Assignment 2 Write Up - Kento Yamamoto, Nick Gerold

The model-based learning works by making a class for each location provided. The class stores internal information such as value, name, and which actions have happened and how often. In addition, the class stores multiple dictionaries which hold the possible actions at the location, the possible results of each reaction, and the probabilities of each of the action/destination pairs. The class also generates which outcome occurs and which action occurs when a random one is needed. More exploration in the beginning leads to a wider, more even spread of probability, as the model tries many options before settling on one, and therefore is more likely to the most optimal path. Focusing on exploitation causes the model to find a decent path quickly, but may not be the optimal one. When finding paths to the goal and filling out the model, the epsilon value determines how likely it is for the model to choose a random action versus the one the model finds the best. A higher epsilon leads to a more even distribution of the probability, as the A.I. is more likely to make non ideal moves that it would not make following the model. The program stops learning once 1000 paths to the goal are reached. That number is significant enough that all combinations of action and destination occur, even with a low epsilon. The large data set also provides a decent average, meaning it is closer to the true values. To determine what the value of the location, the values of the adjacent locations are collected. The adjacent values are multiplied by the probability and the gamma value, with a constant 0.04 subtracted to account for the downside of moving. The gamma value going from high to low did not drastically change the probabilities. The total value of the location is the highest of the possible actions.

For the assignment we focused on Reinforcement learning to understand how model-based and model-free works. Model-free is based around being more "loose" and having a willingness to explore more before finding the correct pathway to take and "exploiting" it. One interesting aspect was finding a workaround for the equation used to calculate the utility for each, instead of just calculating it on paper and putting it in. The equation created nets something similar to that of the Bellman's equation. Creating various dictionaries to contain the utility for each specific action per location and having all the data from the assignment2test.txt file put into a dictionary helped run the code very smoothly. At first I had multiple dictionaries for each action, location, etc, but having it all within one nets a much more simpler method. I tested with different values for par, stroke hard cap, and exploration to see what effects it had on the whole run time and answer. I decided that having a large exploration chance at the beginning nets a more holistic view towards utility rather than picking and choosing something that seems to be the best right off the bat. Lastly to calculate the reward I decided that using a very binary approach seemed correct as golf is a game of "is it in the hole or not." It didn't make sense to give reward based on location, but rather give reward based off of the par, stroke count, and if it is in the hole or not. This seemed more realistic, because you are rewarded if you hit a birdie, eagle, or hole in one. This way my calculation takes into account all these aspects to get the reward.

Kento Yamamoto - Main work on model-free, assisted on model-based

Nick Gerold - Main work on model-based, assisted on model-free