Assignment1 - Defining and Solving RL Environments

Saeyeon Hwang

March 10, 2022

1 Defining RL environments

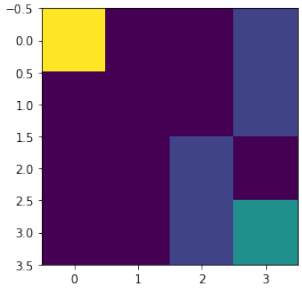
1.1 Describe the deterministic and stochastic environments, which were defined (set of actions/states/rewards,main objective, etc)

Deterministic states are consist of 4x4 states. The actions are consist of 4, down, up, left, right. The rewards are also consist of 4. The coordinates of the rewards are [0,3], [1,3], [2,2], [3,2], and they have 1,2,3,4 values each.

Stochastic environment are based on deterministic environment. So the states, actions, and rewards are same with the deterministic environment. The only difference is that action is decided based of probability.

The objective of two environments(deterministic, stochastic) are finding optimal policy which maximizes the return(rewards).

1.2 Provide visualizations of your environments



4x4 states, yellow = agent position, blue = reward positions, mint = goal position

1.3 How did you define the stochastic environment?

The mentioned difference of deterministic and stochastic environment was the action of stochastic environment is decided based on probability. The actions, 0,1,2,3, mean down, up, right, and left each. If action is 0(down), then 55% probabilities will be given to action 'down', and the rest will have 15% probabilities each.

1.4 What is the difference between the deterministic and stochastic environments?

In deterministic policy, the action must be mapping by the current state. The formula is a = $\pi(s)$.

In stochastic policy, the action will be selected stochastically by the current state. The formula is a = π(a|s). The added values of probabilities are 1.

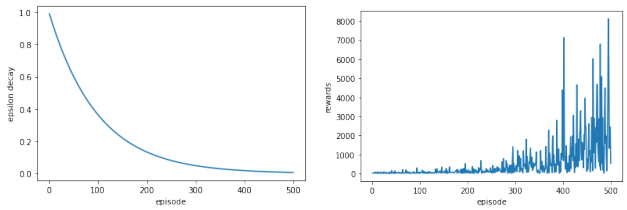
1.5 Safety in AI

I ensure the safety of my environments by using numpy clip method. If I specify the minimum and maximum values in the argument, I could prevent out-of-range values. I assigned minimum 0 and maximum 3 values, so if the number is out of the range, I made the number would be replaced with those numbers.

This is the more specific explanation why we should use clip method. If the agent is in the [3,3] position, the agent cannot go down or right. So by specifying the maximum value to 3, we could replace out-of-range numbers to 3. On the other hand, if the agent is in the [0,0] position, the agent cannot go up or left. So we could replace out-of-range numbers to 0 by specifying the minimum value to 0.

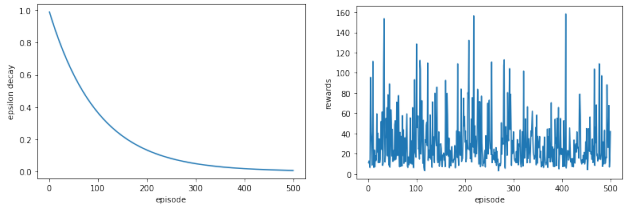
2 Applying Tabular Method

2.1 Applying Q-learning to solve the deterministic environment defined in Part 1. Plots should include epsilon decay and total reward per episode.



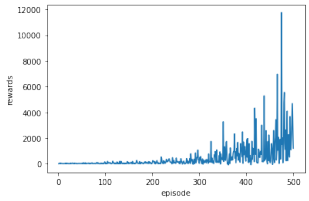
The epsilon is decaying per episode because the formula was epsilon = epsilon * decay. The decay value was 0.9, and epsilon started with 1. And the rewards per episode are increasing. The second graph shows the effects of epsilon decay. If random value is lower than epsilon value, the model exploit. On the other hand, if random value is higher than epsilon value, the model explore. At the first time, the epsilon value starts with 0.9, so there would be less possibilities to exploit, but as epsilon decreases, the possibilities of exploitation would increase. So the model could choose more optimal action.

2.2 Applying Q-learning to solve the stochastic environment defined in Part 1. Plots should include epsilon decay and total reward per episode.



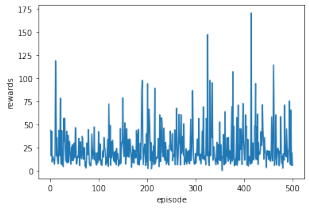
The epsilon is decaying per episode because the formula was epsilon = epsilon * decay. The decay value was 0.9, and epsilon started with 1. And the rewards per episode are increasing. That means the model is well training. The reason that second graph is different with 2.1, is because the environment is non-deterministic. So the rewards starts with higher than deterministic environment, but it looks like non-linear.

2.3 Applying Q-learning to solve the stochastic environment defined in Part 1. Plots should include epsilon decay and total reward per episode.



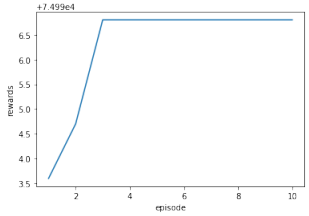
SARSA in deterministic environment is similar with the algorithm Q-learning in deterministic environment. The only difference is Q-learning is updated by the max values of whole action, but SARSA is updated by the next action.

2.4 Applying any other algorithm of your choice to solve the stochastic environment defined in Part 1. Plots should include total reward per episode.



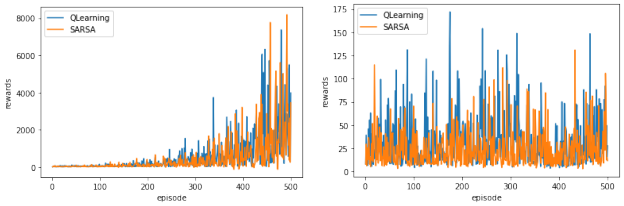
SARSA in stochastic environment is similar with the algorithm Q-learning in stochastic environment. The only difference is Q-learning is updated by the max values of whole action, but SARSA is updated by the next action.

2.5 Provide the evaluation results. Run your environment for at least 10 episodes, where the agent chooses only greedy actions from the learnt policy. Plot should include the total reward per episode.



This is the evaluation result of q-learning in deterministic environment. I run it for 10 episodes, and only chose greedy actions from the learnt deterministic q-learning model. So when you look at the graph, you can find out the rewards quite increase linearly over episodes. (Tried the evaluation to whole models but deterministic q-learning got the best result)

2.6 Compare the performance of both algorithms on the same deterministic, stochastic environment (e.g. show one graph with two reward dynamics) and give your interpretation of the results.



The performance of Q-learning is quite better than SARSA both in deterministic, stochastic environment. This is because of the difference between two models. As mentioned, Q-learning is updated by the max values of whole action, but SARSA is updated by the next action. So Q-learning could get higher rewards than SARSA.

2.7 Briefly explain the tabular methods, including Q-learning, that were used to solve the problems. Provide their update functions and key features.

Tabular method is used when the space is discrete enough to record the value function in the table. Q-learning, SARSA are both tabular method. So in the code, we use Q-table to recorded the states.

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha(r_{t+1} + \gamma \max_{a'} Q(s_{t+1}, a') - Q(s_t, a_t))$$

In Q-learning, by each step of episode, model would choose action which is decided by epsilon greedy values. Then the model could take action, and get reward and next state. So now the q-learning model has state, action, next state and reward. By these four values, we could update Q table. Q table is updated by the sum of immediate reward and the max value of next state. The update rule is followed by the above function

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha\left[R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)\right]$$

In SARSA, by each step of episode, model would choose action which is followed by epsilon greedy policy. Then the model could take action, and get reward and next state. Next, SARSA choose next action which is also followed by epsilon greedy policy. So now SARSA model has state, action, next state, next action and reward. By these five values, we could update Q table. Q table is updated as same way with Q-learning. However, SARSA is updated by q(next state, next action) instead of max q(next state, action).