DSAI-HW4 Report

Intro

利用 RL 解決 Mountain-Car 問題

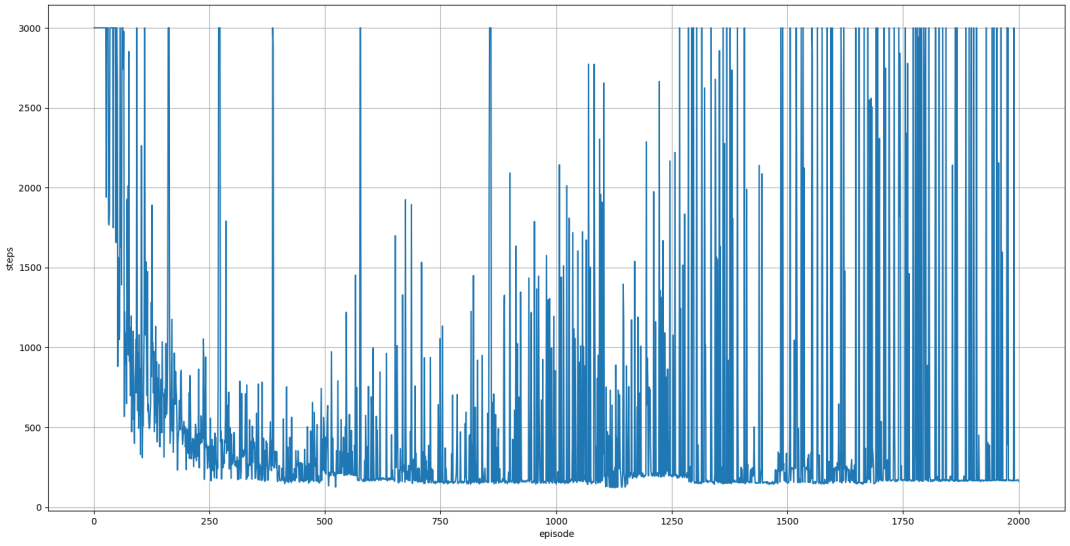
Implement

- 以 **Q-Learning** 實做，方法參考自 莫烦PYTHON (https://morvanzhou.github.io/tutorials/machine-learning/reinforcement-learning/2-2-tabular-q1/)
- 設定 `env._max_episode_steps = 3000` 避免 200 steps terminal
- 調整 reward ，將走到山頂的 reward 設為 +100，而每個 step 得到 -1 無改變
- 因為 observation 中的 position 及 velocity 皆為連續值，將他們離散化，在 min 和 max 之間切為15等分

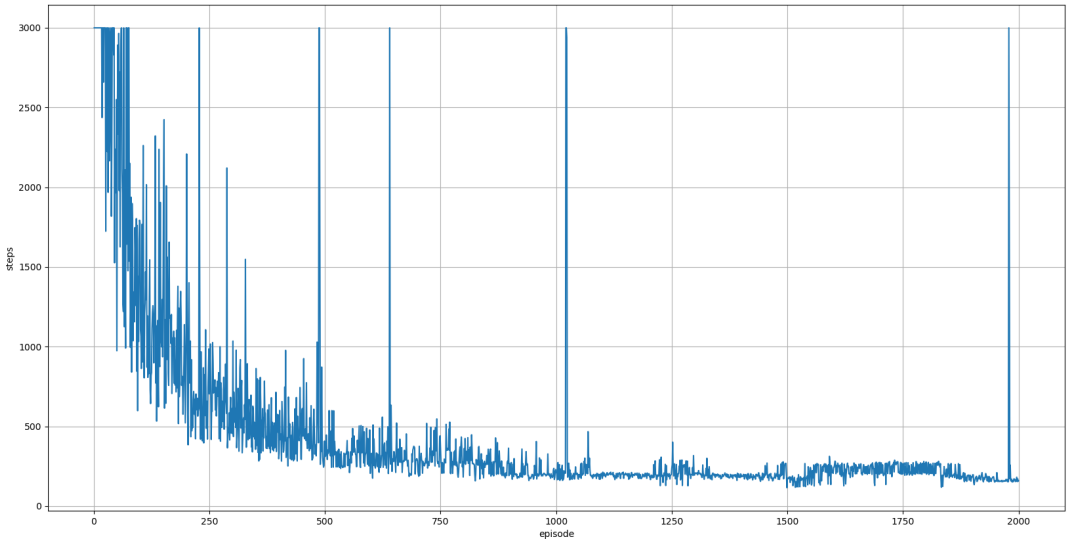
Analysis

將 **position** 及 **velocity** 給離散化

- pos 10等分 * vel 10等分



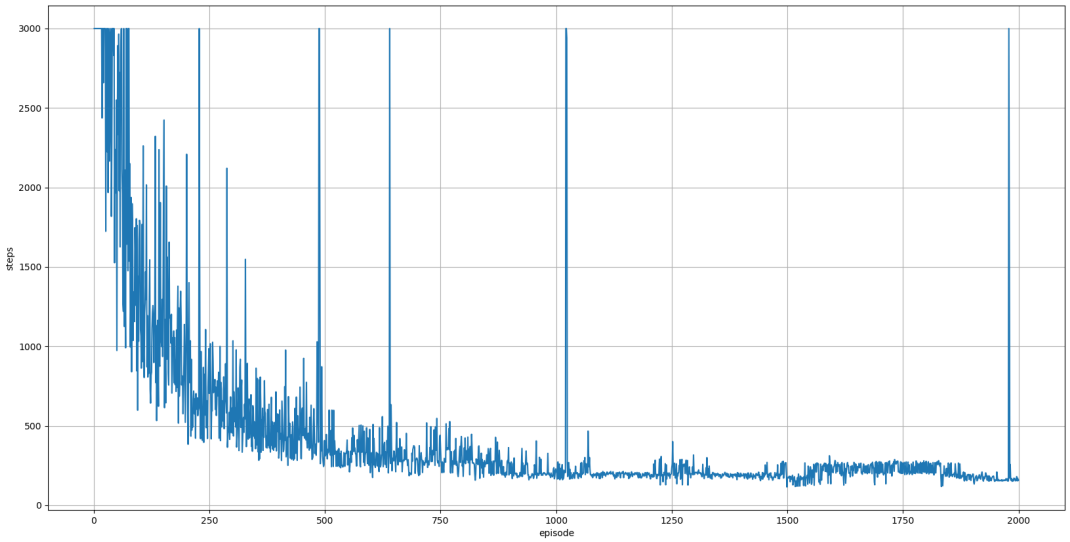
- pos 15等分 * vel 15等分



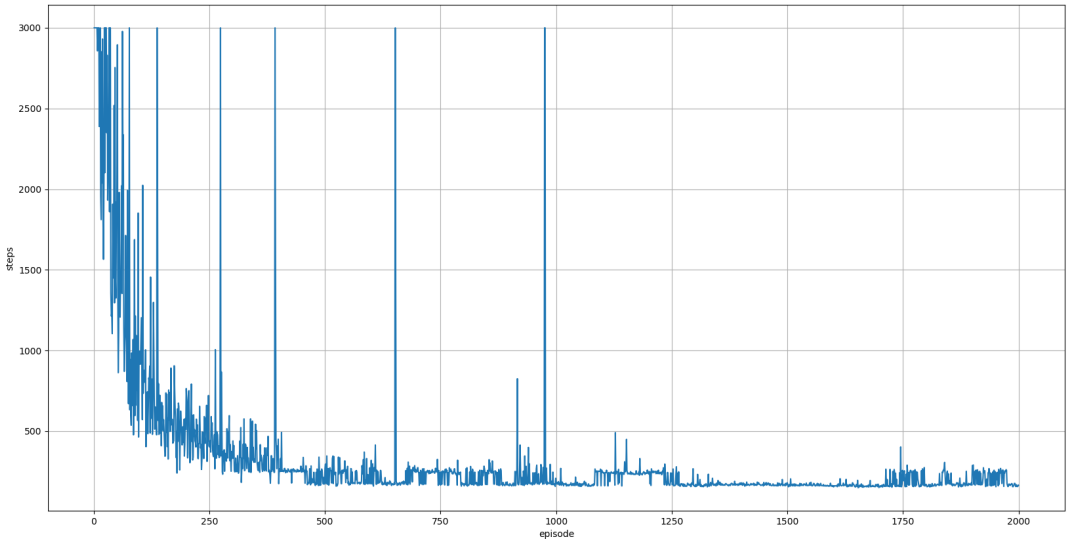
- 發現：
  - 將 state 切的越多，得到的 steps 數量也都在 120-160 之間浮動，並不會因為 Q table 變大，效果會越好。原本認為是因為 epsilon 不夠大，因此有些 state 沒有探索到，但在調整 epsilon 後，performance 也沒有更好。
  - 而在 10x10 及 15x15 比較中，發現 Q table 越大，有助於整體表現的穩定，較少發生 3000 steps 走不完的問題。

調整 **learning rate**

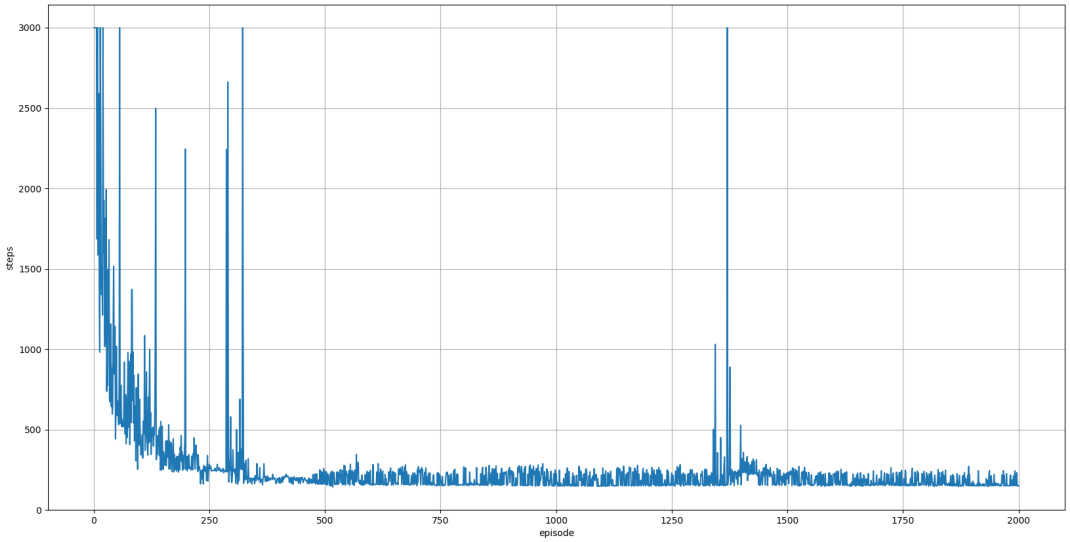
- lr = 0.01



- lr = 0.02



- lr = 0.05



- 發現：
  - learing rate 調整的較大，可從曲線中看出，steps 下降的較快，而在此案例中，也沒發生 performance 下降的問題。

Questions

1. What kind of RL algorithms did you use? value-based, policy-based, model-based? why?
   - Q-Learning
   - value-based
   - 藉由尋找 Q table 中最大的 value，來決定此刻的 action，所以是 value-based
2. This algorithms is off-policy or on-policy? why?

- off-policy
- 依照下圖可以看出在 Q-Learning 中，choose action 和 update Q 是用不同的 policy，所以是 off-policy，而 Sarsa 則是 on-policy

When I was learning this part, I found it very confusing too, so I put together the two pseudo-codes from R.Sutton and A.G.Barto hoping to make the difference clearer.

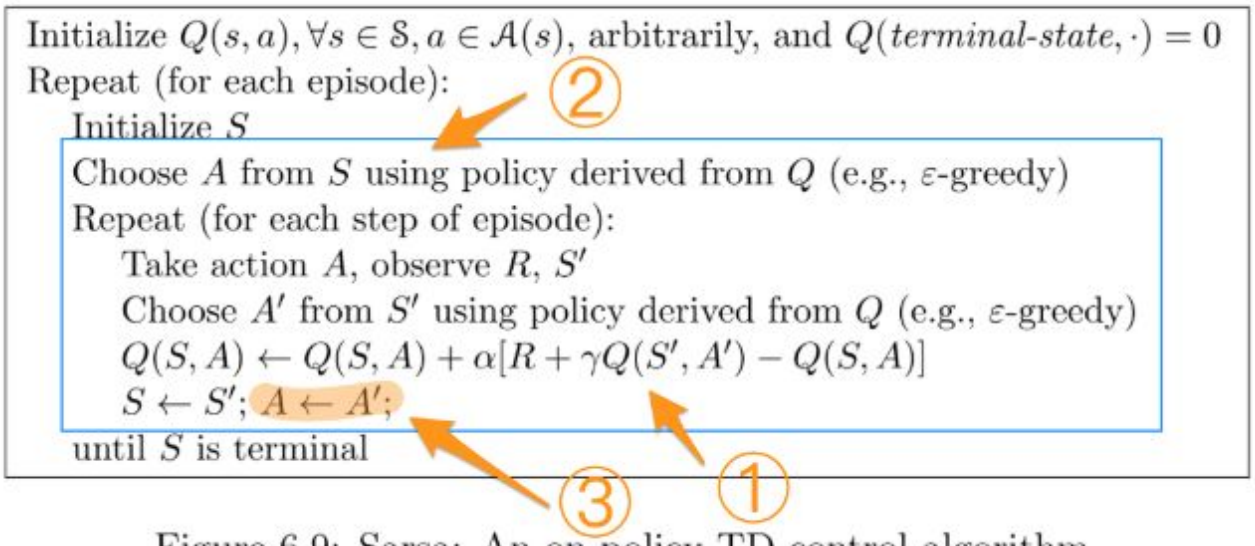


Figure 6.9: Sarsa: An on-policy TD control algorithm.

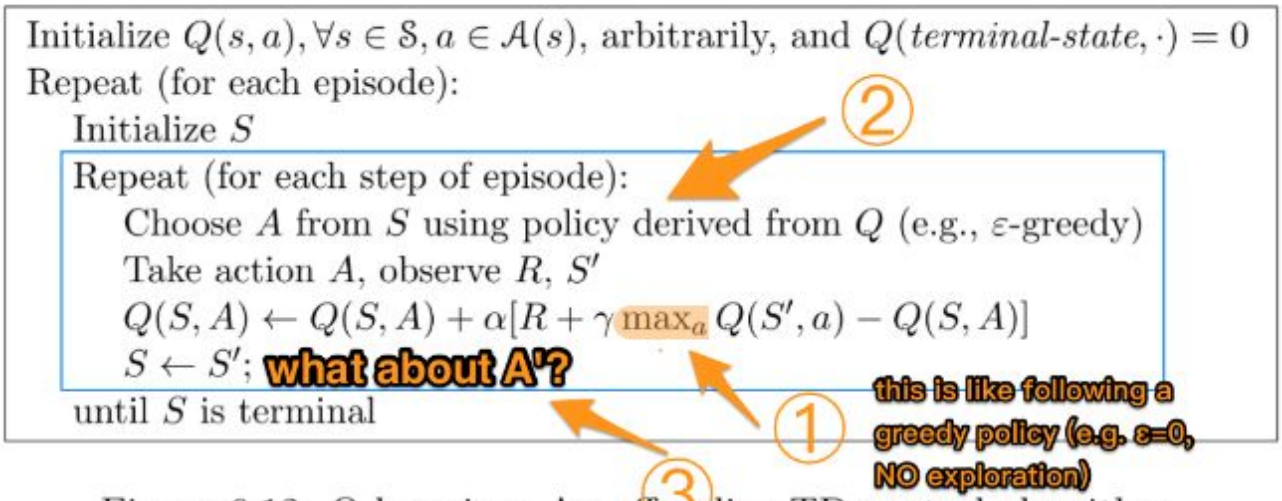


Figure 6.12: Q-learning: An off-policy TD control algorithm.

Blue boxes highlight the part where the two algorithms actually differ. Numbers highlight the more detailed difference to be explained later.

3. How does your algorithm solve the correlation problem in the same MDP?

- 在 mountain-car 中，將 continuous state 給離散化，轉換為 MDP 的問題，再利用 Q-Learing 解決
- 因為是連續的問題，所以 sample 會有 correlation problem ，造成 varience 太小，要解決這問題，可以 random sample 或是 某些 sample 不做 update。