Rosh Sharma
Olivia Glass

HW2

We did trial and error to determine when the iterations caused the difference in policies to be negligible. We did this by making the exploration value 1 and increasing iterations until the policy converged. This ended up being around 12,000 iterations for Model Free. Then, we changed the epsilon value to get convergence on the expected utilities, which came to 0.75 with 16,000 iterations. This allowed for our policy and utilities to converge and stay mostly the same after multiple iterations. We also changed the Learning Rate value, varying it from 0.1 to 0.9, and it helped the results stay converged when it was at lower values.

Next, we repeated these steps for Model Based, changing the exploration/exploitation value, the epsilon value, the discount rate, and the number of iterations. We decided to stop learning when the iterations were at 4000, the epsilon value was at 0.8, and the discount rate was 0.9. These values allowed the utilities and policies to converge. Increasing the epsilon allowed for the convergence of the utility, and the policy stayed the same. It is important to keep exploration high so that you have a better idea of what to do when you're in a state. We could keep exploration high and still have a lot of actions exploited because this was a small data set.

For the Model Free implementation, we first generated an action. This was chosen either at random, if exploring, or the best action thus far, if exploiting. Then, we get an ending state based on what happened when we played that action. The utility increased, calculated by adding it to the recursively defined utility of the end state. This simulator then keeps track of what end state it appeared in. To update a given action, the current utility was initialized to zero, we took the difference between the utility that just occurred and the current utility, and multiplied the difference by the learning rate. These utilities were all stored in a queue, ranking them based on which actions provide the best shots (minimize utility). The results are based on the best action for each state and its utility. We didn’t keep track of the actual probabilities.

For the Model Based implementation, we did the same thing in terms of exploration and exploitation and way of getting the end state. But this time we kept track of the start state, action, and where we ended up. Now that we know what all the end states we end up in, we calculated the probability of these after multiple iterations. Then, we repeated these steps until we finished the hole, updating the current utility. Each state has a tracker for the best action and the utility for that action. For each action, we can get the probabilities of ending up in each end state. Then, for each end state, we computed the expected utility using the Bellman Equation, multiplying the probability and the expected utility, summing over all the probabilities of the end states, and if that is the current best action — less than anything we’ve seen so far — then we keep track of that. Then once we find the best action, we make the current utility equal to one plus the discount times the best action’s expected utility.

For contributions, Rosh did the majority of the implementation, while Olivia helped with debugging and writing the report.