CS660: Sequential Decision Making – HW3: Policy Gradients

Michael Probst
mppr222@uky.edu

Abstract

This assignment implements a reinforcement learning agent that solves the Cartpole and Lunar Lander open AI gym environments. The agent implements a REINFORCE algorithm to solve both problems by converging to an average score of 200 in 190 training episodes for Cartpole, and an average score of 93 in 600 training episodes for Lunar Lander.

Background

REINFORCE

The REINFORCE reinforcement learning algorithm is unique in that the parameters of the neural net are not updated until the end of the episode. Rewards after each action are recorded, and used at the end of the episode to calculate the return with the following equation:

$$G_t = \sum_{t'=t}^{T} \gamma^{t'-T} r_{t'}$$

This equation describes the return as the sum of the discounted reward at each time step from the current time step to the end of the episode where G_t is return, T is the time at the end of the episode, t is the current time step, γ is the discount factor, and $r_{t'}$ is the reward at a time t'.

Return is used in the loss function:

$$J(\theta) = \log\, \pi_\theta(a_t|s_t)G_t$$

Where $log\pi_\theta(a_t \mid s_t)$ is the output of the neural network.

Implementation Decisions

In order to implement REINFORCE, the actions, rewards, and returns of each time step needed to be stored in data structures so that they could be iterated over so that the return each action could be calculated. Also, REINFORCE is prone to high variance. In order to mitigate this, batching and a baseline were used, also utilizing the array of rewards.

Neural Network

REINFORCE requires the implementation of a neural network. The neural net used in this project is a feed-forward network with 1 hidden layer with 32 inputs and 16 outputs. The input layer takes the observed state from the environment which is of size 1 for Cartpole and 4 for Lunar Lander. The number of outputs on the output layer is the number of actions for each environment being 2 for Cartpole and 3 for Lunar Lander. The activator functions on the input and hidden layer are both ReLU activators. The neural network created for this project utilized the pytorch library.

Batching

Some episodes yield very large or very small losses that can have dramatic effects on the behavior of the agent. This is mitigated by batching several episodes and taking the average loss of the episodes is used to update the neural network. Batching transforms the arrays of actions, rewards, and returns into matricies. A batch size of 5 was used in this implementation.

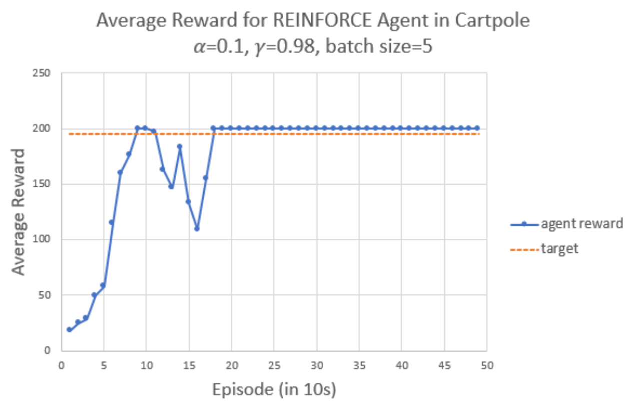
Baseline

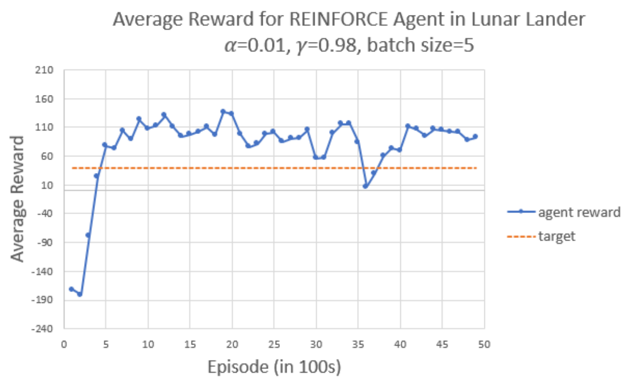
A baseline is used to lessen the effect of return while calculating loss such that the loss equation becomes:

$$J(\theta) = \log \pi_\theta(a_t|s_t)(G_t - b_t)$$

The baseline implemented in this solution was the average return over the episodes in the current batch. That is, the return of each action over all episodes in the batch are averaged. For example, the returns at timestep 1 ,2, …, n are averaged over 5 episodes before the loss is calculated and the network is updated. Since each episode will vary in length, each array of returns for each episode is padded with 0s to match the length of the longest episode in the batch.

Results





The graphs above show the rewards for the REINFORCE agent solving the Cartpole and Lunar Lander environments. After 10 and 100 training episodes for Cartpole and Lunar Lander, respectively, 10 tests were run using the agent's current policy. The average score over those 10 tests are the points recorded on the graphs. The target line is the reward that the agent must converge to in order to consider the environment "solved" for the purposes of this assignment. The target reward for Cartpole is 195 and 40 for Lunar Lander. The agent converged to 200 score in Cartpole after about 190 training episodes, and the agent converged to 93 score in Lunar Lander after 600 training episodes. Both of these scores exceed the target reward.