Q-Learning

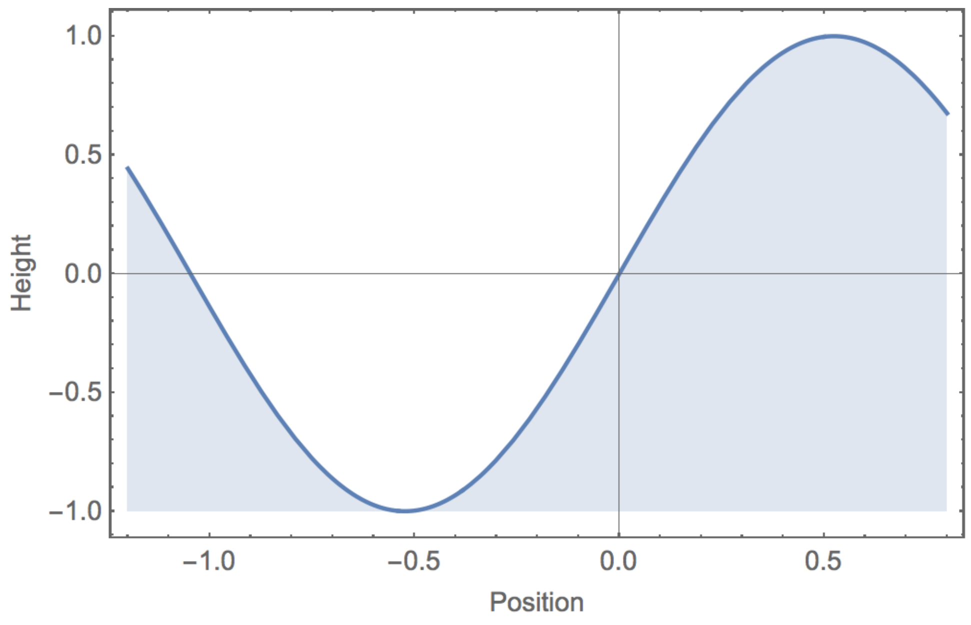
Robert B. Heckendorn
University of Idaho

April 14, 2016

CS504 - Machine Learning
Reward: 200
DUE: Mon Apr 25 at 5:00pm PDT

The 2D Mountain Car Problem

Get car over mountain pass at the right side of the paper when you don't have enough power to just drive over it. Here is the image of hill.



Imagine that you can take one of three actions in terms of acceleration: $a_t = +1$ full forward, $a_t = -1$ full reverse, $a_t = 0$ coast. The car is positioned somewhere between -1.2 and 0.8 on the graph above and must get to 0.8. By accelerating up a hill thereby saving up the energy as potential energy and driving back down the hill you can add the potential energy to your acceleration and use that to send you up over the next hill.

The physics is described by these equations:

$$v_{t+1} = bound(v_t + 0.001a_t - 0.0025\cos(3x_t))$$

$$x_{t+1} = bound(x_t + v_{t+1})$$

bound enforces $-1.2 \leq x_{t+1} \leq 0.8$ and $-0.07 \leq v_{t+1} \leq 0.07$.

Write a C program to solve the car control problem using the Q-learning algorithm on page 242 in the book. Call the program ql. The state space for the reinforcement learning is the 2D space: position and velocity. Assume the space is discretized by steps of .05 in the position direction and .01 in the velocity direction that will make 615 cells to learn in. Let's try the simple reward of +1 if at goal and −1 if not at goal.

Let's see if we can learn to get the car out of the valley from any starting location.

The output for your program is three matrices in the same format we have been delivering matrices: numrows numcols followed by that many rows and columns. The three matrices are the Q matrices for reverse, coast, and forward in that order. Each array is 15×41 with each column representing position and each row a velocity.

I will read in your Q matrices and try to run a mountain car based on them.