Navigation Project Report

Piero Macaluso

Introduction

The purpose of this document is to briefly present the work done in this project, along with the algorithm used and implemented, but also with a showcase of the results obtained.

This document can be divided into 4 parts:

- **Introduction**: the current section;
- **Learning Algorithm**: an overview of the approaches and methods used to solve the problem;
- **Results**: a presentation of the results reached with plots and GIFs of the best episodes;
- **Future Work**: some ideas on how to improve the actual algorithm.

Learning Algorithms

The main starting point of this work was implementing the DQN algorithm (Mnih et al. 2015). DQN algorithm was the first concrete contact point between Deep Learning and Reinforcement Learning, and it made applying Reinforcement Learning to environments with very large state and/or action spaces possible thanks to the usage of Deep Neural Networks. In the context of DQN, at each step, based on the current state, the agent chooses an action ϵ-greedily with respect to the action values, and adds a transition $(S_t, A_t, R_{t+1}, \gamma_{t+1}, S_{t+1})$ to a replay memory buffer (Lin 1992), that holds the last transitions. The parameters of the neural network are optimized by using stochastic gradient descent to minimize the loss

$$R_{t+1} + \gamma_{t+1} \max_{a'} q_\theta^-(S_{t+1}, a') - q_\theta(S_t, A_t))^2 \quad (1)$$

where t is a time step randomly picked from the replay memory. The gradient of the loss is back-propagated only in the *online network* denoted with θ parameters. Thanks to the usage of experience replay and target networks, DQN is able to learn with stability Q values.

To further explore the concepts, some improvements gathered from the state of the art were analyzed and implemented to improve the performances of the algorithm. The improvements implemented will be briefly presented in this section.

Double DQN The vanilla DQN is affected by an overestimation bias caused by the maximization in Equation 1. Double Q-Learning (Hasselt 2010) addresses this problem by decoupling the selection of the action from its evaluation. In this work, the target network was used to implement this improvement using the loss

$$R_{t+1} + \gamma_{t+1} q_\theta^-(S_{t+1}, \arg\max_{a'}(q_\theta(S_{t+1}, a')) - q_\theta(S_t, A_t))^2$$

Dueling Networks The dueling network (Wang et al. 2016) is a neural network architecture designed for value based RL. It has a common feature extractor and then two streams of computation, value function and advantage streams. In the approach of this work, they are merged at the end by with the following equation

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

Gradient Clipping Gradient Clipping (Goodfellow et al. 2016) clips the size of the gradients to ensure optimization performs more reasonably near sharp areas of the loss surface. It a very simple way to make learning more stable.

Hyper-parameters

The first part of the neural network (the *feature extractor*) is composed of the input layer and one linear layer with a hidden size of 64. After this part, there is the *advantage part* composed of one linear layer with a hidden size of 64 and an output layer with as many outputs as action space size. In parallel, there is the *state value* part which is the same as the previous one, but with an output size equal to one. The network exploits *Rectified Linear Unit* (ReLU) as non-linearity. The hyper-parameters used are presented in Table 1.

Results

The number of episodes to play was fixed at 1000, however, the agent took just 365 episodes to reach an average score on the last 100 episodes greater than 13.0. The highest average score on the last 100 episodes was equal to 16.56 reached in 710 episodes.

Hyper-parameters	Value
Memory Buffer Size	10^5
Batch Size	64
Gamma	0.99
Tau	10^{-3}
Learning Rate	5×10^{-4}
Learning Frequency	4 episodes
Epsilon (start, decay, min)	$(1.0, 0.99, 0.01)$

Table 1: Hyper-parameters used in the training process

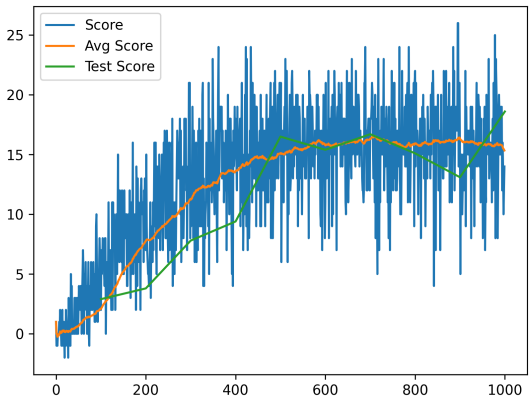

Figure 1: Training Score History

The training was not stopped upon reaching the requested value to determine the solution (after 365 episodes) because it was noticeable that the agent was not so stable. At that stage, the agent frequently got stuck in situations where, while performing complementary actions, he tended to remain stationary in one place as if it were stuck.

To find the best solution, a test phase of 10 episodes was implemented and started every 100 training episodes to evaluate the results without taking into account exploration uncertainties. As shown in Figure 1, the best solution was found at episode 1000 [1] with an average of 18.60 over 10 episodes. In this case, blocking situations have become much rarer.

Future Work

The results obtained were very encouraging and positive. Despite this, there are a lot of further improvement that is possible to apply to the DQN implemented in this project. Insights about this topic can be found in the Rainbow paper (Hessel et al. 2018), such as Prioritized Experience Memory Replay, Multi-Step Learning, Distributional RL and Noisy Nets to overcome the limitations of ϵ-greedy policies.

[1] GIF reporting a handful of seconds of the testing phase here

References

[Goodfellow et al. 2016] Goodfellow, I.; Bengio, Y.; Courville, A.; and Bengio, Y. 2016. *Deep learning*, volume 1. MIT press Cambridge.

[Hasselt 2010] Hasselt, H. 2010. Double q-learning. *Advances in neural information processing systems* 23:2613–2621.

[Hessel et al. 2018] Hessel, M.; Modayil, J.; Van Hasselt, H.; Schaul, T.; Ostrovski, G.; Dabney, W.; Horgan, D.; Piot, B.; Azar, M.; and Silver, D. 2018. Rainbow: Combining improvements in deep reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.

[Lin 1992] Lin, L.-J. 1992. Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine learning* 8(3-4):293–321.

[Mnih et al. 2015] Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A. A.; Veness, J.; Bellemare, M. G.; Graves, A.; Riedmiller, M.; Fidjeland, A. K.; Ostrovski, G.; et al. 2015. Human-level control through deep reinforcement learning. *nature* 518(7540):529–533.

[Wang et al. 2016] Wang, Z.; Schaul, T.; Hessel, M.; Hasselt, H.; Lanctot, M.; and Freitas, N. 2016. Dueling network architectures for deep reinforcement learning. In *International conference on machine learning*, 1995–2003. PMLR.