Report

May 7, 2023

1) Summary

The aim of this project was to train an agent to navigate and collect bananas in a large, square environment. A reward of +1 was given for collecting the current banana(yellow) and -1 for collecting the wrong banana(blue). While the goal of my agent was to collect more yellow bananas while avoiding the blue bananas. The state space had 37 dimensions and contains agent's velocity along with ray-based perception of objects around agent's forwarded direction. Given the information, the agent's task was to select an action from the following 4 actions:

- 0: Move forward
- 1: Move backward
- 2: Turn Left
- 3:Turn Right

This task was episodic, and in order to solve the environment, my agent must get an average score of +13 over 100 consecutive episodes. I used the unity agents API to test and evaluate the agent.

2) Model

I used a Deep Neural Network as a function approximator for the Q-function. This is called Deep Q-learning. The architecture I chose for the Deep Neural Network was of 3 linear layers. The first layer had input dimension 37 and output dimension of 64. The second layer had input dimension 64 and output dimension of 64. And the last layer had input dimension of 64 and output dimension of 4, where 4 specifies the number of actions. The activation function for each layer is a RELU function. The model was trained using Gradient Descent with the Adam optimizer to update the weights.

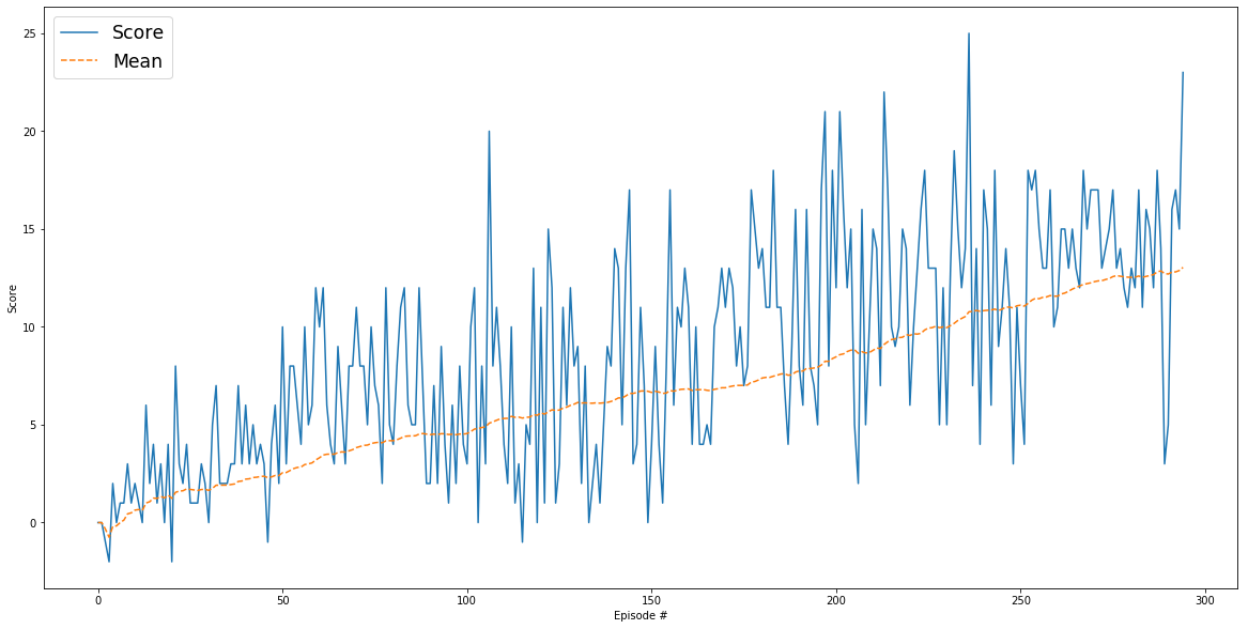
```
class QNetwork(nn.Module):
    def __init__(self, state_size, action_size, seed, fc1_units=64, fc2_units=64):
        super(QNetwork, self).__init__()
        self.seed = torch.manual_seed(seed)
        self.fc1 = nn.Linear(state_size, fc1_units)
        self.fc2 = nn.Linear(fc1_units, fc2_units)
        self.fc3 = nn.Linear(fc2_units, action_size)

    def forward(self, state):
        x = F.relu(self.fc1(state))
        x = F.relu(self.fc2(x))
        return self.fc3(x)
```

Hyper-parameters

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
Note: learning rate is also configurable, you can specify when creating an agent.

3) Plot of Rewards

As stated before the goal for the agent was to receive an average reward of at least +13 over 100 episodes. In the graph below, you can see the agent was able to achieve its task within 295 episodes and an Average Score of 13.04.



4) Future work to consider

I plan on improving the agent with more several different DQN methods in the near future:

- Duelling DQN
- Double DQN
- Prioritized Experience Replay