Solving Tennis environment using Proximal Policy Optimization

Author: Jakub Bielan

Learning Algorithm

Proximal Policy Optimization alternates between sampling data through interaction with the environment, and optimizing a "surrogate" objective function using stochastic gradient ascent. Whereas standard policy gradient methods perform one gradient update per data sample, PPO enables multiple epochs of minibatch updates. It has some of the benefits of trust region policy optimization (TRPO), but they are much simpler to implement, more general, and have better sample complexity (empirically). Experiments test PPO on a collection of benchmark tasks, including simulated robotic locomotion and Atari game playing, and they show that PPO outperforms other online policy gradient methods, and overall strikes a favorable balance between sample complexity, simplicity, and wall-time.

Model Architecture

This implementation of Proximal Policy Optimization is using a Critic Network as a baseline to Actor Network.

The particularity of the Gaussian Actor Network is its capacity to create a normal distribution based on a given mean of distribution mu and a standard deviation sigma. That is then sampled to generate the values of the actions. The standard deviation is a trainable parameter of the network set at 1 and the mean of distribution is generated by a Fully Connected Network based on the input states. During training, the Critic Network's input is also augmented by providing it with the state observed by the agent, the action taken by the agent, the state observed by the other agent, and the action taken by that later.

Hyperparameters

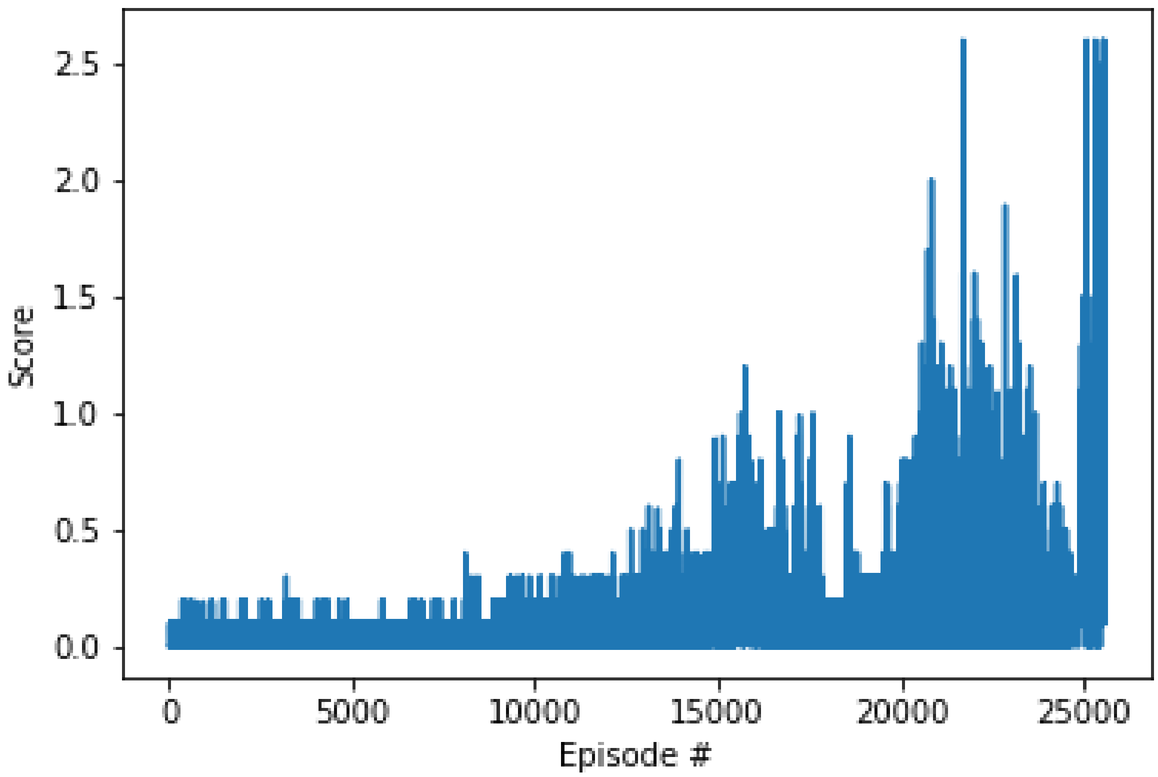
Learning Rate - 0.0003
Noise Insertion frequency - 1 timestep
Noise Decay - 0.9996
Minimum Noise - 0.01
BATCH Size - 256
Gamma - 0.99
Lambda - 0.99
TAU - 0.001
Gradient Clipping - 5
Clipping Size - 0.1

Actor Network:
FC(24, 32) LeakyReLU, FC(32, 32) LearkyReLU, FC(32, 2) Tanh

Critic Network:
FC(52, 32) ReLU, FC(32, 32) ReLU, FC(32, 1) ReLU

Training Result

It took 25563 episodes and about an hour of training to achieve 0.5 average reward over 100 episodes. I trained this model only on CPU because in this case GPU didn't give much better performance.



Future improvements

The most interesting tune in my opinion would be to train agents to compete with each other by giving them the biggest rewards for points and only smaller for hitting the ball.