Hunter Finger

Scott Hofbauer

COMP 560 Reinforcement Learning Write-Up

For both of the methods used, the data was originally read in from the input.txt file. Each line was split on the "/" characters and read into a linked list. The original state, action, resulting state and probability were all stored in the object.

Model-Based Active Learning:

After the input is checked for having the correct action value pairs, the Main class calls a function model.learnModel(). This is the function that trains the AI on the data from input.txt. 5000 was chosen as our epsilon value iterations were chosen as the amount of time the AI got to learn, we felt that after 5000 iterations the utility values have converged to their limit and there is very little change in their values . The starting state is chosen to be in the Fairway, as that is the first shot taken on the golf course. While the state is not "In" which is the goal state, an action is chosen at random from the list of available actions for the current state. Then a probability is chosen at random between 0 and 1. The result state is chosen based on each state's relative probability. We used an exploration value of 0.3 when choosing the result state. The process is then repeated using the new state as the current state to find the next state until we reach the goal state. The path that was taken gets a count added to each branch of the tree. Every 250 iterations, the probabilities in the state, action, result tuples probabilities are updated. When changing the discount value from a value of .9 to .5, the policy for the shots in

the fairway and ravine changed from shooting at the pin to shooting over the pin because that is the more valuable state short term in the learning process. Changing the exploration value from .2 to .5 had minimal effect on the model-based learning system. Each utility value was calculated using the bellman equation which is the reward plus the discount value times sum of the each action-states probability times the states utility function.

$$U(s) = R(s) + \max_{a \text{ in } A(s)} \sum_{s'} P(s' \mid s, a) U(s')$$

The policies for each state were decided by the minimal utility values that were available to each state. We chose the minimal utility values because they were representative to the action that would produce the least amount of shots for the holes.

Model-Free Active Learning:

Our approach to the model-free portion of the assignment was to implement SARSA learning, a different version of Q-Learning. When the model-free learning function is called in Main, a Q-table is created based on the varying states and actions that the algorithm contains. Initially, the table is initialized to all 0 except the goal state which contains the reward. The Q-table was then updated using the SARSA algorithm.

$$Q(s, a) \leftarrow Q(s, a) + \alpha [r + \gamma \max_{\alpha} Q(s', a') - Q(s, a)]$$

After setting Epsilon to 1000 iterations the model prints the highest values in the q table per state action pair. Changing the exploration value drastically as in model-based resulted in dramatically lower q values for the best state, action pairs. Epsilon being changed did not result in any major differences, but we did see as we made the epsilon larger the Q value for each state ended up converging to 9.9 repeating to show that a very large epsilon didn't have a huge impact. However, lowering the discount value resulted in slightly lower Q-values for each state action pair. The policy for each state was set by the highest q-value available for each state based on the SARSA algorithm.

Contributions:

Scott did the heavy lifting on the Model-Based, but the rest of the work was a collaboration between Hunter and Scott.