CSCI 3482 - Assignment P4

Swaraj Shrestha || A00449808

Implementation details – (i) Assgn_P4.py

The program will run learning episodes for a number of times. It updates the Q-values after each action, prints the table after each episode, but the resulting Q-values at the end of the episodes continue to update.

Using the Program –

(1) Edit Assgn_P4.py on lines 12, 15, 26 to specify the parameters and the number of episodes.

Results –

The car gets to the flag consistently and the time taken to reach this goal decreases after each episode on an average. The program was tested with 100, 250, 500 episodes, states = 5, 10, 20, 100, and alpha = 0.01, 0.1, 0.5.

Questions –

1. Try different state numbers (e.g., 5, 10, 20, and 100). How does the number of states effect the learning process? Which value is better? Why?

   The number of states used to discretize the continuous state space in the Q-learning algorithm affects the learning process. Generally, the more states used, the higher the dimensionality of the Q-table and the longer the learning time. In this specific program, the number of states used is set to 10 by default, which means the state space is discretized into 10 x 10 = 100 discrete states. If we try different state numbers (e.g., 5, 20, 100), we can expect the following:

   - If we use a smaller number of states (i.e., 5), the Q-table will have lower dimensionality and the approximation of the Q-function may not be accurate enough, leading to suboptimal or even unsuccessful learning though I found it generally reached the flag though inconsistently. It sometimes finds the optimal Q-value and reaches the goal, but at times takes a long time to do so and could not reach the goal (I closed the window, and the car did not run forever, this might have changed if let to do so). So, state = 5, oversimplifies the dimensionality and does not let the best action be chosen.
   - Using 10 states generally led to the flag being reached. The car did not get stuck and thus was a good state representation.
   - Using 20 states takes a slightly longer time for the car to reach the flag than with 10 states. I did not see a measurable difference in movement precision though it may have been slightly more consistent.
   - If we use a larger number of states, i.e., 100, the Q-table will have a higher dimensionality, requiring more memory and take longer time to reach the flag. This was because there was a lot more Q-values to fill and the exploration took a lot more time. Here, the approximation is more accurate, and the learning can be more successful.

In this program, when the state was set to 10, the first time, it would usually reach the flag around the 360s mark. But when the state is at 100, the time taken was generally over 5000s.

A good practice would be to start with a moderate number of states and adjust it based on the learning performance and memory constraints. So, if the resources support it, a state space of 100 will be best for learning, but in this case, it reached the goal with 10 states with fairly good consistency, so I believe this is good. Even after a few episodes it was a little slow at times, but overall, the car reached the flag and at better times after each episode.

2. Try different values for α (e.g., 0.01, 0.1, and 0.5). How does α effect the learning process? Which value is better? Why?

   The α represents the learning rate and controls how much the agent updates the Q-values at each iteration (0-1). A high value of alpha will give more weight to the most recent information and potentially make the algorithm converge faster, but it may overshoot the optimal values and fluctuate around them. On the other hand, a low value of alpha will give less weight to the most recent information and converge slower, but it will more likely find a more accurate estimation of the optimal Q-values.

   If we change alpha to 0.01, the learning rate will be lower, and the convergence will be slower, and so the earlier episodes took much longer on average to reach the flag. However, if the program is run for a lot more episodes, the observation was that these episodes became more consistent. So, alpha at 0.01 is best when the number of episodes is set to be high.

   When alpha is 0.1 (default), there is a good balance for both exploration and exploitation. The early episodes are faster in comparison, as there is more exploration, the latter episodes too are consistent. As did with states =10, there was the odd slow episode here and there.

   If we change alpha to 0.5, the learning rate will be higher, and the convergence will be faster but there is a risk of overshooting. I found learning to be high for the earlier episodes, but the decrease in time taken to reach the goal as seen with the other values is not there. This could be due to too much exploration leading to inconsistencies.

   Therefore, the best value of alpha will depend on the specific problem and the learning behavior observed during training. In this program it took the longest for the car to reach the flag when alpha was set to 0.01. It reached there the quickest when alpha was set to 0.5. For learning and consistent results later this means 0.01 is the best, but since it reached the flag at each point even when alpha was 0.1, and it was consistent as well, as had decreasing times with each episode, I would go with 0.1.

Note. The starting position of the car is random, this may hinder the results.