Project 1: Deep-Q-Learning-for-Navigation

Udacity Deep Reinforcement Learning Nanodegree Program

Bob Flagg *

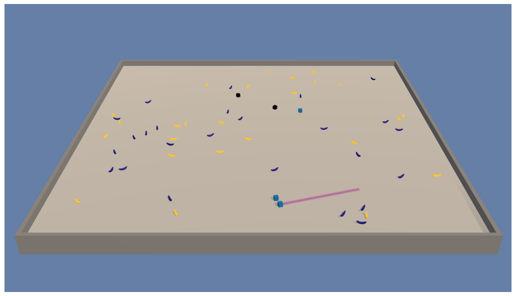

Figure 1: *Unity ML-Agent: Banana Collector*

Keywords: Reinforcement Learning, Deep Learning

1 Introduction

In this project I'll present the following three solutions to the Unity ML-Agent Banana Collector environment:

1. *Deep Q-Learning* [Mnih et al. 2015],
2. *Double Deep Q-Learning* [van Hasselt et al. 2015], and
3. *Dueling Deep Q-Learning* [Wang et al. 2015].

Source code in Python, using PyTorch, is available on github in the repo Deep-Q-Learning-for-Navigation.

2 Background

The Unity ML-Agent Banana Collector is a *sequential decision making problem*, in which an agent interacts with an environment over discrete time steps and seeks to maximize the expected *discounted return*:

$$G_t = \sum_{\tau=t}^{\infty} \gamma^{\tau-t} R_\tau,$$

where $\gamma \in [0, 1]$ is a discount factor that trades-off the importance of immediate and future rewards. See [Sutton and Barto 1998] for a general discussion of this sort of problem. In this specific example, the agent observes a 37 dimensional vector containing the agent's velocity, along with ray-based perception of objects around the agent's forward direction and tries to learn how to best select one of the following actions:

1. move forward,
2. move backward,
3. turn left, and
4. turn right.

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.

3 Deep Q-Learning for Navigation

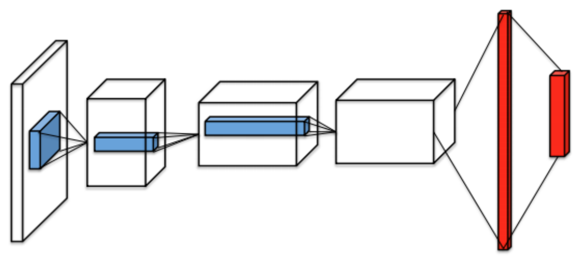

Figure 2: *Deep Q-Networt [Wang et al. 2015]*

In deep Q-learning the action-value function, $Q(s, a)$ is approximated with a *deep Q-network*, $Q(s, a|\theta)$, with parameters θ. To train this *local network*, we optimize the following sequence of loss functions at iteration i:

$$L_i(\theta_i) = \mathbb{E}\big[\big(y_i^{DQN} - Q(s, a|\theta_i)\big)^2\big],$$

with

$$y_i^{DQN} = R_i + \gamma \cdot \max_{a'} Q(s', a'|\theta_i^-),$$

where θ^- representes the parameters of a fixed and separate *target network*.

*Deep-Q-Learning-for-Navigation

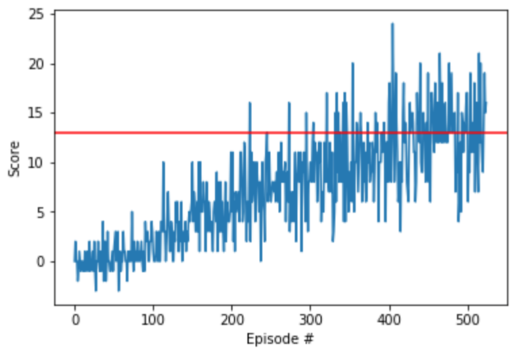

Figure 3: *Scores for Deep Q-Learning*

4 Double Deep Q-Learning for Navigation

In Q-learning and deep Q-learning, the max operator uses the same values to both select and evaluate an action. This can lead to over optimistic value estimates. *Double deep Q-learning* mitigates this problem by using a different target:

$$y_i^{DDQN} = R_i + \gamma \cdot Q(s', \arg\max_{a'} Q(s', a'|\theta_i)|\theta^-).$$

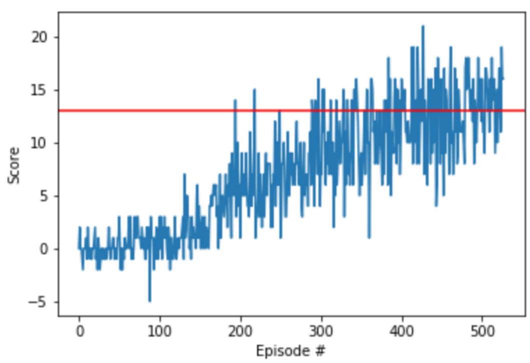

Figure 4: *Scores for Double Deep Q-Learning*

5 Dueling Deep Q-Learning for Navigation

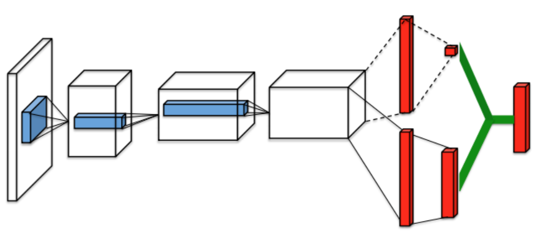

Figure 5: *Dueling Deep Q-Network [Wang et al. 2015]*

The architecture of the Q-network in dueling deep Q-learning has two streams, one estimates the state-value function, $V(s|\theta, \beta)$, and the other estimates an *advantage function*, which in one implementation has the form

$$A(s, a|\theta, \alpha) - \frac{1}{|\mathcal{A}|}\sum_{a'} A(s, a'|\theta, \alpha).$$

The estimate of the action-value function is then

$$Q(s, a|\theta, \alpha, \beta) = V(s|\theta, \beta) - \big(A(s, a|\theta, \alpha) - \frac{1}{|\mathcal{A}|}\sum_{a'} A(s, a'|\theta, \alpha)\big).$$

Note: I don't use convolution in the network for the Unity ML-Agent Banana Collector so there are no shared θ parameters in this example.

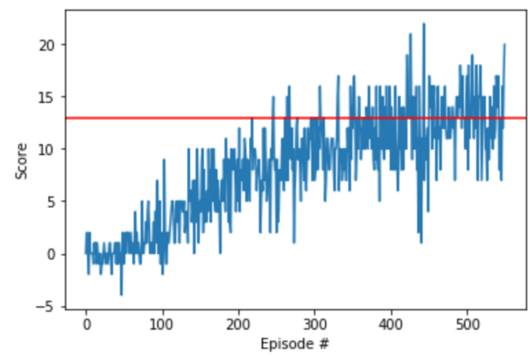

Figure 6: *Scores for Dueling Double Deep Q-Learning*

6 Improving Performance

References

MNIH, V., KAVUKCUOGLU, K., SILVER, D., RUSU, A. A., VENESS, J., BELLEMARE, M. G., GRAVES, A., RIEDMILLER, M. A., FIDJELAND, A., OSTROVSKI, G., PETERSEN, S., BEATTIE, C., SADIK, A., ANTONOGLOU, I., KING, H., KUMARAN, D., WIERSTRA, D., LEGG, S., AND HASSABIS, D. 2015. Human-level control through deep reinforcement learning. *Nature 518*, 7540, 529–533.

SUTTON, R. S., AND BARTO, A. G. 1998. *Reinforcement learning - an introduction*. Adaptive computation and machine learning. MIT Press.

VAN HASSELT, H., GUEZ, A., AND SILVER, D. 2015. Deep reinforcement learning with double q-learning. *CoRR abs/1509.06461*.

WANG, Z., DE FREITAS, N., AND LANCTOT, M. 2015. Dueling network architectures for deep reinforcement learning. *CoRR abs/1511.06581*.