Benjamin Levine
Daniel Su

Comp 560 Reinforcement Learning Writeup

Model-Free Learning

Model Description

For model-free learning, we used a Q-Learning based algorithm which explores and evaluates a utility score for each possible action at every state. This is done by initially randomly selecting possible actions when in each state and determining how far they are from the goal state “In”. A “score” is calculated by keeping track of how many state change iterations were made from the initial state to the goal state. Once the goal state is reached, we average that score with the utility score of the initial action in the initial state. Once utility scores have been assigned to actions, we will select higher scores with a higher probability than lower ones. Additionally, we also keep track with a total_score variable the number of total hits over all our iterations it took to get from the initial state to the goal state. This number gives us an idea of how “effective” our algorithm is with our epsilon and exploit/explore parameters. Finally, we also multiplied a discount

We also have an exploit function which simply always chooses the optimal policy at each state based on the average utility scores assigned to each action. The initial state is randomly chosen.

In our model, we used a epsilon value of 2,000, which indicates the total number of iterations we ran for our model. We also have a explore-exploit ratio between 0 and 1 which determines the ratio between the number of iterations we call explore v exploit.

Observations

1. We started with an initial value of 0.5 for our explore-exploit ratio. We noticed that when we explore more, we often obtain a more optimal policy. However, the total score was

worse with a higher percentage of exploration than exploitation. An optimal score was achieved for a ratio of around 0.2-0.4 for the explore-exploit ratio.

2. We decided to stop learning when for the exploration phase our new score would be consistently within 1 point of our average score. In this case, we defined consistently as for 5 shots in a row. Once this was achieved, we stopped exploring that particular action-state pair. We also had an explicit explore-exploit ratio which determines how much to focus on exploring and how much to focus on exploitation.
3. Adjusting our discount factor changes how much our model values a future expected reward. It controls the focus between immediate and future rewards. When we adjusted our discount value, which we kept at 1.0 for our model, the model tended to explore repeated actions less.
4. For our model, when we used high values for epsilon which resulted in less iterations to learn before stopping but also produced far less optimal solutions, while using small values forced the model to continue learning until it had fully converged.

Model-Based Learning

<u>Model Description</u>

For our model-based learning, we used the Bellman algorithm as described in lecture to update utility values of each state during each iteration. An iteration of learning is performed by going through each possible state, choosing either the currently most optimal action or a random action based on exploration vs exploitation, then updating the transition probabilities based on the observed transition switch, and finally updating the utility value for the state based on the new transition probabilities. A gamma value, exploration value, as well as epsilon values serve as hyperparameters to the learning process.

<u>Observations</u>

1. With the chosen value of epsilon, it seems that changing the exploration amount did not change the optimal policy significantly. This is likely due to the fact that the epsilon set of .002 in utility change as a stopping condition is very low and this means that the algorithm will essentially keep performing training until something close to the global optimal solution is found each time. However, one interesting note about changing the exploration value is that an optimal solution was reached (stopping condition was met) in approximately 100 less learning iterations when using a larger exploration value such as .5-1 compared to a smaller one such as 0-.5.
2. First, we trained the model for an arbitrary amount of iterations and had the learning process print out the total amount of change in utility values for each iteration of learning. After observing approximately how small changes are at the point when the solution seems to be converging. We decided to set an epsilon of .002 total utility change within an iteration as the stopping point as that seemed to serve as a good heuristic for knowing that our solution had converged.
3. Changing the gamma value essentially seemed to slow down the learning rate, which logically makes sense based on how it is used in the equation. More iterations of learning were required to reach a solution when lowering the gamma value generally speaking.
4. Using high values for epsilon naturally took less iterations to learn before stopping but also produced far less optimal solutions, while using small values forced the model to continue learning until it had fully converged.