11155009 方品仁 HW2 Report

# Cartpole V1

## Introduction

This environment corresponds to the version of the cart-pole problem described by Barto, Sutton, and Anderson in “Neuronlike Adaptive Elements That Can Solve Difficult Learning Control Problem”. A pole is attached by an un-actuated joint to a cart, which moves along a frictionless track. The pendulum is placed upright on the cart and the goal is to balance the pole by applying forces in the left and right direction on the cart.

## Experiments

The experiment is aimed to find out the performance between basic Q-learning and Deep Q-learning (DQN) on a continuous action space. For discrete action space, basic Q-learning with a simple Q-table can store all possible states and actions. While for continuous action space, it’s not an easy work. A simple way is discretization. Split the action space into multiples bins. Then for an action value, we can then categorize the action to the closest bin. The action can then become a discrete action space. The performance of this method is determined by the number of bins. The more bins we can have, the more accurate action we can choose. But there is a tradeoff between the performance and the required training time and memory consuming.

Another way to deal with continuous action space is DQN, replacing the basic Q-table with a neural network. As other comparisons between traditional models and neural network, DQN tend to have a better performance than basic Q-learning.

## Training

### Basic Q-learning

* Number of bins: 7
* ε-greedy policy: Start with ε = 0.95
* Q-value update rule:
* Episode per run: 3000
* Decay: 0.045

### DQN

* Use experience replay and target network to stabilize the training process
* 2 hidden layers and outputs Q-values for each action
* **Epsilon-Greedy** policy is used to balance exploration and exploitation
* Optimizer: Adam optimizer
* Learning rate: 0.0002
* Epsilon: 0.95
* Gamma: 0.97
* Batch size: 32
* Replay buffer size: 10000
* Update target net for every 100 steps

## Result

一張含有 螢幕擷取畫面, 行, 文字, 繪圖 的圖片

AI 產生的內容可能不正確。 (basic Q-learning)

一張含有 螢幕擷取畫面, 繪圖, 行, 圖表 的圖片

AI 產生的內容可能不正確。 (DQN)

一張含有 螢幕擷取畫面, 繪圖, 文字, 行 的圖片

AI 產生的內容可能不正確。 (comparison)

|  |  |  |
| --- | --- | --- |
|  | Basic Q-learning | DQN |
| Average score over tests for 5 times | 146 | 500 |

# Atari, Assault-v5

## Introduction

You control a vehicle that can move sideways. A big mother ship circles overhead and continually deploys smaller drones. You must destroy these enemies and dodge their attacks.

## Experiments

1. Influence by replay buffer size

We know that small replay buffer size can cause overfitting to recent experience. Therefore, we tend to have a bigger replay buffer size like 10000. And my question is, will a more bigger buffer size lead to a better performance?

1. Influence by different CNN structure

[[reference](https://cs229.stanford.edu/proj2016/report/MeiYouChan-DeepQLearningOnArcadeGameAssault.pdf)] In the paper, the authors mention using max pooling layers in their convolutional neural network (CNN) for a few key reasons. Despite initially not wanting to introduce translation invariance (since the position of game entities is important), they found that max pooling helped compress their large state space into a smaller vector of size 768. This was crucial for making the network more manageable and efficient.

In the typical CNN in RL, max pooling is not recommended to be added into the model because they can reduce important spatial information that is crucial for decision-making. Since RL tasks, like game playing, often require precise understanding of the environment's state, including the position of objects, max pooling may hinder the agent's ability to make informed decisions. Retaining more detailed spatial features is typically preferred to improve the model's performance in such tasks.

1. Influence by different training frame area

The picture below is a frame of the game. In my opinion, the upper part of the frame (the score part and the mother ship) and the lower part (the part below the green line) is not important for the agent. Therefore, I only use the middle part to train the agent.

一張含有 螢幕擷取畫面, 文字 的圖片

AI 產生的內容可能不正確。

## Training

* Replay buffer size: 10000
* Stack frames number: 4
* Preprocess of a frame: clip the upper part and the lower part, grayscale conversion, resizing to (84, 84), and normalize to [0, 1]
* Update target net for every 1000 steps
* **ε-greedy policy** for action selection
* Epsilon decay: exponential decay
* Loss function: Smooth L1 Loss
* Optimizer: RMSprop optimizer with a learning rate 0.00025
* Warm up steps: 10000 warm up steps for collecting experience
* Number of Episodes: ideally 10000 episodes
* Clipped Rewards: The rewards are clipped to the range [-1, 1] to prevent overly large updates and maintain stability.
* Evaluation during training: average reward of the last 50 episodes

## Result

1. Influence by replay buffer size

I tried the buffer size with 10000 and 50000. And the result was that the bigger buffer size, 50000, didn’t significantly improved the performance. Instead, it slowed down the speed of training and caused the consuming of memory.

1. Influence by different CNN structure