Project 1. Navigation

Introduction

In this project, a DQN agent is trained to navigate and collect bananas in a large, square world.

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 needs 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. 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 environment is considered solved if the agent gets an average score of +13 over 100 consecutive episodes.

Algorithm

Deep reinforcement learning uses nonlinear function approximators to calculate the value actions based directly on observation from the environment, which is represented using a deep neural network.

To address the instabilities from reinforcement learning, the following two key features are used in the Deep Q-Learning algorithm:

Experience Replay – a replay buffer is used to contain a collection of experience tuples. By sampling a small batch of tuples from the replay buffer, it can break harmful correlations, learn more from individual tuples multiple times, and recall rare occurrences.

Fixed Q-Targets- to avoid harmful correlations by updating guess based on another guess, the parameters of the network, w, are updated with the following update rule:

$$\Delta w = \alpha \cdot \overbrace{(\underbrace{R + \gamma \max_a \hat{q}(S', a, w^-)}_{\text{TD target}} - \underbrace{\hat{q}(S, A, w)}_{\text{old value}})}^{\text{TD error}} \nabla_w \hat{q}(S, A, w)$$

where w^- are parameters of a target network that are not changed during the learning step.

The selected Q-Network contains 4 fully-connected layers. The first three fully-connected layers have ReLU activation function. Also note that the input size of the network is the dimension of the state space, where the output size of the network is the dimension of action space. See Figure 1 below for the network architecture.

Figure 1. Q-Network Model

The detailed algorithm is described below:

Initialize replay memory **D** with capacity **BUFFER_SIZE**
Initialize action-value function **q** with random weights **w**
Initialized target action-value with weights $w^- \leftarrow w$
For episode **i** = 1 to **n_episodes**+1:
    Prepare initial state **S**
    For time step **t** = 1 to **max_t**:
        Choose action **A** from state **S** by policy $\pi \leftarrow \varepsilon - greedy(q(S, A, w))$
        Take action **A**, observe reward **R** and prepare next state S'
        Store experience tuple **(S, A, R, S')** in replay memory **D**
        Set $S \leftarrow S'$
        Obtain random minibatch of experience tuple (s_j, a_j, r_j, s_{j+1}) with size of **BATCH_SIZE** from replay memory **D**
        Set target $y_j = r_i + GAMMA * max_a(q(s_j, a, w^-))$
        Update the weights with $\Delta w = LR * (y_j - q(s_j, a_j, w))\nabla_w q(s_j, a_j, w)$
        After every **UPDATE_EVERY** step, soft update
                $w^- \leftarrow TAU * w + (1 - TAU) * w^-$

        If **DONE**
            Break
        Update the value of epsilon

Experimental Evaluation

This environment is solved with 429 number of episodes to reach an average score of +13.04 with 100 consecutive episodes. The plot of rewards per episode is displayed in Figure 2 below.

There are several hyperparameters need to be selected. Table 1 shows the hyperparameter values selected for the trained agent.

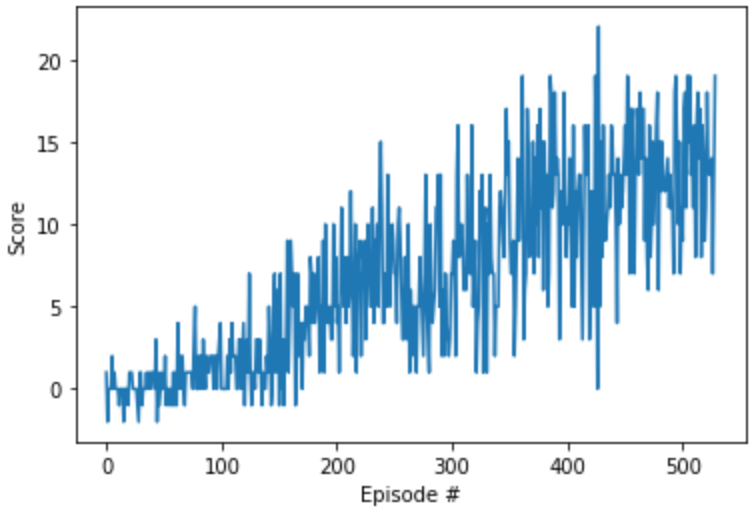

Figure 2. Episode # vs Score

Hyperparameter	Description	Value
BUFFER_SIZE	Replay buffer size	1e5
BATCH_SIZE	Minibatch size	64
GAMMA	Discount factor	0.99
TAU	The weight of local network parameters for soft update of target parameters	1e-3
LR	Learning rate	5e-4
UPDATE_EVERY	How often to update the network	4
n_episodes	Maximum number of training episodes	2000
max_t	Maximum number of timesteps per episode	1000
eps_start	Starting value of epsilon, for epsilon-greedy action selection	1.0
eps_end	minimum value of epsilon	0.01
eps_decay	multiplicative factor (per episode) for decreasing epsilon	0.995

Table 1. Selected Hyperparameter Values

Future Work

To further improve the performance of the Deep Q-Learning algorithm described above, the following improvements should be considered to avoid overestimation of Q-values and prioritize experience that are important but infrequent:

- Double DQN
- Dueling DQN
- Prioritized Experience Replay

Reference

[1]. Mnih, V. et al. Human-level control through deep reinforcement learning. Nature 518, 529–533 (2015)