Report

December 24, 2020

1 Project 1: Navigation

This project solves the single agent Banana Collector Unity environment.

1.1 Environment details

The Banana Collector environment involves collecting bananas in a large, square world.

Each yellow banana collected gives +1 reward, and each blue banana gives -1.

There are 37 continuous dimensions in the state space including the agent's velocity, direction, and ray-based object perception.

The action space consists of four discrete values: move forward, move backward, turn left, and turn right.

The environment is considered solved when the agent gets a score of +13 over 100 consecutive episodes.

1.2 Implementation

The code borrows heavily from the code in the Deep Q-Networks lesson.

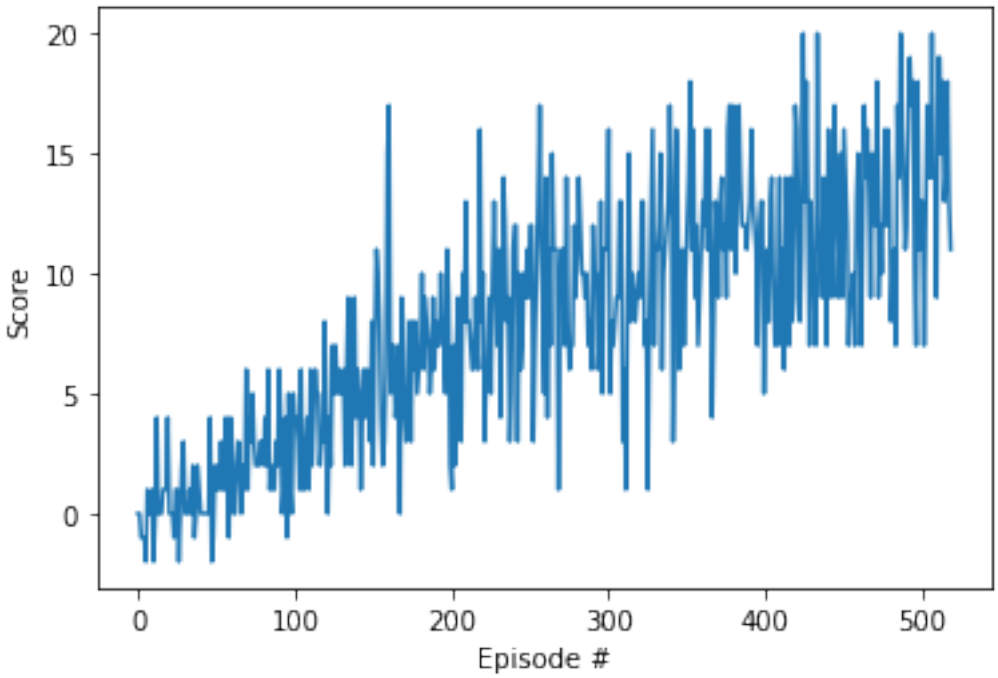
The code trains a DQN agent to solve the banana collection environment. The agent's model is a simple deep neural network consisting of two fully connected layers with 64 units each and relu activation. The agent learns by drawing batches of experiences (S, A, R, S' tuples) of size 64 from its replay buffer. This feature prevents learning from being skewed by sequential correlations between experiences, which limits oscillation and divergence.

The agent also learns using fixed Q-targets. To implement this, a separate target network is maintained to compute TD targets. This networks parameters are updated very slowly (only .001 the amount of the main network) and this also minimizes training oscillations and divergence.

1.2.1 Summary of hyperparameters

- (discount rate): .99
- (learning rate): 5e-4
- (soft update rate for target network): .001
- Batch size: 64
- Network updates every 4 steps

1.3 Plot of rewards



The environment was solved in 519 episodes with an average score of 13.00.

1.4 Ideas for future work

There are many different ways the current implementation can be improved: - Prioritized experience replay can be used to assign a priority to each experience in the replay buffer based on TD error. This will allow more important experiences to be utilized more frequently. - Dueling networks can be used to learn the state values in one network and the advantage values in another, with the networks able to share some layers. This has been shown to lead to better policy evaluation when many actions have similar values, such as in this environment. - A combination of these methods can be used, also known as a rainbow method. Other extensions can also be incorporated, such as multi-step bootstrap targets, distributional DQN, and noisy DQN.