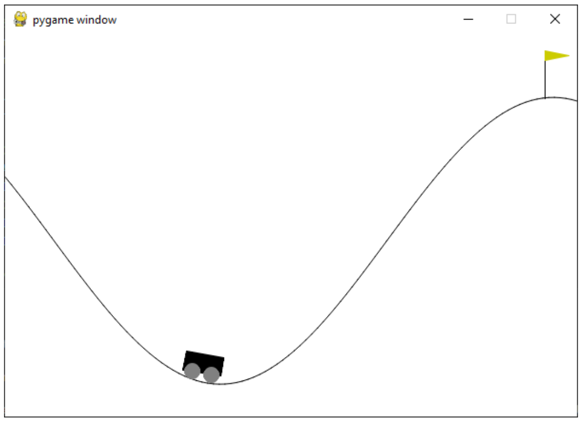
# CSC 449 - Program #3 - Mountain Car

Dillon Dahlke

## Program Description

This program deals with the very popular "Mountain Car" reinforcement learning problem. This problem starts with a car between two "mountains" . The goal of the car is to reach the flag at the end. The only trouble is that the car cannot make it up the hill by going straight up it. It must first gain momentum by moving backwards and then forwards. The environment looks something like this:



## Algorithms

Originally, this program was to be done using Sarsa(Lambda) with gradient descent function approximation using fourier basis functions. This is achieved by constructing action dependent features for different action values. It turns out the radial basis and fourier basis functions both work very well for the mountain car problem. The fourier basis is derived from the fourier series coefficients and is created by plugging the state variables into the corresponding formula to get a feature vector.

The basic value function approximation has two main components. These components are a feature vector and a weight vector. In any state, the value function estimate is approximately equal to the dot product of the weight and feature vectors. The only difference between regular TD value approximation and SARSA value approximation is that SARSA uses action value functions. For each step of the current episode, the weight values are updated according to the gradient of the action value function. This is where gradient descent comes in. This is continued until the next state is terminal. In this case, we move on to the next episode.

## Program Structure and Libraries

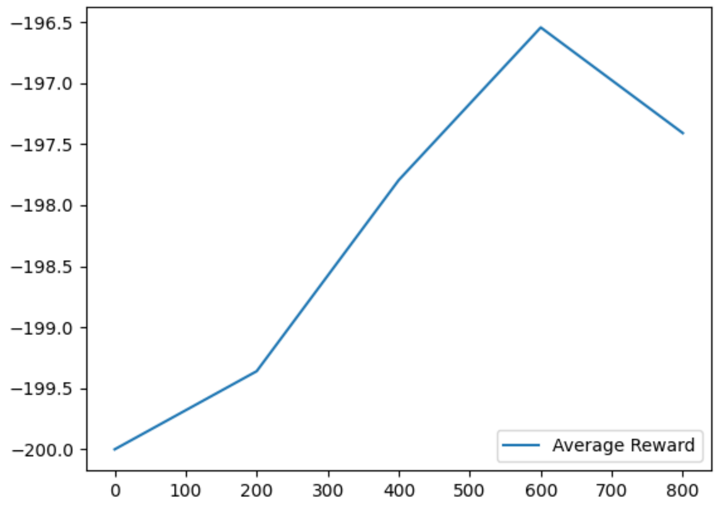
Along the way I ran into lots of problems trying to put the SARSA function approximation algorithm into code. I think I had some misunderstandings with the fourier basis and that didn't allow me to correctly update weights. I also had issues with the SARSA lambda portion. I tried to implement accumulating traces and ran into more problems.

In the end, the only way I was able to get the mountain car to reach its goal was with the classic TD and SARSA methods. The program uses a few Python libraries that are available with PIP. These libraries include:

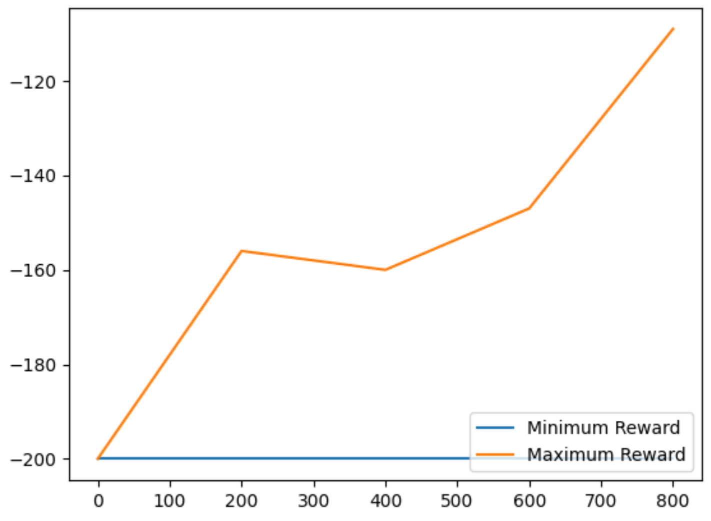
- gym
- numpy
- matplotlib

Gym is used to handle the mountain car environment. It is what could be seen in the introduction of the report, NumPy handles a lot of specific math calculations, and matplotlib handles the graphing of rewards and other values.

I was able to get the car to get its first completed episode at around 50. Included is a graph of the average reward for about 1000 episodes:



Here is also a graph for maximum and minimum reward for each episode:



Adjusting the learning rate, discount factor, and epsilon all have a major impact on how fast the agent learns. Changing one parameter will inevitably change others.

## Conclusion

Overall, this program did not go quite as planned, as I would have liked to get the value function approximator working so I could model my own surface plot of the value function. I didn’t realize until a little too late how seemingly simple algorithms can take a really long time to get working just right in practice. Even though things didn’t go quite how I wanted them to, I still have learned a lot about plotting function parameters and eventually see them start to converge.