16-662 Autonomy Homework 4

David Hill

27 April 2024

1 Q-Learning Training

Figure 1 displays the reward of each successive episode of training plotted over time. While training, the rewards randomly dip to a low number because the agent took a very non-optimal path to reach the goal in that episode. These dips are actually good though, because the agent learns quickest when it fails dramatically. As the q-learning algorithm converges to an optimal solution, these dips should disappear. As seen in Figure 1, they have not completely disappeared by the end of training, suggesting that the solution is not quite optimal.

The final Q-learning policy is shown in Figure 2. The arrows point in the direction that the policy has determined will result in the greatest value. In general, the arrow directions make sense, moving towards the gold square while avoiding the red squares.

While the arrows generally make sense, they are not always pointing in the most optimal direction. This non-optimality is due to the fact that the value of taking a few extra steps is only slightly smaller than the optimal path. Thus, it would take more iterations for the Q-learning algorithm to differentiate between all of the small advantages of the optimal path. Additionally, even when the trained Q agent makes an optimal decision, the actions are probabilistic, so there is a probability that the action does not result in the expected state. This is just another reason why the agent may take unexpected routes in the space.

The random agent gets an average reward of -2960.52 while the Q agent gets an average reward of 446.45.

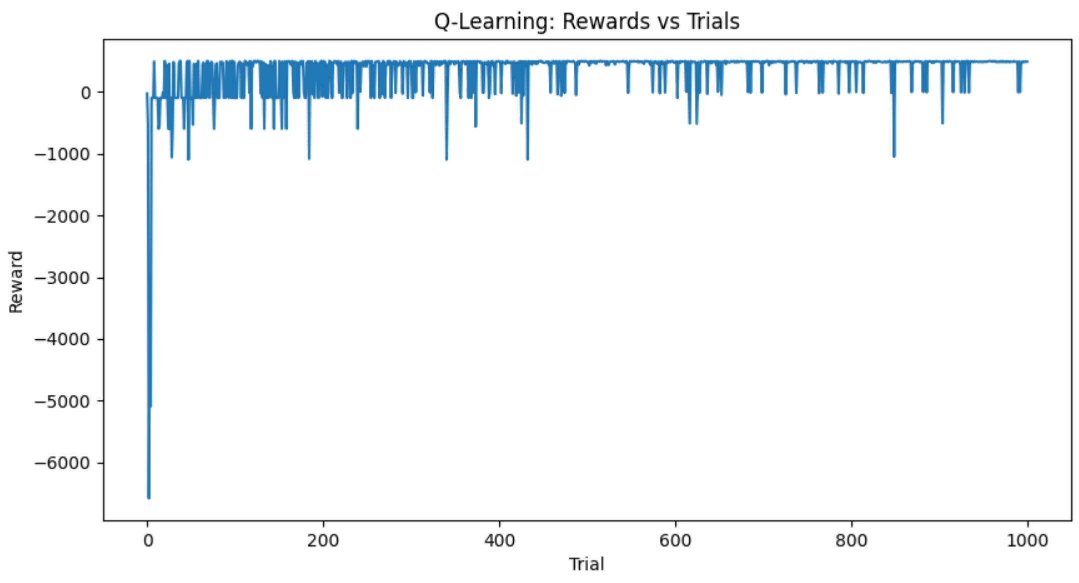

Figure 1: Cumulative reward of each episode during Q-learning training.

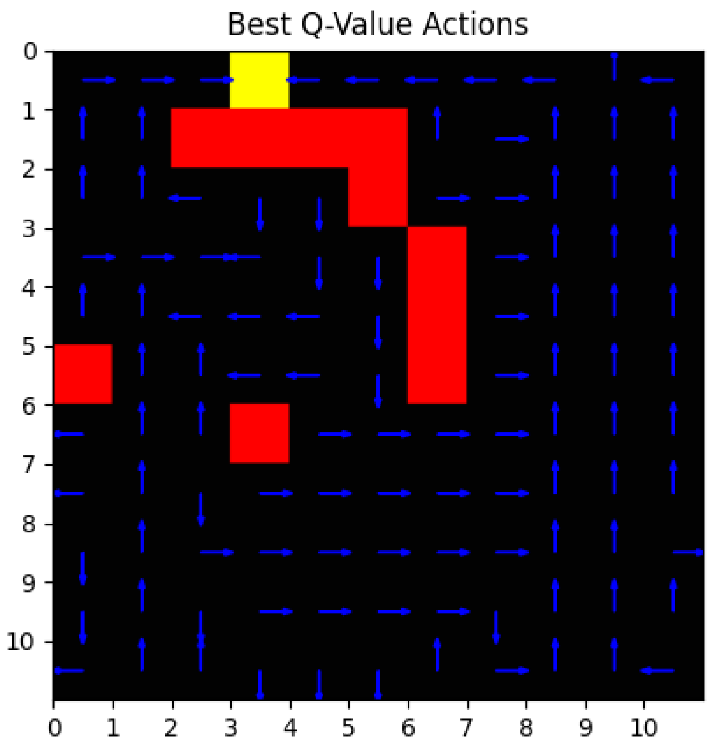

Figure 2: Final Q-learning policy represented as arrows pointed in the direction with the highest value.

2 Q-Learning Hyper-Parameters

There are four major hyper-parameters used in this Q-learning algorithm: alpha, gamma, epsilon, and rho. Alpha is the learning rate and may be the most important to tune. If alpha is too small, the q-learning training will take too many iterations. If alpha is too large, the training will forget past results too quickly causing the q matrix to jump around in value and never converge.

Gamma is the discounting value. This should be close to 1. If it is too small, the future will not matter enough and the q matrix value will be too myopic. Making gamma small will result in more circular and indecisive behavior because it will ignore the big picture more and more.

Epsilon is the probability that the agent will take a random action instead of a greedy action during training. A large epsilon will help cover the action space faster, but if it is too large it will make episodes take longer and cause the values in the q matrix to be drastically different from the optimal value. One approach that would theoretically work well is starting with a large epsilon and moving to a smaller epsilon as you go into later episodes. This would theoretically enable you to cover the whole action space early, but then converge to the optimal values later.

Rho is not really a hyper-parameter you would choose, but rather is determined by the problem. Rho represents the probability that an action will "fail" and result in a random outcome. This is what makes all actions intrinsically have a probability distribution for their effect. If rho is large, it will be hard to determine the optimal path, but the optimal path will also be further away from obstacles (red squares) because travelling right next to them increases the likelihood that you will randomly walk into one.