Navigation project

Project Description

This project consists of navigating an agent through an environment composed of yellow and blue bananas. The agent gets a reward of +1 when collecting a yellow banana and a reward of -1 when collecting a blue banana.

The state of the agent is continuous composed of 37 dimensions. Whereas the action space is discrete composed of 4 actions.

Algorithm used

In order to solve this problem, we used a Deep Q-network composed of two neural networks. The first one will be updated at each action, we symbolize its weights by w. The second one will work as a target which we will update its weights less frequently, we symbolize its weights by w_-. The Q-learning formula is the following:

$$\Delta w = \alpha \cdot \left(R + \gamma \max_a \hat{q}(S', a, w^-) - \hat{q}(S, A, w)\right) \nabla_w \hat{q}(S, A, w)$$

Where w_- are the weights of the target network, and (S, A, R, S') is an experience tuple.

For training we use experience replay technique using a buffer keeping the last 100,000 experience tuples (S, A, R, S').

The target network weight are updated at each learning step using the following formula:

$$w_- = \tau \cdot w + (1 - \tau) \cdot w_-$$

To comply with the fixed target logic, τ is very small (the value 10^{-3} is used in this project).

Since, future reward should count less then present rewards, we use a discount factor γ of 0.99.

Last but not least, to find a balance between exploitation and exploration we choose the probability to choose randomly an action ε to start at 1, with a decay of 0.995, and a minimum value of 0.01.

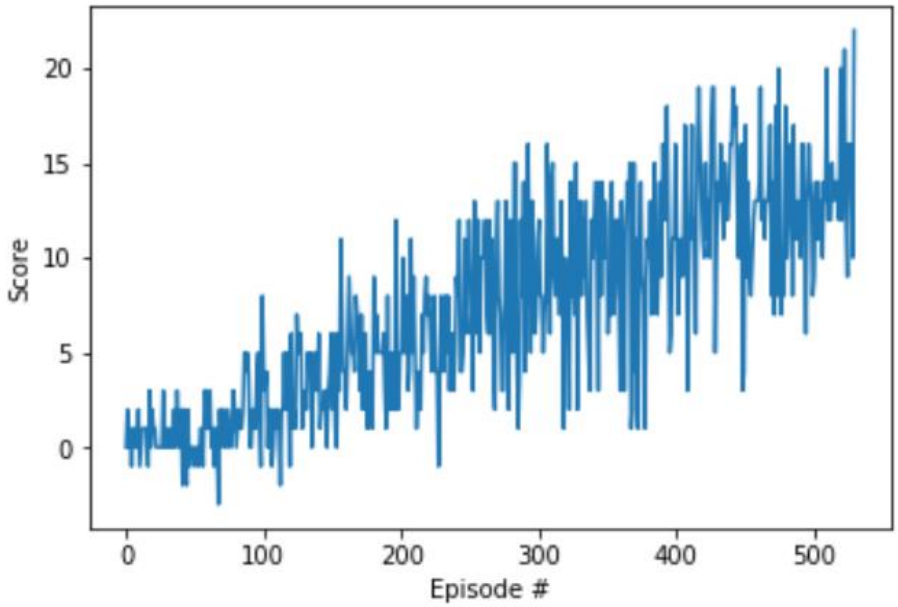
Neural network architecture

Instead of using a function q, we use a neural network. It is composed of two dense layers containing 64 units with a relu activation, and an output dense layer containing 4 units, the dimension of the action space. It takes as an input the state and output a vector containing a probability for each action.

The optimizer used is Adam with a learning rate of $5*10^{-4}$. For the training a batch size of 128 experiences is used.

The networks are not updated at each action, but at each 4 actions.

Results



The criteria to stop training is to have a score of 13 over 100 episodes. It took 530 episodes to achieve that.

Ideas for future work

Experience replay could be explored further by giving an extra weight to experiences having a higher error.

Use Double DQN to balance out Deep Q-leaning action values overestimation. By getting the argmax action value from the target network, but evaluating the target action value using the local network.

We can also explore further varying the algorithm and the networks hyperparameters.