Navigation Project

Troy Chevalier

August 18, 2018

The goal of this project was to train an agent to navigate (and collect bananas!) in a large, 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 your agent is to collect as many yellow bananas as possible while avoiding blue bananas.

Learning algorithm

Environment

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. Four discrete actions are available, corresponding to: 0 - move forward, 1 - move backward, 2 - turn left, 3 - turn right.

The task is episodic: in order to solve the environment the agent must get an average score of +13 over 100 consecutive episodes.

Model architecture

The solution is a variant of the Q-learning algorithm. Experience replay [1] is used to alleviate the problems of correlated data and non-stationary distributions. The idea is to learn the optimal action-value function $Q^*(s, a)$.

The optimal function obeys the *Bellman equation* identity. A function approximator is used to estimate the action-value function: $Q(s, a; \theta) \approx Q^*(s, a)$.

A non-linear function approximator (neural network) is used. Following the terminology of [2], we refer to the neural network function approximator with weights θ as a Q-network.

The architecture is a three layer feedforward neural network with two hidden layers each with 64 parameters.[3] The architecture uses leaky RELUs for the activation functions. The network architecture is depicted in Fig. 1.

The approach is *model-free* and it is also *off-policy*: it learns about the greedy strategy while exploring the state space. It follows the greedy strategy with probability $1 - \epsilon$ and selects a random action with probability ϵ.

[1] Long-Ji Lin. Reinforcement learning for robots using neural networks. Technical report, Carnegie-Mellon Univ Pittsburgh PA School of Computer Science, 1993

[2] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013

[3] The solution is based on the DQN code provided as part of the Deep Q-Networks lesson.

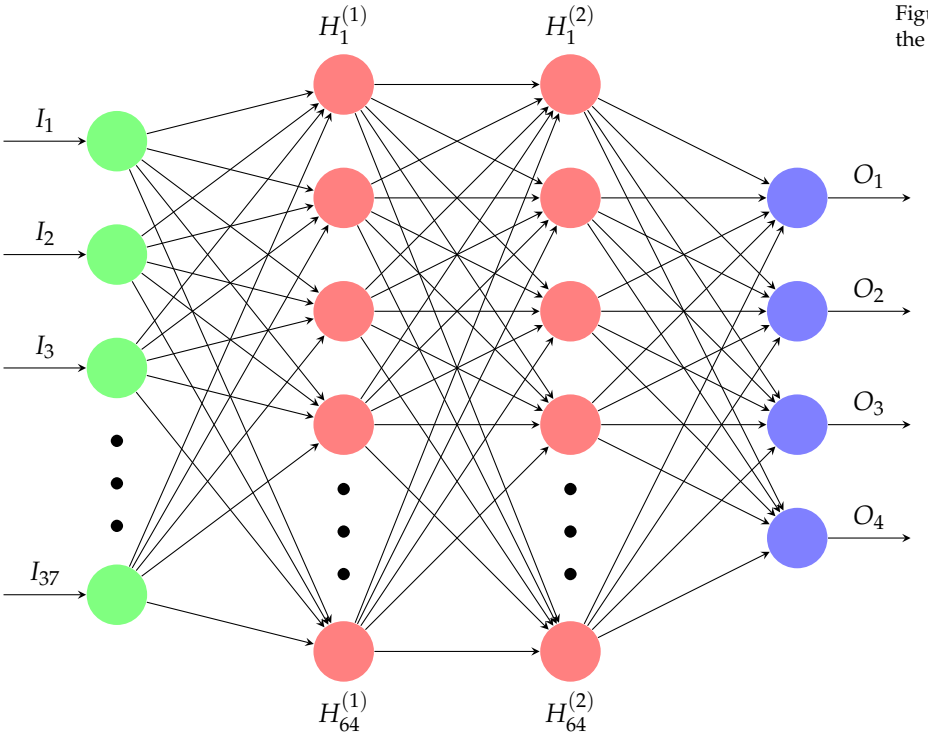

Figure 1: Q-network used to estimate the action-value function $Q^*(s, a)$.

Hyperparameters

I tested various configurations of hyperparameters, including a single hidden later and a larger network capacity (96 parameters for the hidden layers). The three layer neural network performed best.

Training

The environment is considered solved if the agent achieves an average score of +13 over 100 consecutive episodes. This wasn't very challenging, so I used a goal of +16 over 100 consecutive episodes.

Table 1 displays the average rewards at intervals of 100 episodes. The plot of rewards per episode is shown in Fig. 2. The agent solved my goal of +16 average rewards after 741 episodes.

Episode	Average score
100	3.04
200	7.47
300	11.46
400	13.94
500	15.07
600	14.79
700	15.47
741	16.03

Table 1: Average rewards over 100 episodes.

Future work

There are two ideas that I want to try for future work: prioritized experience replay and processing of frames using convolutional networks.

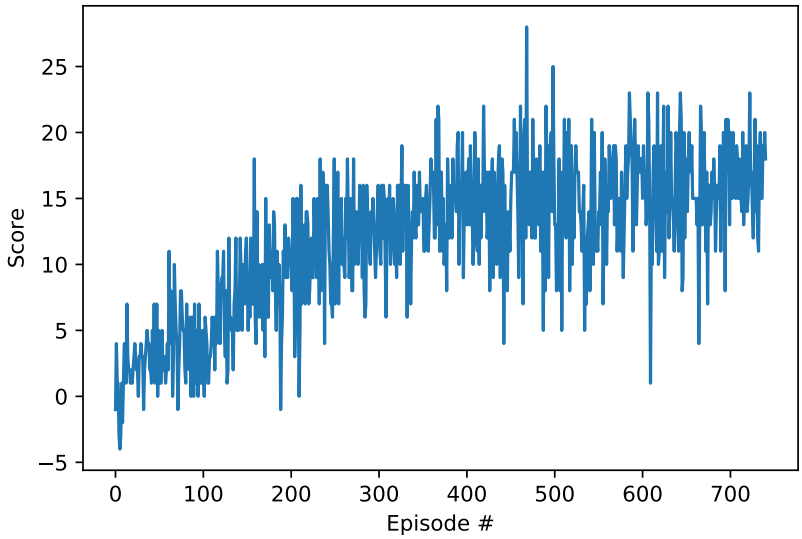

Figure 2: Rewards per episode.

Prioritized experience replay

As described in [4], uniform sampling of experiences from the replay memory does not take into account that some transitions are more important and hence does not learn efficiently. I would like to enhance the experience replay code to use prioritized experience replay.

[4] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. *arXiv preprint arXiv:1511.05952*, 2015

Convolutional networks

As demonstrated in [5], state of the art performance can be achieved through the use of convolutional networks trained directly on the pixel data.[6]

[5] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013

[6] Combining the existing 37 dimensions of state with the pixel data would likely lead to even better results.

References

Long-Ji Lin. Reinforcement learning for robots using neural networks. Technical report, Carnegie-Mellon Univ Pittsburgh PA School of Computer Science, 1993.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. *arXiv preprint arXiv:1511.05952*, 2015.