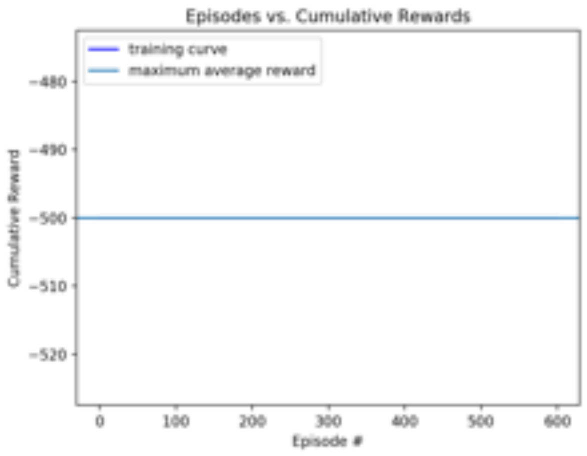
Rahul Maganti
12/5/19

**ESE 680: RL // Take-Home Final**

*Problem 3: Acrobot*

a) Selecting positive & negative actions exclusively is not enough to solve this problem. The training curve is as follows:

The reward when selecting random actions over trajectories of length 500 steps is -500, the minimum reward achieved by agent in this environment. Running for 600 epochs, the average reward remains constant over time.



b)

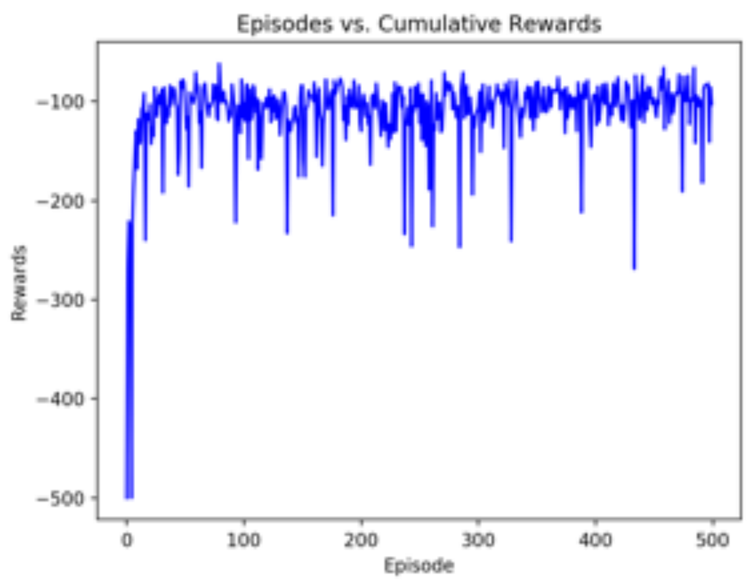
The environment consists of a Continuous State space and a discrete action space A = {-1, 0, 1} . I first considered the usefulness of model-free vs. model-based algorithms. In this case, an algorithm like q-learning would constitute a more difficult approach, requiring me to discretize the state space such that the q-values could be learned for each state.
In the class of model-based algorithms, I started with the simplest version of the policy gradient methods we have explored: Monte Carlo REINFORCE. While this algorithm did eventually solve the problem, it required on average, 5000 iterations for clear convergence to a maximum reward.

To reduce convergence, I decided that an actor-critic (A2C) approach would be more suitable. To reduce variance even further, I also added a baseline to the policy gradient method, with advantage  = Gt - value_function_estimate. With these additions, the agent reaches an optimal reward (> -100) in 500-600 epochs.

Given that the action space is discrete, I chose a policy that sampled from a Categorical distribution. The input to this distribution was approximated by neural network with input_dim = 6, hidden_layer_dim = 50, and output_dim = 3. The input and output dimension values were chosen from the environment parameters (dimension of a given state = 6 & A = {-1, 0, 1}). The size of the hidden was chosen to be sufficiently high as to accurately approximate non-linear functions but not too big as to be computationally intractable.

The learning rate of the algorithm was set to .01. Generally, a large learning rate can lead to faster convergence but yield a set of final weights that are sub-optimal.
Typically learning rates are between .001 and .1. A learning rate of .01 implies that the weights of the network are updated by 1% (weight error) during back-propagation. This strikes a decent balance between fast convergence and optimality.

c) **Training Curve:**


Episodes vs. Cumulative Rewards
Rewards
Episode