Nevigation - Collect Banana in Unity Environment

Howard Cho

ABSTRACT

This project is using deep reinforcement learning to improve the DQN algorithm. This project examines four extensions to the DQN algorithm and implemented to analyze performances.

Keywords: DQN, DDQN, Dueling

INTRODUCTION

The deep Q-network is designed to produce a Q value for every possible action in a single forward pass. Several layers of nodes are used to build up progressively more abstract representations of the data. This project focused fully-connected layers to produce the vector of action values. The training techniques is focused on experience replay and fixed Q targets. The implementaion of this projec focused on improving the DQN. The implementations follow the progresses using double DQN (DDQN), and Dueling networks. It also has been updated deep neural network to modify to see model's improvement.

METHODS

First approach was taken to update DQN model to Dueling Network. In order to determine which state are valuable or not, DQN have to estimate the corresponding action values for each action. dueling architecture can assess the value of each state without having to learn the effect of each action. As shown in figure 1. The dueling network has two stream to separately estimate (scalar) state-value and the advantages for each action then the output module implements to combines them for both network output Q-values for each action. The module that combines he two streams of fully connected layers to output a Q estimate used:

$$Q^{\pi}(s,a) = V^{\pi}(s) + A^{\pi}(s,a)$$

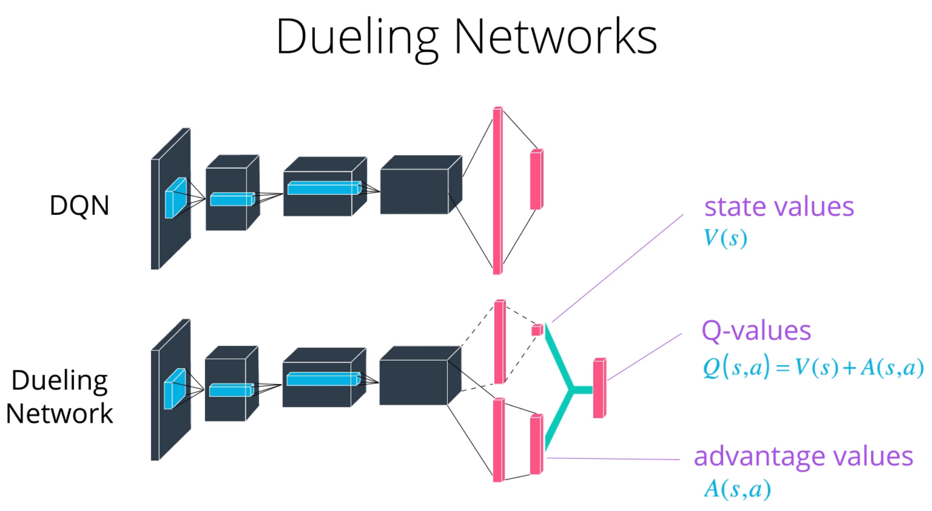

Figure 1. Single stream Q-networks(top), and the dueling Q-network(bottom)

Deep Q-learning tends to overestimate action values. Overestimation can occur when the action values are inaccurate, irrespective of the source of approximation error. The standard Q-learning update for parameters after taking action A_t in state S_t and observing the immediate reward R_{t+1} and resulting state S_{t+1}. Update parameter:

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

where,

$$TDtarget = R + \gamma max_a \hat{q}(S', a; w^-)$$

The target network, with parameters w^-, is the same as the online network except that its parameters are copied every time steps from the online network. Double Q-learning can prevent overoptimistic value estimates. In double DQN TD target;

$$TDtarget_{DDQN} = R + \gamma \hat{q}(argmax_a \hat{q}(S', a, w); w')$$

This second set of weights w' can be updated symmetrically by switching the roles of w and w'.

This project keep the following replay buffer which contains a collection of experience tuples (S,A,R,S'). The tuples are gradually added to the buffer as we are interacting with the environment. The purpose of using replay buffer is to prevent high correlation from the sequence of experience tuples while agent interacts with the environment.

Last, modifying network can helps to improved. Increase one more fully-connected layer quickly solved problems. However, I need to becareful with overfilling.

Plots

This result of plots shows that Figure7 reached average score quickly. DDQN-Dueling with three hidden layer methods performs very well.

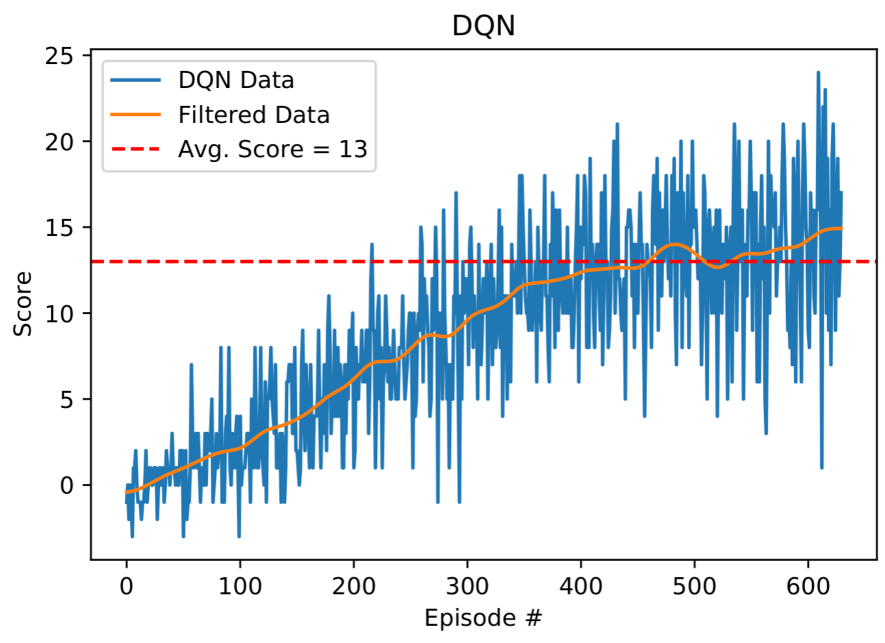

Figure 2. DQN Plot

Improvement Suggestion

I would like to implement Rainbow which will perform outstaning.

Ziyu Wang (2016) (Ziyu Wang, 2016).

Hado van Hasselt (2015) (Hado van Hasselt, 2015).

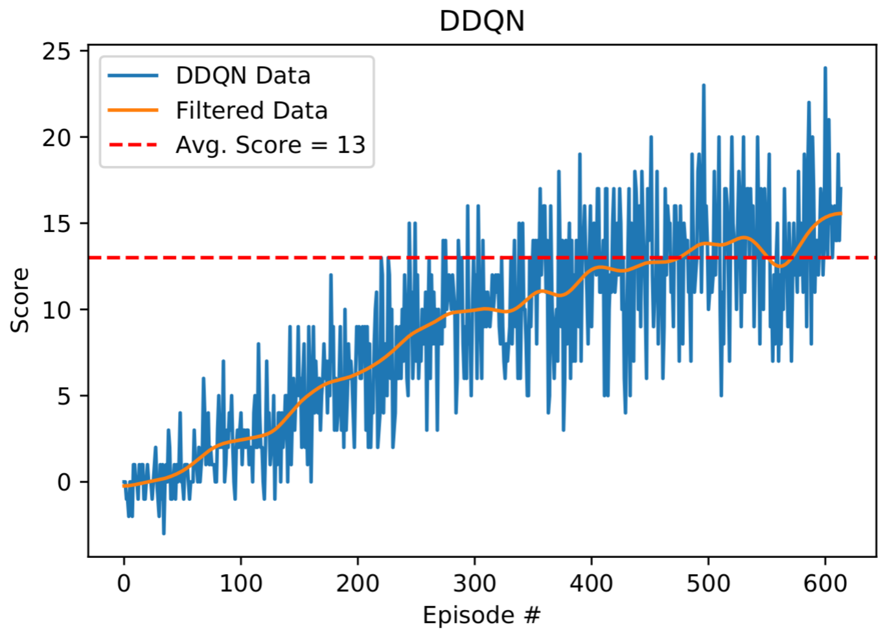

Figure 3. DDQN plot)

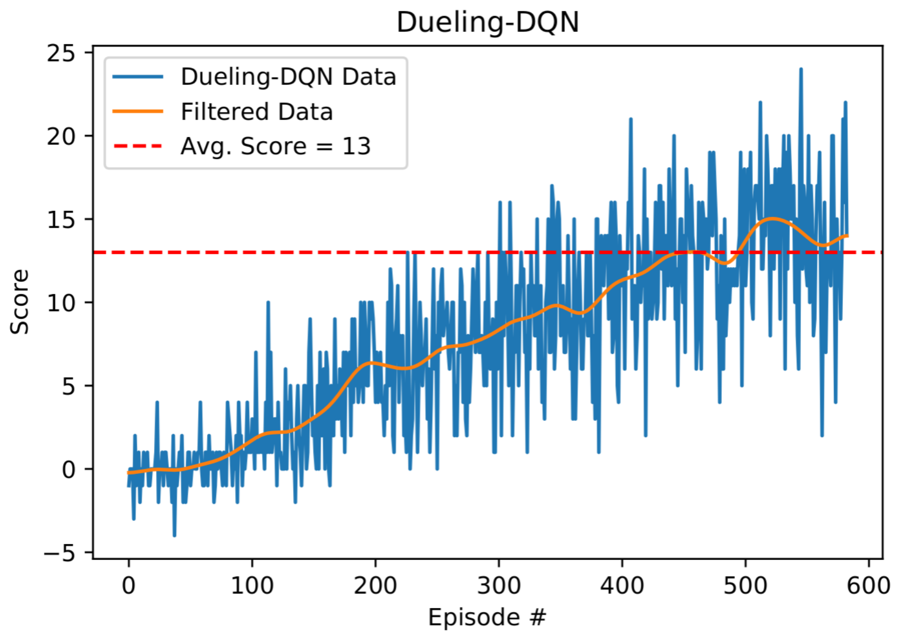

Figure 4. Dueling-DQN plot)

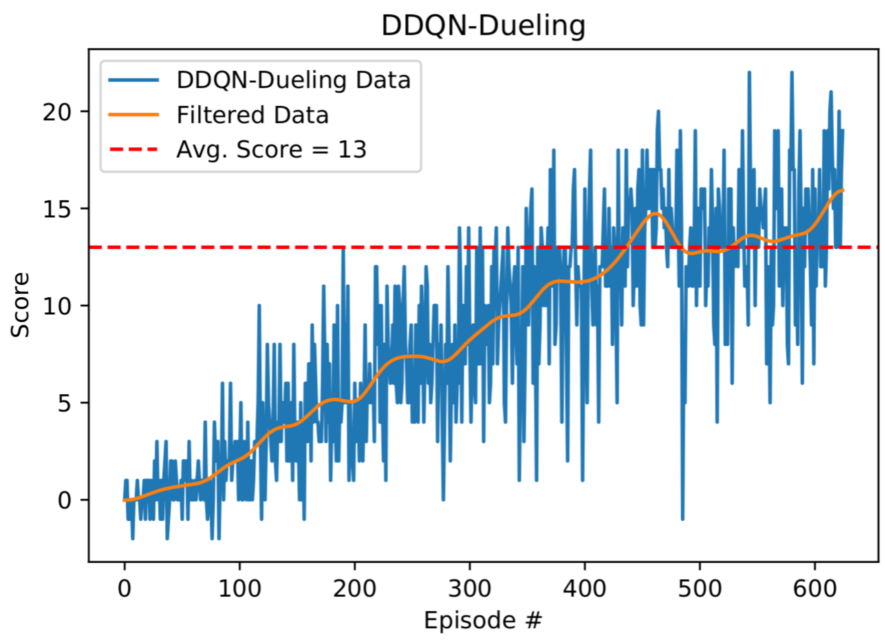

Figure 5. Dueling DDQN plot)

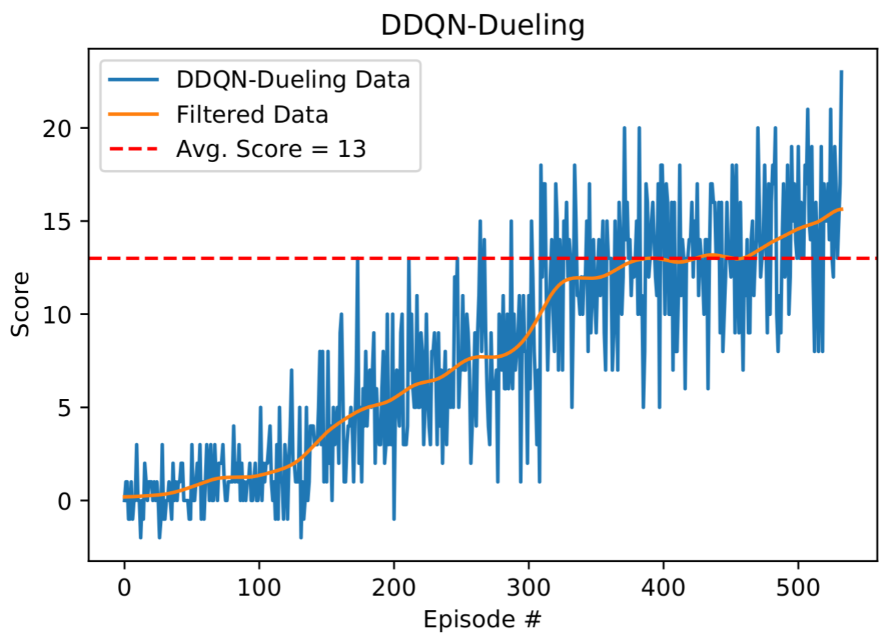

Figure 6. Dueling DDQN2 plot)

REFERENCES

Hado van Hasselt, Arthur Guez, D. S. (2015). Deep reinforcement learning with double q-learning. *CoRR*, (1509.06461):1–3.

Ziyu Wang, Tom Schaul, M. H. H. v. H. M. L. N. d. F. (2016). Dueling network architectures for deep reinforcement learning. *CoRR*, (arXiv:1511.06581):3–4.