16-662

Robot Autonomy HW4

Boxiang Fu

boxiangf

04/09/2025

Q1.

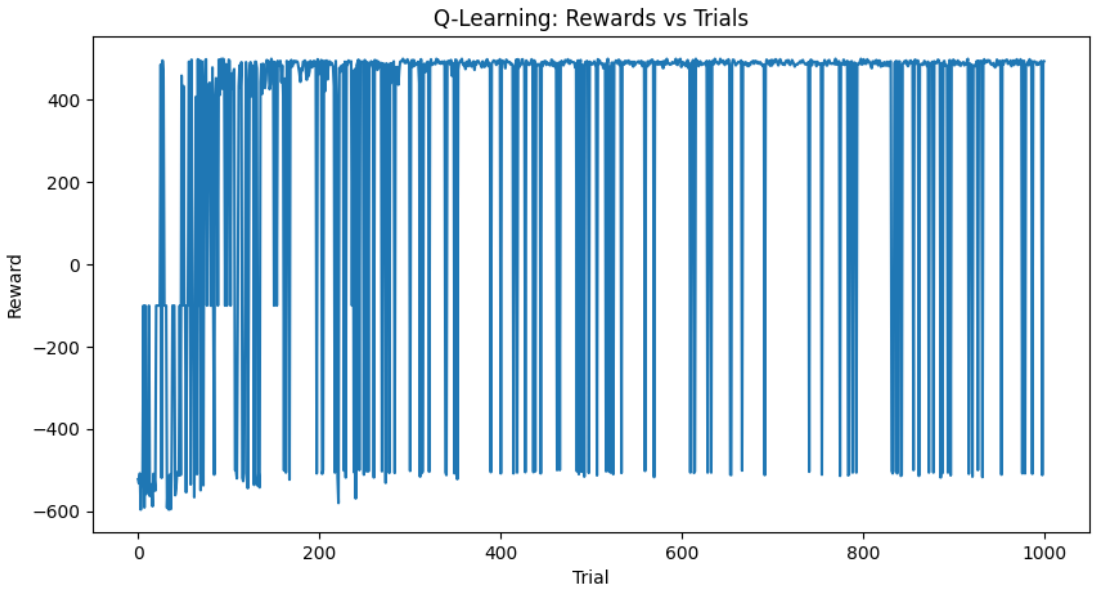
a. During training, the rewards randomly dip to a low number because the agent happens to state transition and land on a bomb location. This is a terminal condition, so the trial ends and the reward dips to a low number (approx. -500). The reason for the sporadic dip is because the implementation uses epsilon-greedy exploration. That is, with an epsilon probability, the agent goes off-policy and randomly explores an alternative action rather than the one with the maximum Q-value. This causes the agent to sometimes inadvertently choose an action and lands the agent on a bomb location. It is a part of the learning process as the agent now learns that the payoff for this action is very low and would not try this action again on-policy.

b. The arrows plotted in **q_values.png** indicate the direction of the learned best action to take at each state for the agent to maximize its cumulative rewards (i.e. the state transition to make to obtain the highest Q-value after learning). The arrows intuitively make sense as they tend to point to the gold location and away from bomb locations. This is what we expect for a sufficiently learned Q-policy as reaching the gold location in the minimum number of steps gives the highest cumulative rewards. However, we note that sometimes the arrows might not be optimal (e.g. the arrow pointing into a bomb location at (6, 0) index). This is most probably a result of insufficient learning as the map boundaries are usually traversed less. Such problems could be easily solved by increasing the number of trials.

c. No, the Q-agent does not always follow the optimal path with respect to the Q-values in the videos generated. The reason for the off-policy path is because of the non-zero **ENV_RHO** term causing random off-policy actions when **environment.step(action)** is run. This is true even when we set the explore state in **agent.get_action(state, explore=False)** is set to False. The reason for the off-policy action is to simulate a more stochastic environment in where the input control might not result in the actual desired action from that control input.

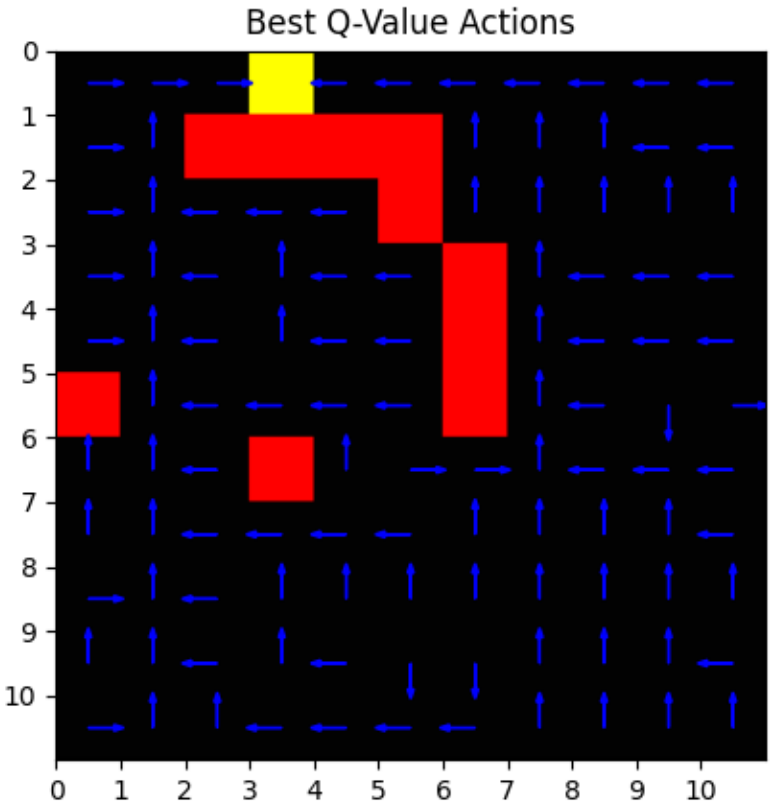
Q2.

The **q_learning_training.png** is shown below:



Q3.

The **q_values.png** is shown below:



Q4.

The average rewards obtained by the random agent and Q agent in the benchmark runs is shown below:

```
Benchmarking Random Agent for 500 trials
Average reward random agent: -435.808

Visualizing Random Agent
Saving 59 frames to visualizations/random_agent_0.mp4
Finished after 58 steps with total reward of -557.0
Saving 26 frames to visualizations/random_agent_1.mp4
Finished after 25 steps with total reward of -524.0
Saving 6 frames to visualizations/random_agent_2.mp4
Finished after 5 steps with total reward of -504.0

Training Q Agent for 1000 trials

Benchmarking Q Agent for 500 trials
Average reward Q agent: 485.256

Visualizing Q Agent
Saving 14 frames to visualizations/q_agent_0.mp4
Finished after 13 steps with total reward of 488.0
Saving 14 frames to visualizations/q_agent_1.mp4
Finished after 13 steps with total reward of 488.0
Saving 8 frames to visualizations/q_agent_2.mp4
Finished after 7 steps with total reward of 494.0
```

Average rewards for random agent: **-435.808**

Average rewards for Q-agent: **485.256**

Q5.

a. Q_ALPHA:

Empirically, increasing the alpha value to a certain extent causes the learning to occur faster and makes the Q-agent find the gold location in a smaller number of epochs. However, once the alpha value is increased dramatically (i.e. experimentally over 0.9), the agent's behavior becomes unstable and sometimes causes the agent to remain stuck in a loop.

The reason for this is because alpha controls how much new experiences override old Q-values. A large alpha favours recent experiences and learns fast from new rewards, whereas a low alpha averages the Q-values over time and results in a more stable value. However, if the alpha is too high, the Q-value becomes unstable and fluctuates a lot since only the information from the previous iteration is saved; whereas if the alpha is too low, the learning is very slow and requires many epochs to converge.

Average rewards for Q-agent with Q_ALPHA = 0.1: **485.256**

Average rewards for Q-agent with Q_ALPHA = 0.99: **-373.584**

b. Q_GAMMA:

Empirically, setting the gamma value to a high value (i.e. near 1) causes the agent to reach the gold location more often. Setting the gamma value to a low value causes the agent to be more likely to remain stuck in a loop.

The reason for this is because gamma controls how much the agent cares about future rewards. For a low gamma value, the agent is short-sighted and prefers small rewards in its vicinity rather than taking risks exploring for far away greater rewards (which quickly gets discounted away due to gamma); hence causing it to remain stuck in a loop. For a high gamma value, the agent is more far-sighted and takes a greater risk exploring further away to the big +500 gold reward. This is because the payoff of this reward is not discounted as much and offsets the penalties taken to path towards this location. For a relatively sparse reward map as the one in the assignment, a high gamma value causes the agent to look-ahead more and achieve a higher chance of actually finding the gold location.

Average rewards for Q-agent with Q_GAMMA = 0.99: **485.256**

Average rewards for Q-agent with Q_GAMMA = 0.01: **-114.906**

c. Q_EPSILON:

Empirically, increasing the epsilon value causes the agent to explore more random actions during the exploration phase, whereas decreasing the epsilon value causes the agent to focus more on the current best action during the exploration phase. In the extreme cause of epsilon being zero, the model will not train as no new paths are being explored.

The reason for this is because epsilon controls how often the agent goes off-policy during the training phase and explores random actions instead of using the on-policy known best action. A high epsilon value helps with exploring more terrain, but would not converge as fast due to the high stochasticity in actions. A low epsilon value causes the agent to explore very little terrain, and may result in the agent being stuck in a local optimum and not converge to the global optimum of reaching the gold location.

Average rewards for Q-agent with Q_EPSILON = 0.1: **485.256**

Average rewards for Q-agent with Q_EPSILON = 0.99: **27.966**

d. ENV_RHO:

Empirically, increasing the env_rho value causes the environment to become more random and less predictable. Even if the agent chooses an action, the environment may cause the agent to follow through a different action. Decreasing env_rho causes the environment to become more deterministic and follow through the on-policy action chosen.

The reason for this is because env_rho controls how often the agent goes off-policy and chooses a random action instead of using the on-policy known best action. This stochasticity is regardless of training or runtime. A high env_rho value causes a higher chance for random actions to be chosen, and vice versa. For our deterministic environment, setting a low (or zero) env_rho value is more suitable. However, for more dynamic environments, a high env_rho value causes the agent to be more general and be robust to unexpected scenarios.

Average rewards for Q-agent with ENV_RHO = 0.01: **485.256**

Average rewards for Q-agent with ENV_RHO = 0.99: **-445.186**