Ege Dinçer
60190

Q1:

Because our agent becomes deterministic through the Markov Decision Process. Since the stochasticity is disabled through the process agent acts precisely, making it a reflex agent. It's called offline planning because agent doesn't try and learn things in real-time. It estimates all the grid's values, where agent knows the grid, and then starts to act. Therefore, this is called offline planning.

Q2:

I have changed the noise parameter which causes agent to do explore the grid once in a while. I have disabled that in order to make agent not explore the grid as the grid doesn't have any places that require to be explored and exploration has huge costs in this instance.

Q3:

For the first parameters I have used 0 Noise and a low discount parameter as 0.2. Because I wanted agent to get to the closest exit due to having a low discount parameter this is possible. I also wanted it to stay away from the longest path which is the exploration part. With these parameters it directly wanted to go the shortest exit. For the second parameters I have just slightly increased the noise while discount remaining so that the agent will avoid the cliffs and try the upper path. For the third and fourth in order to make the agent consider the last exit I have used larger discounts as 0.9. I have used 0.2 and 0.3 noises respectively. 0.2 is for agent to continue along near the cliff to the last exit. 0.3 is for the longer path. For the last everything was 0 to make agent move without reaching anything basically making it in an infinite loop. Other's share -1 living rewards so that the agent will try to reach the exit.

Q4:

The first line of code refers the value iteration which does the offline learning, so the agent learns the values of the states before moving anywhere. The second line of code refers the q-value learning which the agent learns the q values real-time, so the values start to appear as agent moves and explores through the actions The q value learning also calculates the best actions to be done from the current state that the agent is in but the value iteration only tells the values of the states, it doesn’t explicitly indicate the best actions to be done from that state.

Q5:

There is no epsilon and learning value to satisfy to find the optimal policy to be learned after 50 iterations. Because in each case the agent tends to do not optimal moves through the grid. I have tried many different combinations as high learning rate low epsilon vice versa for in case however, the agent is constantly stuck between the two states. Sometimes it tried the third state, yet it didn’t find the optimal policy.

Q6:

Because in large grid’s Q-learning becomes inefficient to find the optimal policy. A simple Q-learning algorithm wasn’t scalable for the larger grids. We would require a lot of training episodes for the algorithm to work on near optimality as it to learn the policy. It has to explore the area through many iterations which gets near impossible to be completed in a finite amount of time as the grid gets larger and larger. Therefore, the tabularized q-learning doesn’t work for the larger grids.

Q7:

Initially the pacman tries to reach for the food as its main objective with the least number of steps. It’s the priority however when a ghost gets near to the pacman, its priority changes as pacman tries to avoid from the ghost as its priority and getting the food becomes it’s second priority. Therefore, this means that at the same distance like if the next step of the pacman is taking the food but the ghost will hunt it with this step, pacman will avoid doing that action. This is how it was implemented in the featureExtractors.py and pacman exactly behaves in this manner.