Optimal decision making for complex problems
Lunar Lander

Francois Delarbre Simon Lorent

Academic year 2017 - 2018

1 Introduction

In this project, we have chose to try to solve the lunar lander[1] probleme from openAi gym. To do so, we have tryied two methods, the first one, by adapting code found[2] for an other problem, which use deep convolutional Q-learning. For the second one we tryied to implement A3C algorithm by ourself.

The problem consist of landing the lunar lander on the moon. It always start at the same position but the background change each time the game is played. To land, it may perform four actions, right, left or main engine, or do nothing. It have unlimited fuel so it can fly as long as it may want.

2 Deep Convolutional Q-learning

Deep convolutional Q-learning, as the name says, make use of a convolutional neural network, which take as an input images of the problem, and output the Q value for each actions.

2.1 Eligibility trace

The eligibiltiy trace consist of taking in account more of the pasts reward than the normal Q-learning. In traditional Q-learning, one compute the temporal difference as $TD = r_{k+1} + \gamma max\hat{Q}(x_{k+1}, u)\hat{Q}(x_k, u_k)$. In the case of Eligibility Trace, one use

$$TD^{(n)} = r_{k+1} + \gamma r_{k+1} + \gamma^2 r_{k+2} + ... + \gamma^{n-1} r_{k+n-1} + \gamma^n max\hat{Q}(x_{k+1}, u)\hat{Q}(x_k, u_k)$$

This allow to take more in account what happened in the past. Thus, the neural network uses the predicted Q values by it output and the computed one to compute the mean squared error loss to update its weights.

2.2 Experience replay

For each action taken, the algorithm push to a buffer, this allows to have the benefits of experience replay, by sampling this buffer in order to train the neural network.

2.3 Performances

After a few epochs, the average reward on the last 100 steps is increasing, and it seems that it has understood that it has to land between the flags, but it seems that it takes longer to understand how to manoeuvring to land there.

[1]`https://gym.openai.com/envs/LunarLander-v2/`
[2]`https://www.superdatascience.com/artificial-intelligence/`

Appendix A

Installation

1 Needed components

In order to run the code, it is needed to install first openAI gym lunar lander v2 environement, then tensorflow is needed for the A3C part, the deep convolutional Q-learning requiere pytorch, cuda version to be installed. And thus it need a working version of cuda installed on a linux machine.