Deep Q Learning to solve Unity ML Banana Collector environment

This project is one of the three projects in the Udacity Deep Reinforcement Learning Nanodegree. The code was expanded and adapted based on a DQN tutorial developed by the Udacity DRL Team, in which the original code was used to solve Open AI's Lunar Lander environment.

The Environment

In this project, a Deep Q Learning agent was trained to navigate and collect bananas in a 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 the agent is to collect as many yellow bananas as possible while avoiding blue bananas.

State space

The state space has 37 dimensions and contains the agent's velocity, along with ray-based perception of objects around the agent's forward direction. Given this information, the agent has to learn how to best select actions.

Action space

Four discrete actions are available, corresponding to:

0 - move forward.

1 - move backward.

2 - turn left.

3 - turn right.

Deep Q Learning Algorithm

Deep Q Network

A Multilayer Perceptron (MLP) is used to predict the actions values for the four discrete actions. There are three hidden layers in the network, with 64, 128, 64 hidden units in each layer respectively.

Replay Buffer

A replay buffer was used to store experience for experience replay, which has the advantage of learning from rare states and leads to better convergence.

Fixed Q Targets

During training, temporal difference learning was used to compute the target values that were used to train the DQN. The target values were computed using a separate neural network, which fixes the target for better convergence. The target network was updated by a value of tau once every 64 training steps.

Epsilon-greedy policy

Epsilon-greedy policy was used to choose actions to allow sufficient exploration for the agent.

Loss

Mean squared error is used to compute the loss between the target values and the action value estimated by the local network.

Results

The task is episodic and the environment is considered solved when the agent achieves an average score of +13 over 100 consecutive episodes. The agent was able to solve the environment in 600 training episodes in which it achieved an average score of 13.57.

Improvements

Double Deep Q Network (DDQN) can be used to improve the performance of the agent, in which two separate networks are used to compute the Q-values and prevents the overestimation of Q-values. Prioritized experience replays can also be used to prioritize training to rare experiences for better learning. Dueling DQN that estimates the advantage function and state functions and later combines them to estimate the action function can also be used.