José Andrés Millán Romera

Udacity DRLND Project 1: Navigation

Introduction

For this project, I have trained an agent to navigate (and collect bananas!) in a large, square world.

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 your 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 agent's forward direction. Given this information, the agent has to 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 in order to solve the environment, your agent must get an average score of +13 over 100 consecutive episodes.

Algorithm

The algorithm employed for this project consists in the basic DQN upgraded with:

- Double DQN
- Prioritized Experience Replay
- Dueling DQN

Double DQN

Double DQNwas proposed for solving the problem of large overestimations of action value. It utilises Double Q-learning to reduce overestimation by decomposing the max operation in the target into action selection and action evaluation. We evaluate the greedy policy according to the online network, but we use the target network to estimate its value.

Prioritized Experience Replay

PER makes use of the idea that when we sample experiences to feed the Neural Network, we assume that some experiences are more valuable than others. If we sample with weights, we can make it so that some experiences which are more beneficial get sampled more times on average. The benefit of each sample is valued based on the absolute value of the TD Error, which is later smoothed and normalized to calculate the probabilities.

Dueling DQN

Dueling DQN splits the Q-values in two different parts, the value function V(s) and the advantage function A(s, a). The former contains how much reward will be collected from state s. And second stores how much better one action is compared to the other actions. Combining (sum) the value V and the advantage A for each action, we get the final Q-values.

Future improvements

Two different types of improvements could be implemented.

The first one is realted to the environent and model. The environmnent can be updated to get as input the image that the agent perceives. Hence, our model should incorporate a convolutional network before the desnde layers.

Second, three more extensions could be implemented:

- Multi-step bootstrap targets
- Distributional DQN
- Noisy DQN

Model

Architecture

1. Fully connected layer - input: (state size) output: 64

2. Fully connected layer - input: 64 output64

3a. Fully connected layer - input: 64 output: 1

3s. Fully connected layer - input: 64 output: (action size)

Parameters

DQN:

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

PER:

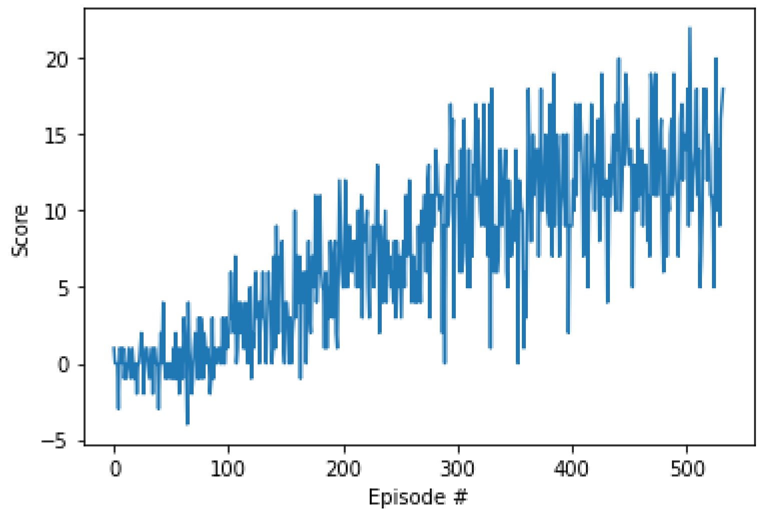
e = 5e-2

beta = 0.6

alpha = 0.4

Results

Episode 100Average Score: 0.24Last Score 1.00Episode 200Average Score: 4.00Last Score 6.00Episode 300Average Score: 7.88Last Score 11.0Episode 400Average Score: 10.56Last Score 9.00Episode 500Average Score: 12.65Last Score 15.0Episode 533Average Score: 13.08Last Score 18.0**Environment solved in 533 episodes!Average Score: 13.08**


Score
Episode #