Joshua Tabor

Professor Andrew Palay

Comp 560.01

4/23/2020

Comp 560 Assignment 2 Write Up

- How did changing the initial value for exploration (starting close to 1 versus starting closer to 0) change the resulting computed policy?
    - When the exploration value was high, we are exploring heavily at the beginning and later taper off, the utility values for model free tended to be more balanced, that is there weren't as many states/action triples that had high utilities
    - On the other hand when exploration was low there were several states that had high utility values since they were being explored so frequently, while some states had very low values since they didn't have action with high utilities

- How did you decide when to stop learning?
    - For me deciding when to stop learning was a process of trial and error with several different variables
    - If you allow the system to run too long certain variables such as discount values or learning rates will begin to plateau and the learning it no longer useful
    - I tried to let the learning go on long enough that the system would still be accurately adjusting the utilities without having too many states' utilities begin to plateau
    - In the end I decided to let each state update their utilities at least 10 times (by starting from each state 10 times) as most utilities we're beginning to converge after 10 iterations

- How did changing the discount value (starting close to 1 versus starting closer to 0) change the resulting policy?
    - Playing around with the discount value, if I allow the system to run for a constant amount of episodes, in this case epsilon = 5,000 and change only the discount value, I discover that the expected utilities for the triples increased as the discount value approached 1.
    - On the other hand as the discount value approached zero, the Bellman equations quickly converged towards zero and the utility values in return approached 1

- How did changing the value for epsilon change the resulting policy? → Epsilon is the number of episodes run

- Initially I ran 100 episodes in order to obtain my transition probabilities, however this left some states without any information, so I did not bother to even compute standard deviation
- Epsilon = 1,000 episodes → average standard deviation of 3.2% for n = 10
- Epsilon = 5,000 episodes → average standard deviation of 1.2% for n = 10
- Epsilon = 7,500 episodes → average standard deviation of 1.05% for n = 10
- Epsilon = 10,000 episodes → average standard deviation of .7% for n = 10
- For every triplet I calculated the relative frequency across *epsilon* number of episodes and calculated the standard deviation between the 10 different simulations. I then took the standard deviation of all the triplets and averaged those
- I ended up using 7,500 because I felt that it was close enough to 1% avg. st. dev. while still saving some computation time

For this assignment the objective was to determine the optimal policy for shooting at a certain hole on a golf course given certain probabilities for each state/action/state triplet. I begin by reading in each triplet, creating an object in order to store their information along with their probabilities, labeled "weight" throughout the code. From there I move on to implementing the two solutions: model based and model free. For model based I keep a running count of how often a state/action pair comes up (example: "Fairway", "At") and how often a triplet comes up (example: "Fairway", "At", "Hole"). I use the equation # of triplets / total # of state-action pairs in order to determine the transition probabilities. Using these probabilities I used value iteration in order to determine the policy. For the other approach I talked to Juan Garcia about some possible ideas and he directed me towards the Q-learning algorithm. In order to get a more diverse of algorithms out there I used this algorithm for model-free learning. The equation is something along the lines of:

$Q(S,A) = (1 - \alpha) * Q(S,A) + \alpha * sample$ and $sample = \gamma * maxA(Q(S',A))$

Which in short is saying that the expected utility of the current state is the current utility plus the maximum expected utility of future moves. In terms of running the main class there is an input to read in a file name and then select whether to run model based or model free learning.

All work was done by Joshua Tabor, and a huge thanks to Juan Garcia for walking me through a tidal wave of questions.