5, 5, 3)
            bucket_edgestheta = np.linspace(-0.418, 0.418, 6)
            bucket_edgestheta_dot = np.linspace(-5, 5, 6)
            bucket_indexv = np.digitize(state[0], bucket_edgesv) - 1
            bucket_indexvtheta = np.digitize(state[1], bucket_edgesvdot) - 1
            bucket_indexdot = np.digitize(state[2], bucket_edgestheta) - 1
            bucket_indexdottheta = np.digitize(state[3], bucket_edgestheta_dot)

            return tuple([bucket_indexv, bucket_indexvtheta, bucket_indexdot,

    def choose_action(self, state):
        # if the random is more than epsilon
        if random.uniform(0, 1) < self.epsilon:
            return self.env.action_space.sample()
        else:
            discretized_state = self.convert_cont_to_discrete_space(sta
            q_values = self.q_table[discretized_state]
            return np.argmax(q_values)

    def learn(self, num_episodes):
        total_rewards = []
        for learning_epoch in range(num_episodes):
            state = env.reset()
            total_reward = 0 #every episode, reset the
            for time_step in range(500):
                action = self.choose_action(state) #learner chooses one
                next_state, reward, done, _, _ = env.step(action) #the actio
                discretized_state = self.convert_cont_to_discrete_space

                # Use Q-learning update rule
                max_q_next = np.max(self.q_table[self.convert_cont_to_d
                self.q_table[discretized_state + (action,)] += self.alp

                state = next_state
                total_reward += reward

            # update total reward

```

```
        total_reward = total_reward + reward
        total_rewards.append(total_reward)
    return total_rewards
```

```
In [144]: env = gym.make('CartPole-v1')
learner1=QLearnerSolver(env)
total_rewards = learner1.learn(500)
plt.plot(range(0, 500), total_rewards)
plt.xlabel("Episode")
plt.ylabel("Episode Reward")
plt.title("Q-Learning on Bucketed CartPole Environment")
plt.show()
```

