Assignment 3 - COMP3106
Fareha Sultan
100968491

1. Briefly describe how your implementation works. Include information on any important algorithms, design decisions, data structures, etc. used in your implementation. [10 marks]

My implementation closely follows the methods and steps that were used in class to solve problems involving temporal difference Q-Learning. For this reinforcement learning assignment, it starts by reading the file with the trial data which is saved into an array where the first value is the state (the position) and the second the action taken.

Next, I loop through all the squares the mouse,M, can be in which has an inner loop of all the squares the cat, C, can be in. The MC are stored as the keys in a q-value dictionary where the values are all the squares the Mouse, M ,can take from its current square. The values themselves are a dictionary which have the keys as actions up (U), down (D), left(L), right(R) and do not move (N) and initial value of 0.

To perform temporal difference Q-learning , I loop through the trial data, collect all the data I need for the Q-function which I called calculate(). We are given the alpha, the learning rate , with a value of 0.2 and gamma, discount factor, with a value of 0.9. The reward, r(s), is calculated using the reward function which returns a 1 when Mouse and Cat are in different squares and returns a -1 when both are in the same square. From the trial, we can get the values for Q(s',a'), the subsequent state and take the max action for that state. We pass in all the parameters to the function calculates() which uses the following formula:

Q(s,a) <- Q(s,a) + alpha * (r(s) + gamma * max_a'(Q(s',a')) - Q(s,a))

It returns the q value that we use to update the dictionary qvalues for that action.
This keeps iterating until we reach the second last trial element. At this point, all values have been updated and we can call the qvalue and privacy function to output the results.
The qvalue function basically returns the value from the qvalues dictionary stored at the given state and action. The privacy function returns the action that gives the max value from all actions at a given state.

2. What type of agent have you implemented (simple reflex agent, model-based reflex agent, goal-based agent, or utility-based agent)? [3 marks]

Q(s,a) represent the value of a state given we take the action in it. It calculates the long -term reward for state-action pair. It is also sometimes called the “action-utility” function which means that it is a utility based agent, it predicts the long-term rewards based on the current state S given a transition state S’. It can not be a simple reflex agent because it keeps track of the previous q-values for states-actions and uses the max value which maximizes its utility (optimizes the q-value)

3. Describe, at a high-level, the optimal strategy for the mouse (note that your implemented agent might not actually learn the optimal strategy through temporal difference Q-learning). [5 marks]

We can implement SARSA (state-action-reward-state-action) learning for the mouse. It will iteratively estimate the Q(s,a) from subsequent state-action each time an action was taken to transition from s to s' (subsequent state). The Bellman equation will include the actual action , a', that was taken in the subsequent state-action pair.

$$Q(s,a) \leftarrow Q(s,a) + \alpha\left(r(s) + \gamma Q(s', \underset{\text{the actual a'}\uparrow}{a'}) - Q(s,a)\right)$$

SARSA backs up the q-value as it is the actual state-action pair that is taken and it is also "on-policy". The policy will be used for determining a'. The optimal policy is the following formula:

Optimal policy:

$$\pi(s) = \underset{a}{\operatorname{argmax}}\ Q(s,a)$$

This is different Q-learning and requires a policy which will allow the mouse to take the optimal action every time .

4. Suppose there is a state-action pair that is never encountered during a trial through state space. What should the Q-value be for this state-action pair? [7 marks]

For state-action pair that have never been encountered during a trial should have the Q-value of 0. As in my implementation, I have included every possible Q(s,a) that is possible but if they are not visited (encountered) they return a value of 0 , as does every Q(s,a) initially in the assignment implementation. They are not encountered and therefore their values are not updated. At Q(s,a) = 0, they also do not account for the subsequent q-value since they are not returned from the max function. Overall, a state-action that is not encountered will have the Q-value 0.

5. For some cases (even with a long trial through state space), the optimal Q-value for a particular state-action pair will not be found. Explain why this might be the case. [7 marks]

If the state space is very large or continuous, it is not easy to compute the value function or Q function for every state-action pair. The number of iterations are too large preventing it to calculate the optimal Qvalue. With longer trials, we approximate Q(s,a) using linear combinations of basis functions.

estimate

$$Q_\theta(s,a) = \theta_1 f_1(s,a) + \theta_2 f_2(s,a) + \cdots + \theta_n f_n(s,a)$$

parametrized by theta (sub theta)

Learning the parameters, can replace the challenge of approximating Q(S) considering we have a good set of basis functions. This can be useful to determine the optimal Q-value as it updates the parameters as it iterates. It also allows us to learn the Q-value for states that are never found/explored which is the case in our question.

6. In the test cases provided, the trials through state space were simulated using a random policy. Describe a different strategy to simulate trials and compare it to using a random policy. [8 marks]

According to the Lecture “Temporal Difference Q-learning in Practice” , there are two approaches, the one we implemented in the assignment where the defined fixed policy is random. In this case, the challenge is that it is slow. The second method is faster and it is Q-learning with Exploration + Exploitation. The trials are performed under the following policy:

$$\pi(s) = \underset{a}{\operatorname{argmax}}\left[f\left(Q(s,a), N(s,a) \right) \right]$$

Where f is the exploration function.

$$f(x,y) = \begin{cases} R^{+} & \text{if } y < N_E \\ x & \text{otherwise} \end{cases}$$

This policy uses the current estimate of Q and updates the Q-value as we iterate through the trials. It is different from the random policy because it will explore spaces that it would not normally explore under a fixed random policy however there is a trade-off between exploration and exploitation. It doesn’t rely only short term rewards but rather looks what may end up giving maximum long term reward.

Reference: Class Notes