Texas Hold'em Poker

B10705013 資管三 陳彥廷

Export System

在作業的一開始，我嘗試使用類似專家系統的方式來設計我的 ai，將我自己平常玩德州撲克的方法透過 if-else 判斷式來實現。

具體來說，我主要的判斷參數包括我的手牌以及公共牌、當前所在的 street、當前的賭注這三項，主要會依據牌型的強度以及當前的賭注來決定是否要跟注、加注或是棄牌。

這個方法的實驗結果如下，可以觀察到在 Max Round 提升的情況下，Win Rate 也會提升。
我認為主要的原因在於我的策略相對較為保守，
很容易前幾回合就出現棄牌的狀況，而 baseline 則大多不會棄牌，
所以贏的局幾乎都是大比分的局 (因為演算法再拿到比較大的牌時 raise 會較為積極)，
而輸的局我的分數也都是落在 800-1000 之間，
很少出現輸光的情況。

Baseline	Game Round	Max Round	Win Rate
baseline1	20	20	0.3
baseline1	20	50	0.5
baseline2	20	20	0.2
baseline2	20	50	0.35
baseline3	20	20	0.4
baseline4	20	20	0.05

Monte Carlo

第二種方式我選擇的是 Monte Carlo 加上一些 domain knowledge。
Monte Carlo 是一種透過大量隨機模擬來計算勝率的方法，
透過這個方法來計算當前手牌加上公共牌的勝率，
並依據一些 domain knowledge 來決定是否要跟注、加注或是棄牌，
勝率越高 (>0.9 or >0.7) 就會大幅度的 raise，
勝率一般的話則會參考當前的賭注來決定是否要跟注或是棄牌，

勝率如果過低有時候會選擇 random fold，
另外，如果當前落後或是剩餘回合不多則會進入狂暴模式，
大幅度的 raise 來試圖翻盤。

因為要將時間壓在 5s 之內，所以我只能模擬 50000 次 (模擬越多次預測的勝率會越精準，但數字越高邊際效應越強)，
以結果來說，這個方法的不確定性很高，從 baseline3 開始勝率的變動都很大，
輸的場結果大都也在 950-1000 左右，
但整體來說依舊比 Export System 的結果好。

Baseline	Game Round	Max Round	Win Rate
baseline1	10	20	1.0
baseline2	10	20	0.9
baseline3	10	20	0.7
baseline4	10	20	0.2
baseline5	10	20	0.4
baseline6	10	20	0.7
baseline7	10	20	0.3

Deep q-learning

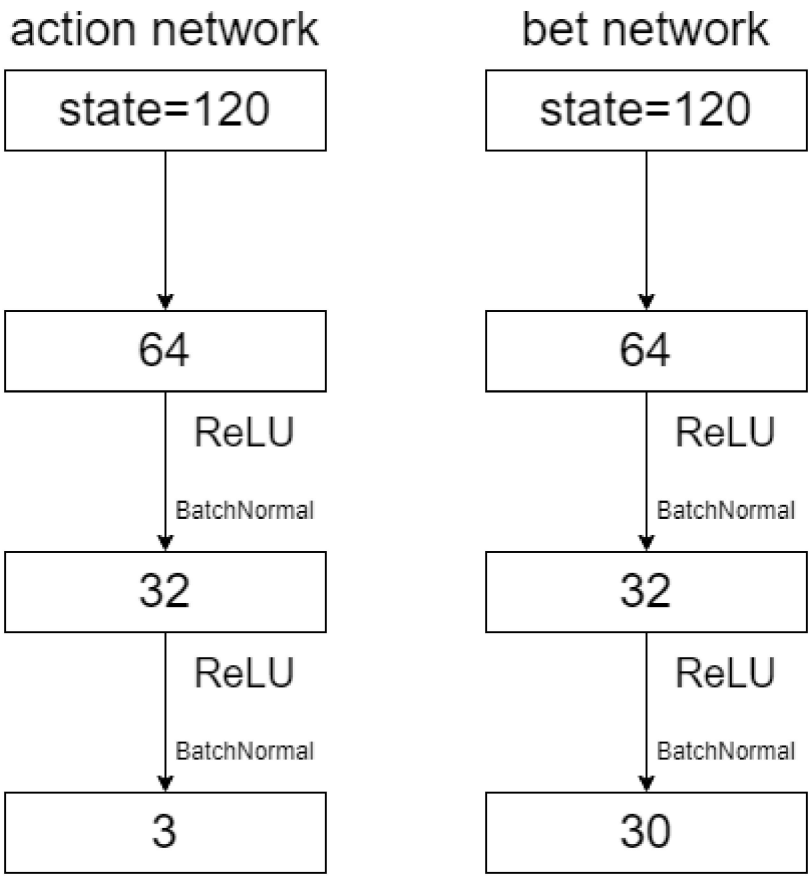
第三種方式我參考網路上的方式實作了 deep q-learning 的方法。
這個方法主要是透過 Q table 來記錄每個 state 下的 action 的 value，
並使用深度學習以及梯度下降來當作更新的方法。
我所對 action network 的 loss function 是使用 MSE，
而對 bet network 的 loss function 是使用 cross entropy，
並把兩個相加作為最終的 loss function。

除此之外，我也使用的 replay buffer 以及 epislon greedy 的方式來增加訓練的效率。
replay buffer 是一個用來記錄過去的 state、action、reward、next state 的資料結構，
使我們在梯度下降的時候可以隨機取樣以增加訓練的效率，
而 epislon greedy 則是一種隨機選擇 action 的方法，
在模型剛開始訓練的時候會有一定的機率隨機選擇 action，
隨著訓練的進行這個機率會逐漸減小。

而我使用的 reward function 策略除了勝負之外，也會特別降低 fold 的 reward 來避免大量的 fold，
也會使用蒙特卡蘿模擬的勝率來當作 reward 的一部分，最後也會針對最後贏的籌碼數來給予

reward。

我將當前的手牌以及公共牌還有賭注的大小當作 state，做出了以下的神經網路模型，
action network 會輸出 fold、call、raise 的機率，
bet network 會輸出 raise 的大小。



最後在訓練的部分，為了使模型更加通用化，
我會隨機選擇 baseline4 到 baseline7 來當作對手，
並做 20 round 的對戰進行 500-1000 次的訓練，
來使我的模型不會過度擬合某一種對手的策略。

但是最後的結果並不如我的預期，我認為可能在於 state 數量太多導致模型不太好收斂，
導致我在觀察結果時常常感覺模型在學習的過程中有點不太穩定，
有時候會出現一些很奇怪的行為，例如在自己有最大牌的情況下 fold，
或是在已經 raise 很多的情況下直接 fold 牌，
導致不管是輸贏幾乎差距都非常大。

Baseline	Game Round	Max Round	Win Rate
baseline1	10	20	0.7
baseline2	10	20	0.4
baseline3	10	20	0.8

Baseline	Game Round	Max Round	Win Rate
baseline4	10	20	0.3
baseline5	10	20	0.3
baseline6	10	20	0.3
baseline7	10	20	0.3

觀察到以上的現象之後，我嘗試降低 state 的數量，像是改成只使用當回合的操作搭配蒙特卡羅模擬的勝率來當作 state，以及嘗試調整 reward function 來增加模型的穩定性，
也有調整過神經網路為一層來使模型更容易收斂，
但最後的結果依舊不太理想，所以我最後選擇放棄這個方法。

Conclusion

在這些方法中，我認為 deep q-learning 的方法最有潛力，
但是由於我在訓練的過程中遇到了一些問題，
使得最後的結果不太理想，
因此在比較了三種方法後，我最後選擇使用蒙特卡羅來當作我最後的策略，
因為這個方法在我的測試中表現最好。

P1	P2	Game Round	Max Round	Win Rate (P1 : P2)
Expert	MCTS	10	20	1:9
Expert	DQN	10	20	4:6
MCTS	DQN	10	20	8:2

Reference

1. Deep Q-Learning
2. Monte Carlo