DRLND – Navigation project

Introduction

The objective of this project is to train an artificial intelligent agent to navigate and collect object in a large, square world. We are going to use the banana environment provided to collect banana. The agent will be trained using reinforcement learning algorithm to learn about the environment and collect banana based on the reward.

A reward of +1 is provided for collecting a yellow banana, and a reward of -1 is provided for collecting a blue banana. Thus, the goal of the agent is to collect as many yellow bananas as possible while avoiding blue bananas.

The state space has 37 dimensions and contains the agent's velocity, along with ray-based perception of objects around the agent's forward direction. Given this information, the agent must learn how to best select actions. Four discrete actions are available, corresponding to:

- 0 - move forward.
- 1 - move backward.
- 2 - turn left.
- 3 - turn right.

The task is episodic, and to solve the environment, agent must get an average score of +13 over 100 consecutive episodes.

Approach

In this project, value based deep reinforcement learning, specifically the Deep Q-Network(DQN) will be used to solve the environment. DQN combines Q-learning with deep neural network to achieve learning. Using a deep neural network with Q learning is known to cause convergence and stability issues. To handle this, DQN could be tweaked with an experience replay buffer and additional Fixed Q-Targets.

The experience replay buffer could be used to hold a finite number of past experiences. As the exploration happens, these are added to experience replay buffer. The learning agent could then sample from this replay buffer. This approach removes temporal correlation and would result in better convergence. With the Q targets, there would be another network with identical architecture, such that the learning phase would be logically separate from when interacting with environment for experience. The target Q-network's weights are updated less often than primary Q-network. DQN involves off policy learning, so that network which is learning is separate from the second network which provides the action-value estimate. At a certain interval the learned Q-targets updates the first network. This approach adds to stability of the algorithm to learn.

Solution

To solve banana environment, a basic DQN agent is implemented. The neural network model is used to map the States as input to Action as its output. Also, an experience replay buffer is added, so that it holds the fixed number of past experiences.

Network Model

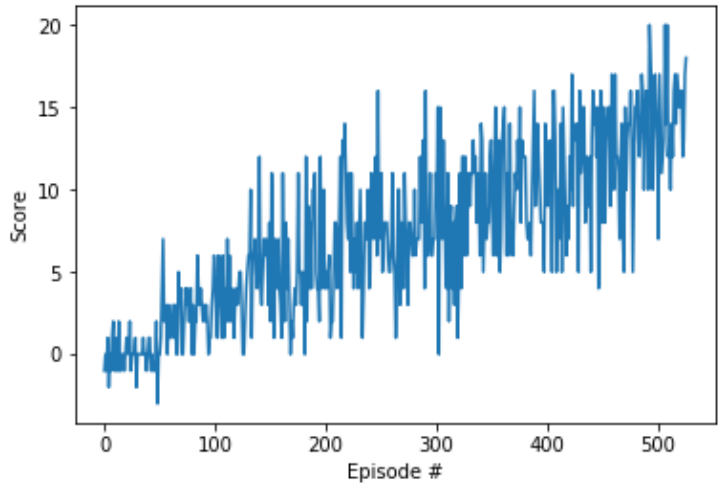
The network consists of two hidden layers of 32 nodes both of which use relu activation after it. Worked with different values for the hidden layer such as 16 and 32. 32 seem to work better.

Hyperparameters

The other hyper parameters that I experimented with includes epsilon value, epsilon decay, buffer size, batch size, discount factor, learning rate, how often to learn.

```
eps_start=1.0           # start value for epsilon
eps_end=0.01            # end value for epsilon
eps_decay=0.995         # Decay factor for epsilon
BUFFER_SIZE = int(1e5)  # replay buffer size
BATCH_SIZE = 64         # minibatch size
GAMMA = 0.99            # discount factor
TAU = 1e-3              # for soft update of target parameters
LR = 5e-4               # learning rate
UPDATE_EVERY = 4        # how often to update the network
```

The solution with a goal score of 13.0 was achieved at 526 episode. The graph below clearly suggests the learning improving over time/number of episodes.



Future Enhancements

The basic DQN network has its shortcomings such as Over-optimistic estimations, experience replay information loss. There are enhancements that came out for DQN such as Double DQN, Dueling DQN, Prioritized Experience replay and Rainbow network each one of which solves specific issue.

I would also work on further hyperparameter tuning, layers in neural network, loss function used, learning rate, discount rate etc…