MountainCar-v0

Environment

- 使用 OpenAI GYM 中的 MountainCar-v0
- Action space: 向左、向右、不動
- Observation space: 位置、速度

Reward

- 原始環境中給的 reward 是每個 step 沒有碰到旗子就回傳 -1
- 不過這樣的 reward 無法順利的訓練，因此使用 observation 中的位置資訊修改 reward 為 abs(position - (-0.5))，position 的範圍為 -1.2～0.5，所以這樣的修改下可以保證 reward 介在 0～1，而且當車子停在中間山谷的時候 reward 最小

Method

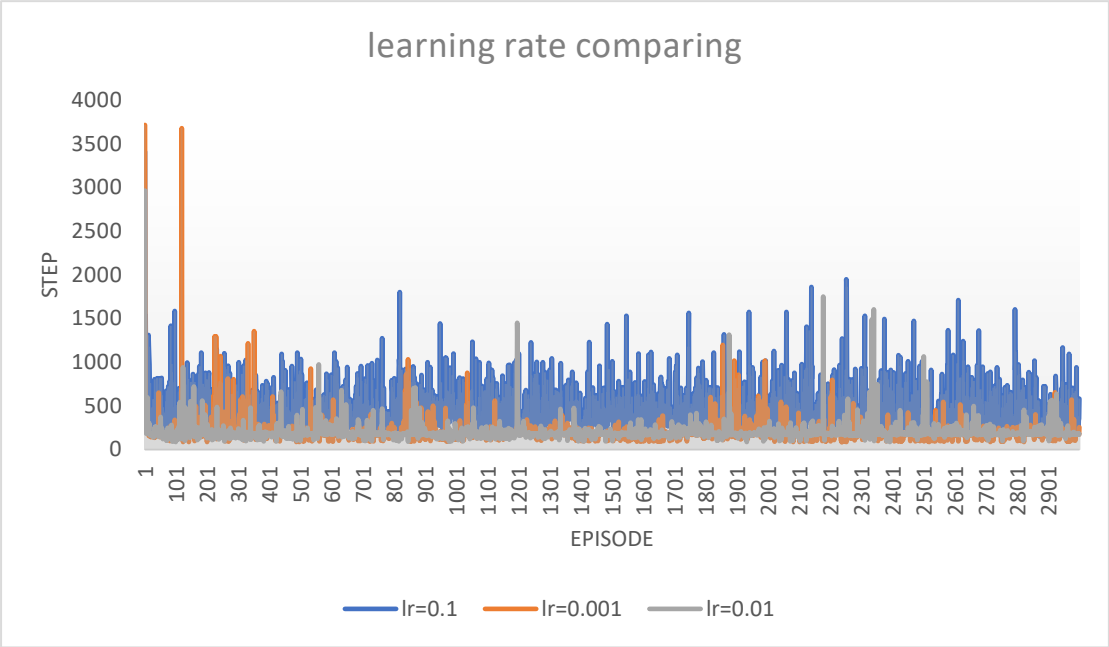
- 先用較傳統的 Q-learning 與 SARSA 實作，發現無法應付 Mountain Car 這樣的任務
- 因為 action space 為 discrete 的，因此採用 DQN 實作

DQN

- Q-Learning 是一個傳統的用 Q-table 做 action 的決策的方法
- 但是現實中的 state 與 action 很多，無法用單純的 table 存起來，因此 DQN 就是用一個 NN 來取代 table
- 此外 DQN 還加入了 Experience replay 和 Fixed Q-targets 機制

Analysis

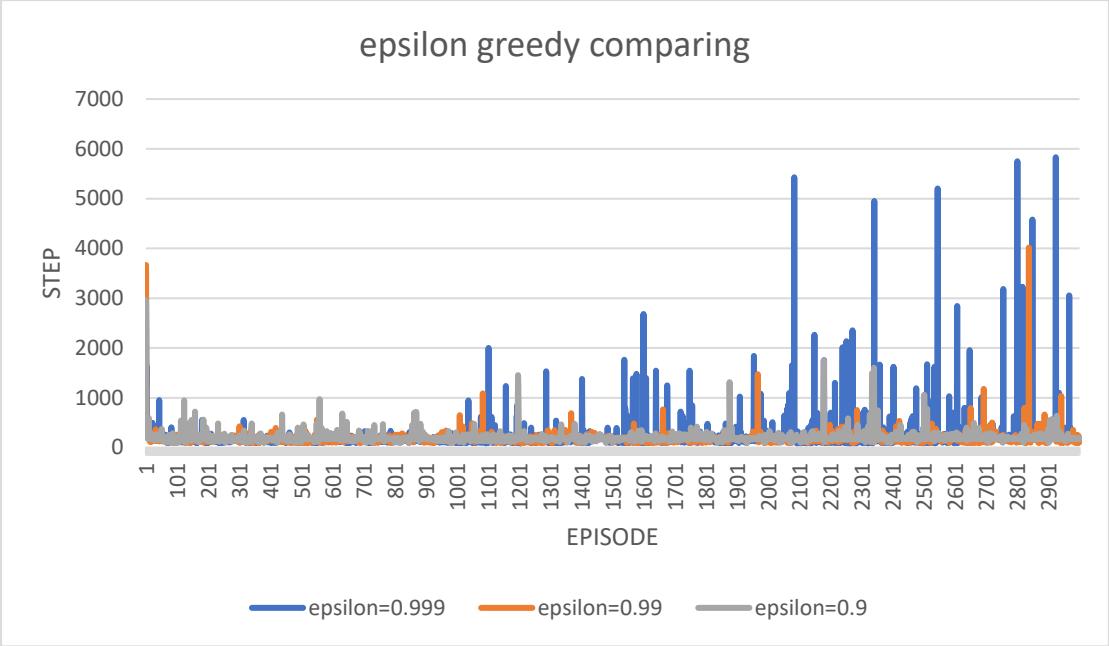
- Learning rate



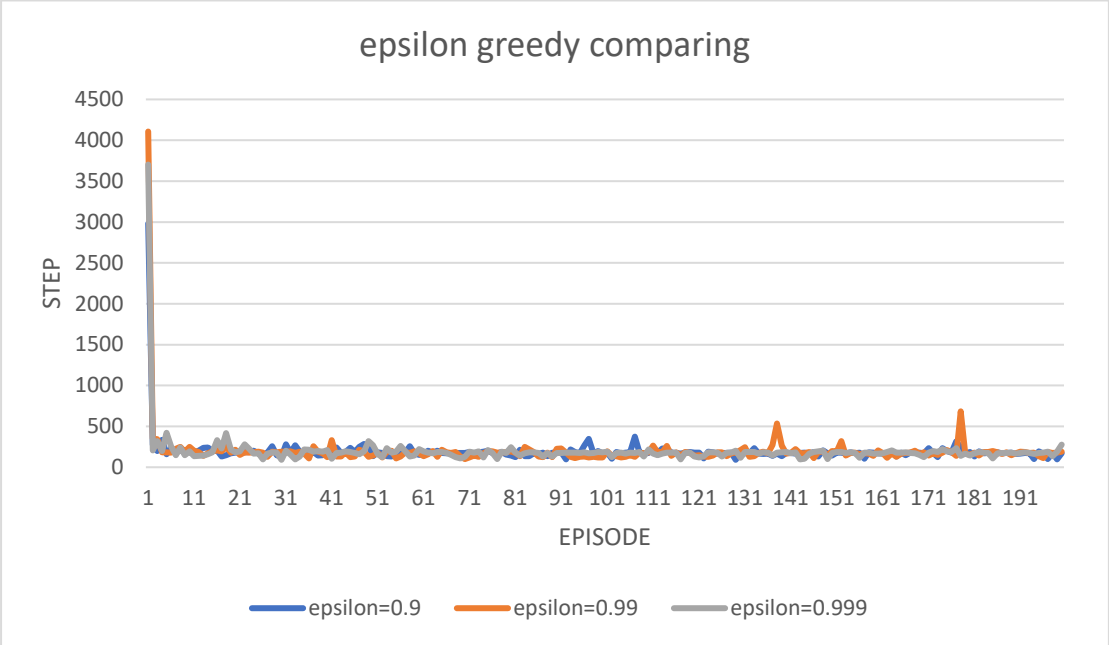
從不同 learning rate 的 learning curve 可以看出來，當 learning 為 0.01 時花費的 step 數量最少，因此選用 0.01 為 learning rate

- Epsilon greedy

Epsilon greedy 主要是控制 select action 時的隨機選擇機率

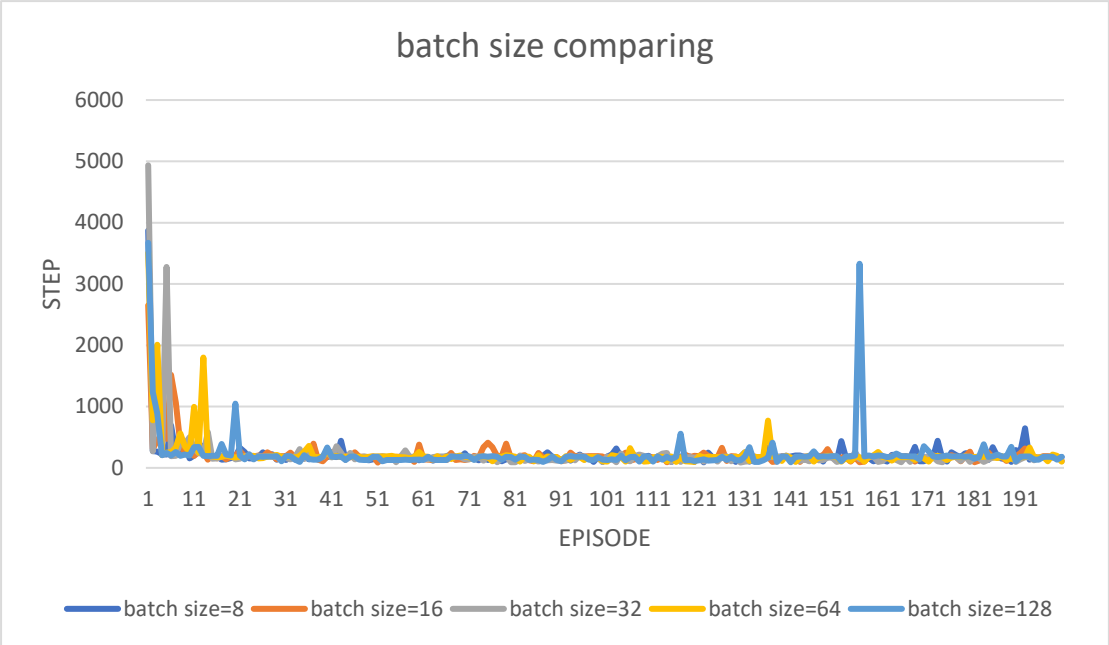


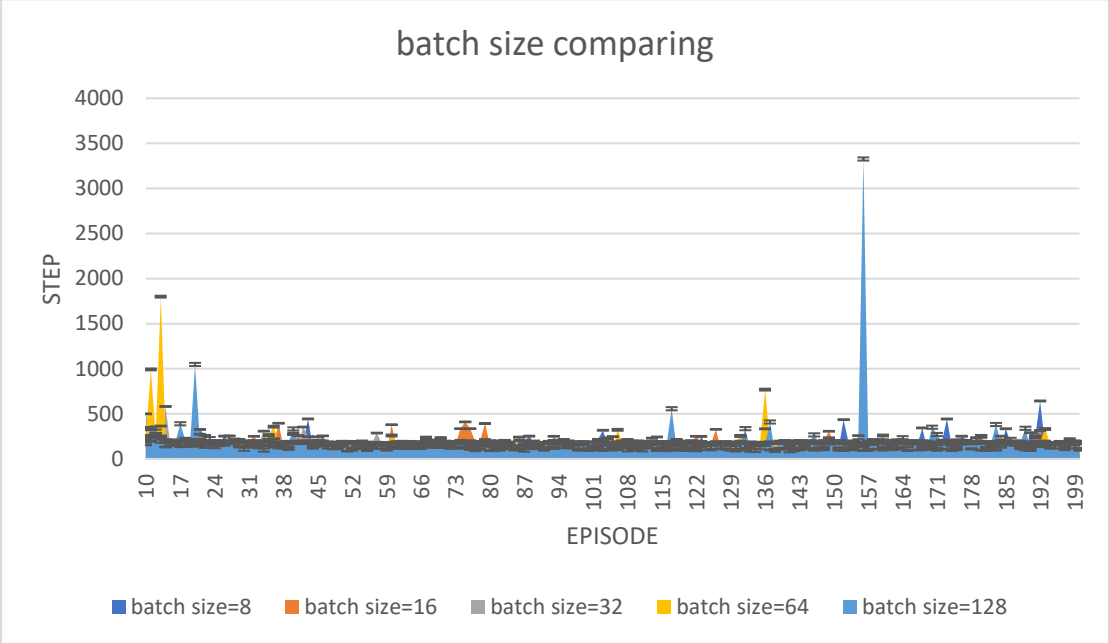
從結果的 learning curve 來看，可以發現 0.99 和 0.999 的後面的結果浮動較大，我推斷原因為 epsilon 變大之後，導致隨機選 action 的機率非常低，因此緊緊依靠 NN 來預測，因此這樣一個浮動的表現可能代表著 NN 訓練到 overfitting 了，並且看起 200 以內就可以收斂了，因此將上限 episode 調降至 200，調整後的 learning curve 如下圖



雖然差距不大，但 epsilon 為 0.999 時浮動是比較小的，經測試訓練時 epsilon 設為 0.1 與 1 皆無法訓練，所以將 epsilon 設為 0.999

- Batch size





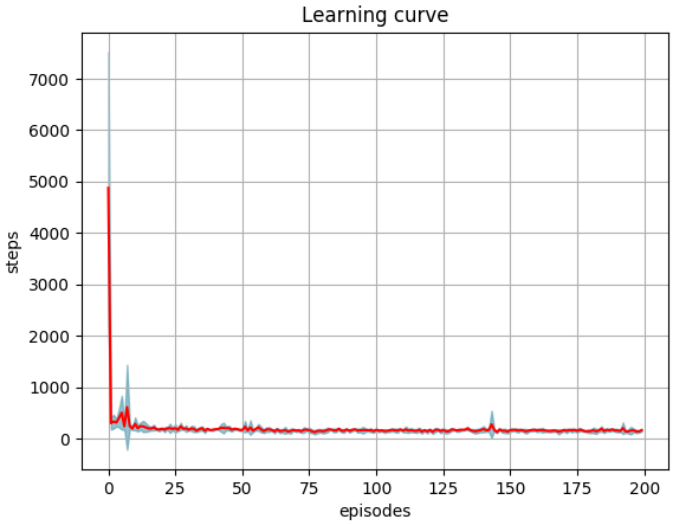
從曲線圖只可以發現 batch size 越大浮動會越大，但這樣不代表整體的結果，因此用 test.py 測試後結果如下表

Batch size	8	16	32	64	128
Step	168	246.6	143.7	179.3	178.3

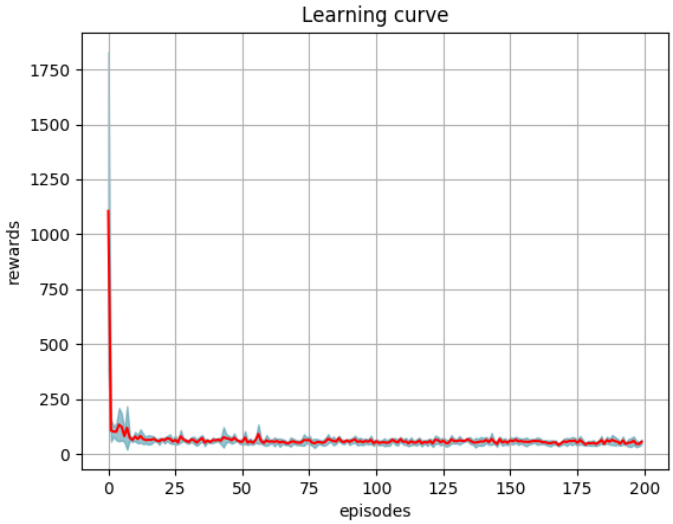
因此選用表現最好的 32 為 batch size

Result

- Learning curve (steps)



- Learning curve (rewards)



- Result:

  兩張 learning curve 的走勢是相近的，用 test.py 在測試後的平均步數為 152.9，平均 reward 為 58.8

Problem

1. What kind of RL algorithms did you use? value-based, policy-based, model-based? why?

   我所使用的演算法是 DQN，它是一個 model-free 的方法，因為無法事先得知所有 state 下的所有 action 會得到的 reward，因此採用的是類似 Q-Learning 這樣的從環境中得到反饋然後更新 Q-table 的方法

   DQN 是一個 value-based 的方法，因為 policy-based 的方法是較適用於 continuous 的 action space，而在 mountain car 中 action space 是 discrete 的，所以選用 value-based 的 DQN

2. This algorithms is off-policy or on-policy? why? (10%)

   DQN 是一個 off-policy 的方法，因為它是預測下一個 state 的最大 action value，但是下一個時刻並不一定會根據這個結果選 action

3. How does your algorithm solve the correlation problem in the same MDP? (10%)

   一般在做 mini-batch SGD 的時候都是假設其中的樣本是互相獨立的，但在 DQN 中的樣本都是從連續畫面而來的，所以說樣本之間的關聯性很大，可能會造成不收斂，因此加入 Experience replay 和 Fixed Q-targets 機制來解決這樣的問題

   Experience replay 是一個利用 memory pool 的方法，memory 中會存許多的樣本，當要訓練的從裡面隨機取 batch 數量的樣本，這樣可以減低樣本間的關聯性

   Fixed Q-targets 則是會使用兩個結構相同但參數不同的神經網路，並在一段固定的時間才同步兩者，這樣也可以有效地打斷關聯性