CS181 Spring 2016 Practical 4: Reinforcement Learning — Team EXT3

Robert J. Johnson — Dinesh Malav — Matthew McKenna

May 1, 2016

Abstract

Swingy Monkey is a very basic game written in Python. Our project focused on using this game as a simplified context to develop and gain exposure to reinforcement learning algorithms. By implementing a variation of Q-Learning, our study showed that after x amount of epochs, our score was able to reach y, a vast improvement over random movements from the monkey.

1 Technical Approach

Initially, we conducted cursory research on different approaches that were available utilizing reinforcement learning. Based on the YouTube video included in the practical specification, we decided our main focus would be an implementation of a particular reinforcement learning algorithm, Q-Learning. We then coded up a solution in Python to develop a policy for the monkey to navigate the course. In basic terms, Q-Learning is an unsupervised learning algorithm that essentially stores value information regarding a particular approach to an outcome-based event in a matrix and uses this matrix to make further decisions in later epochs. We attempted to implement the process described at:

$$http://mnemstudio.org/path-finding-q-learning-tutorial.htm$$

Two major problems arose initially that constrained our implementation. The first major issue was how to interpret the incoming stream of objects in the jungle. In Swingy Monkey, we are faced with what is essentially a continuous input of data, and thus, we decided early on that the best way to for both ourselves and the algorithm to interpret the position space was to discretize the playing field into 'bins. The second issue was the algorithm's ability to infer how the gravity changes during each epoch of the game. Since the pattern is random, we had to quickly find a way to deduce what the rate was in order to determine our policy for navigating the trees.

Our implementation of the algorithm begins by getting the current state of the monkey and the forest. The ultimate goal for the algorithm is to learn which actions have the greatest "reward" and use this information combined with the current input to maximize utility and make informed decisions regarding the next action to take. Our reward information is stored in matrix R and our information on learned rewards is stored in a similar matrix Q.

Two functions ended up doing the bulk of the work for the solution. The first was *action_callback*.

The results of *action_callback* were eventually passed over to *reward_callback*.

2 Results

Our monkey was able to be trained to reach a score of X after Y epochs.

3 Discussion

The implementation discussed here is not perfect, and there are other reinforcement learning approaches that we could have explored. Q-Learning was selected due to its ease of implementation and potential to effectively implement a policy for the monkey's traversal. Research in reinforcement learning is rapidly changing and evolving, so there will likely newer and more novel solutions developed for similar discrete problems such as this.

All code for this project can be found at:

$$https://github.com/HarvardCS181Practical2016$$

.