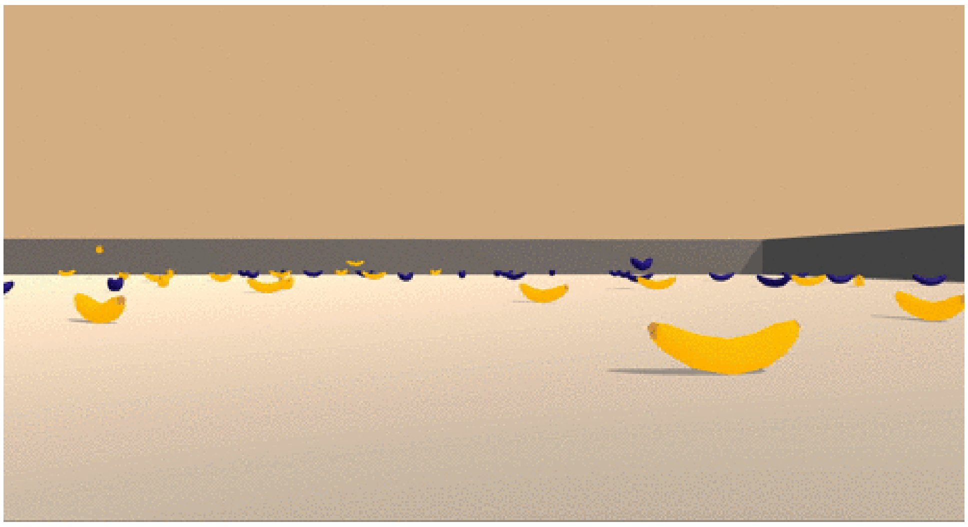
Project 1 - Navigation in the Banana Collection World



By José Ilberto Fonceca Junior

Abstract

This document summarizes the procedures to train a deep reinforcement learning agent with Deep Q-Networks to solve the Banana collector environment brought by Udacity.

Introduction

This project aims to train a banana collector agent capable of navigating in the Banana collector environment and getting as many rewards as possible during its navigation. In this environment, the agent can achieve 2 types of rewards: yellow banana (+1) and blue banana (-1). The states of the agent are specified by a ray tracing based approach that measures the income light produced by the objects at sight (whatever object that can be seen in the viewing cone of the agent inside the environment at each time step).

Environment

States

We are interested in training an agent able to navigate in this banana collection world where very little information is given to the agent. In this world, the agent can only see in a viewing cone in front of him and a state vector with 37 dimensions is created using ray-based perception of objects. According to this post, the vector state is created in the following manner:

- Values 1-36: Ray values. Representing 6 vectors for different portions of the viewing cone. These vectors have 6 dimensions that indicate the ray segments "intensity" from the agent toward the possible positions of bananas from positions 1 to 5 (being 0 or 1), the last one indicates the solid angle of the current observation region.
- Value 37: The linear velocity of the agent.

We can notice that the state space is continuous leading us to use a Deep Q-Network framework suggested in the Human-level control through deep reinforcement learning.

Actions

So in this world, the agent can take 4 actions:

- 0: move forward
- 1: move backward
- 2: move left
- 3: move right

Installation and requirements

Plese, visit the Deep Reinforcement Learning repository maintained by Udacity in order to install all dependencies to work with the code used here. All the steps presented there can be break down into the steps in file requirements.txt:

```
youruse@yourcomputer:~$ conda env create -f environment.yml
youruse@yourcomputer:~$ python -m ipykernel install --user --name drlnd --display-
name "drlnd"
```

Finally we should download and extract the banana environment in the same folder of your project. Please, refer to the list below to download it:

- Linux: link
- Mac OSX: link
- Windows (32-bit): link
- Windows (64-bit): link

Algorithm

The agent learns how to navigate the environment using a Deep Q-learning algorith with epsilon greedy policy. The agent has 2 densely connected neural networks:

- the local: it improves at every accumulated time step (k, since we are not taking actions at every frame we observe in the environment, but we are waiting k frames to do something). By calculating the action-value function of a batch the of size (b) using both the local and the target networks, we obtain 2 different action-value functions that are then used to update the local;

- the target: comparing it with the local one every N batches of size k, we update it using a separate weight (tau) called the soft update rate of the neural network.

In order to add past experiences into every batch, we also use a Replay buffer that stores previous tuples (states, actions, rewards, next_states) in order to add a memory into the system and avoid some greedy behavior towards the last course of actions adopted by the agent.

This combination of neural networks and a replay buffer allow us to learn at small steps and from the past experiences how to navigate this environment and obtain better rewards.

- The specifications of the neural netwroks

The table below presents all the parameters used in the Neural networks.

Layer	Dimensions	Type	Activation	Information
Input	37			tensor with dimensions (batch_size, 37)
Hidden	64	Linear	ReLU	
Hidden	32	Linear	ReLU	
Output	4	Linear		tensor with dimensions (batch, 4)

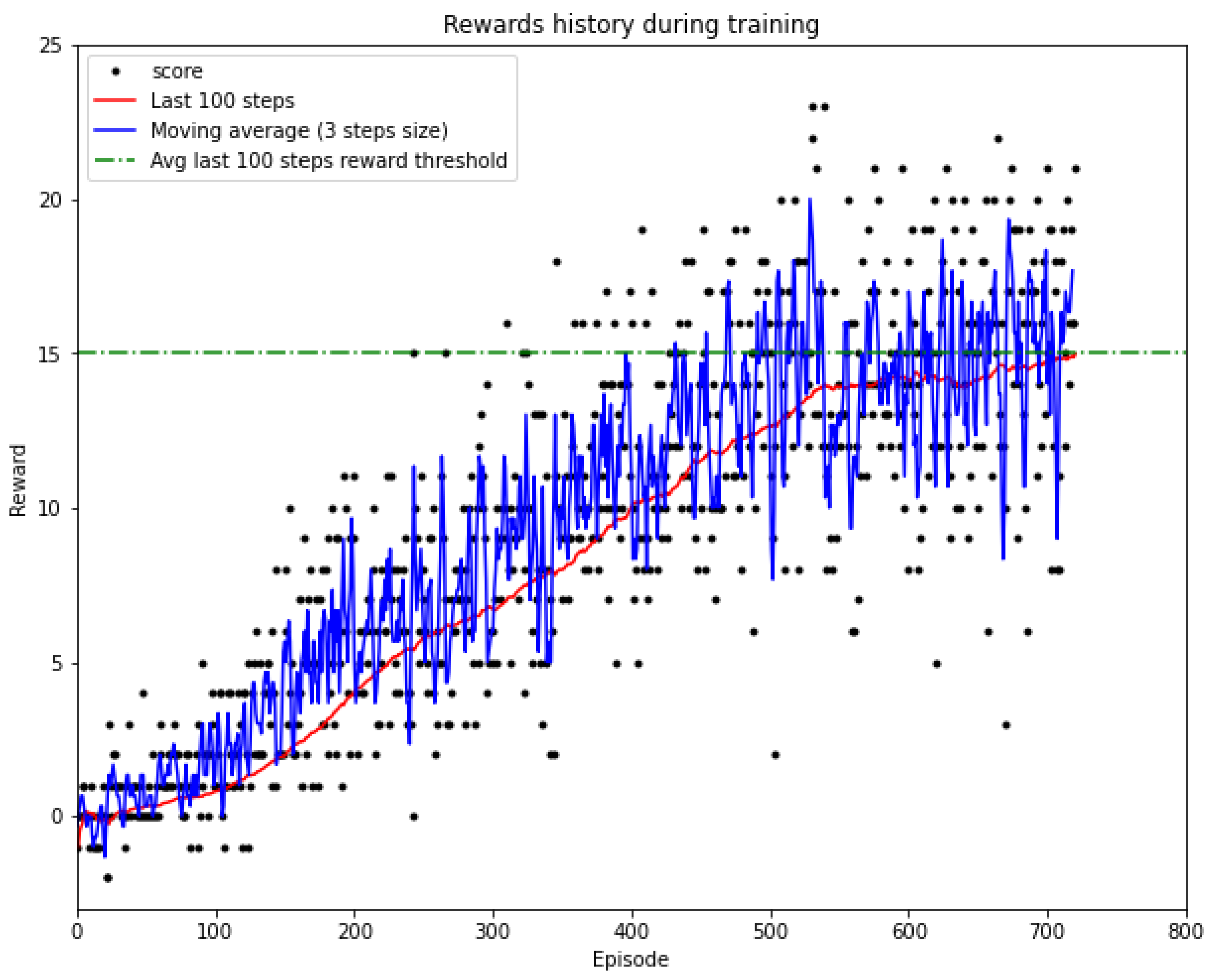
- The structure of the agent

The hyper parameters for the agent are given below.

Parameter Name	Value	Description
Replay initial size	0	Replay buffer initial size
Replay size	100000	Replay buffer maximum size
Batch size (b)	64	Batch size for each update
Gamma	0.99	Discount rate
Tau	0.001	Soft update rate
Batches until update (N)	4	Number of batches until the target network updates
Learning rate	0.00005	The learning rate of the local neural network
Maximum time steps	3000	Maximum time steps taken during the training phase
Reward threshold	15	The number of rewards obtained during the last 100 episodes for truncating the training

Results

Using this set of parameters, our agent was able to navigate through the environment collecting the most rewards it could, needing less than 800 episodes to obtain an average reward of 15 over the last 100 episodes. These results are shown in the figure below.



Conclusions and future work

The agent was able to generalize the environment enough to obtain the most rewards in the given maximum time steps. Nevertheless, we do not know how it would perform in a scenario where there is no limitation in the training of the agent and if it would be ever able to navigate indefinitely in the scenario. Specially, we do not know how much it struggles to work around extreme scenarios, such as when it is mostly surrounded by blue bananas.

The natural prospects of the preset work is to check those situations and make sure the agent is able to generalize well enough the state space in order to navigate it fluidly and collect the most rewards even in the most extreme scenarios.