人工智慧導論 期末報告

B09703028
財金三呂紹嵘

一、摘要

此專題以訓練德州撲克 AI 為主題，使用各種不同方法來讓自己的 Agent 在五戰三勝、每戰 20 局的遊戲中將勝率最大化。這次的專題中使用了 Deep Q Learning 和 Monte Carlo Simulation 等方法來嘗試訓練 Agent，並使用了 Tensorflow 來進行 Neural Network 的建立和 pickle 儲存 preprocess 好的資料。在訓練完後能贏過 Baseline 0~5 的 Agent，詳細內容將在下文進行更深入的分析。

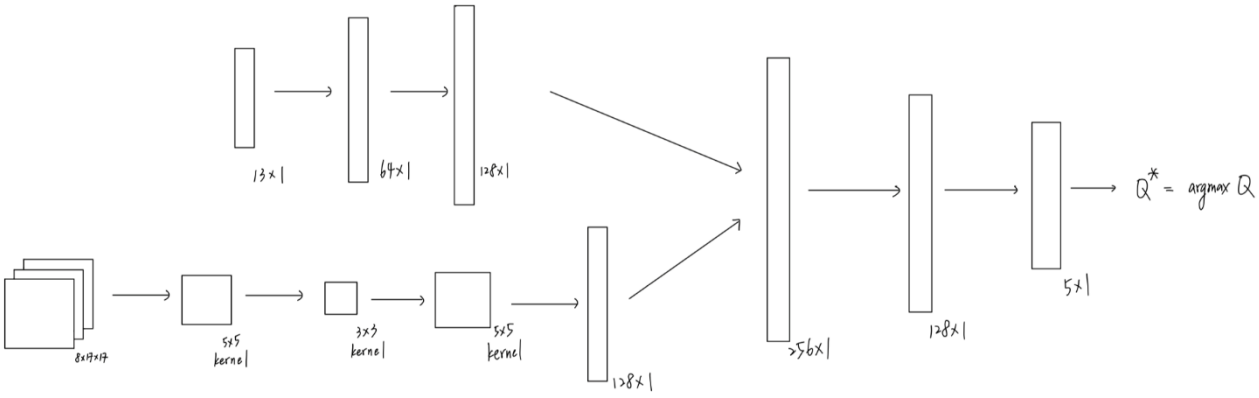
二、方法

1. Deep Q Learning

此方法將的 Deep-Q network 的 input 可以分成兩種。第一種是將每一輪的資訊包括大盲注、小盲注、獎金池、每人持有金額、與位置等資訊化成一個向量，之後用 one-hot-encoding 將 street 資訊(preflop、flop、turn、river)化成一個四維向量再加上 pretrain 好的起手牌贏面一起加入原始向量中使其變成一個 13 x 1 的一維向量；第二種則是將現在 state 中自己跟場上牌型化成 8 個 4 x 13 的矩陣，每個矩陣所對應到的牌種是 1 其他為 0。前兩個是自己的牌型，後五個則是場上的牌型，最後一個則是前 7 張牌的疊加以考慮交互作用。以 hole cards ST、SQ，community cards DJ、DQ、DK、CQ 為例，所得出的第八個矩陣為下圖所示。

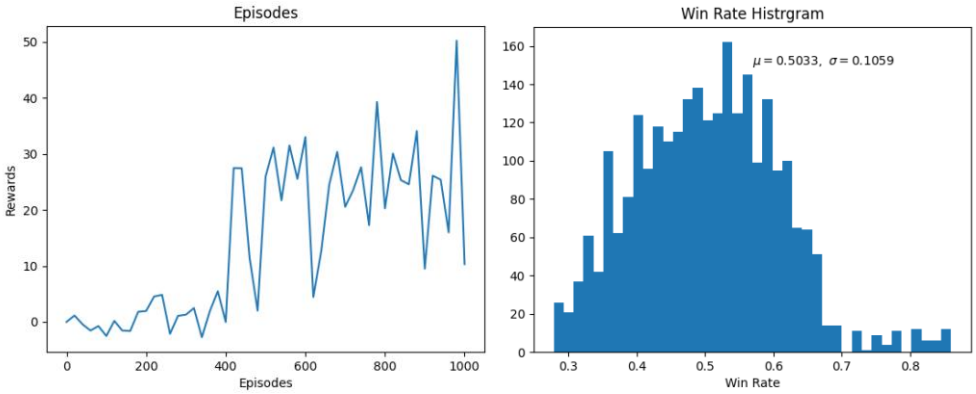
	2	3	4	5	6	7	8	9	10	J	Q	K	A
♣	0	0	0	0	0	0	0	0	0	0	1	0	0
♦	0	0	0	0	0	0	0	0	0	1	1	1	0
♥	0	0	0	0	0	0	0	0	0	0	0	0	0
♠	0	0	0	0	0	0	0	0	1	0	1	0	0

建立出的 Neural Network 是以上述的一維向量與二維矩陣維輸入，一維向量會通過兩層的 hidden layer，二維矩陣則會通過三層的 hidden layer 後再 Flatten 成一個一維的向量，最後將此二向量合併後再通過兩層 hidden layer 得到最後的輸出，過程中都會進行 dropout 的以降低 overfitting 的機率(參數參考 offthewallace 所設進行修改)。本次的輸出(出手策略)可分為五種: Call、Fold、Raise Min、Raise Max、和 Raise to Win，詳細流程如下圖所示。



針對 Reinforcement Learning 中訓練 hyperparameter 的部分 $Q(s,\ a) = r + \gamma * \max_{a'} Q\ (s', a')$，其中 r 是以每一輪中所獲得/損失的金額 x 取對數，即: $\text{sign}(x) \times \ln(1 + |x|)$。折現因子 γ 設為 0.99，總共訓練了 1000 個 Episode，從開始到結束每局會有 e 的機率會隨機選取動作，e 會從 1 隨著 Episode 增加而線性降低至 0.3。除此之外，在正式開始 train 之前有 1000 個 pretrain steps 來讓 Agent 去隨機摸索該採取的行動。在訓練的 1000 個 Episode 中，前 200 個 episode 是讓我的 Agent 隨機和 Random Agent 和 Call Agent 進行訓練，中間的 600 個 episode 加入了 baseline 1-4 一起進行訓練，最後的 200 個 episode 則是和所有的 baseline 進行訓練。我將每 100 個 episode 的訓練模型都存下來，總訓練時長約 3 小時，最後選用 model_1099.ckpt 作為讀取檔案。每個 episode 所得到的 reward 如下圖所示，可見 Agent 有學到一些正確的策略。

在訓練完成後，將訓練好的 Agent 對 baseline 1-5 進行測試 20 局，對上各個 Agent 的勝率分別為 100%、80%、95%、60%、35%。效果普遍合乎預期針，唯有對上 baseline 5 時會有太積極的下注，導致最後勝率未達標準。



2. Monte Carlo Simulation

在這個方法中，我先跑了 20000 次模擬預估了共 2652 種 hole cards 在 preflop 時的勝率，並將其預先存起來。得出的勝率分布如右圖所示。

在 preflop 時直接讀檔以節省時間，並採取相對 DQN 比較保守的做法，在 preflop 時勝率若小於 50.33%時就會 fold，其他策略則可以大致分為以下幾種:

(1) 建立一個 Win Table，若每一輪的金額超過對應能贏的金額，則在後續幾輪都 fold 以確保最終勝利。

(2) 建立一個 Lose Table，如果採取 fold action 會讓自己低於對應的金額，則根據預估出來的勝率 call 或是 raise，防止因為 fold 而最後輸。

(3) 如果是最後一輪且勝負未定，則 all in 來最大化所能獲得的金額。

(4) 在 preflop 時，不論勝率有 50%的機率會 raise 25 來騙對手且不會總下注超過 100。

(5) 在 flop/turn/river 時如果勝率超過 80%且正落後，則有 20%/40%/80%的機率會 all in 或 raise 到能超過 Win Table 的金額。

(6) 如果在開牌後每輪勝率低於 40%且不會導致輸牌的前提下 fold。

在加上一些其他小策略後，對上 baseline 1-5 同樣地測試 20 局分別能獲得 90%、75%、90%、40%、55%的勝率。雖然對上 baseline 5 勝率有所提升，但在相對強勢的 baseline 2 和 4 中勝率有降低的趨勢。

三、問題討論

在觀察我的 Agent 進行訓練的時候，發現 baseline3 有時候會 raise 不能 raise 的金

額，導致最後明明要 raise 被系統判定為 fold。另外也發現在德州撲克的一般規則中如果 raise 之後下一次 re-raise 應該不管金額都可以有 all in 的選項，但是這份 code 好像如果 all in 的金額低於 reraise 的最低金額就不會有這個選項，這個機制可能會使一些策略變成 fold，值得再去進一步檢視。

四、總結

綜合上面兩種 Agent 的特色，我最後決定將兩者的優點保留，以 DQN Agent 為主軸加上 Monte Agent 的一些策略作為最終的 Agent，同時保留 DQN 向不同 Agent 訓練所得到的激進策略又同時建立 Win Table 和 Lose Table 和一些 Monte Player 的策略以防 Agent 做出不合理的舉動 (如: 在確定或快要贏的時候沒有注意到而下注更大、在第一輪先下手對方還沒有動靜就 raise 過大的金額等)。另外，除了和上述的 baseline 進行對抗，我也請了競技撲克社的一些社員和我的 Agent 進行對弈。並參考他所指給的建議進行策略的修正與改進。

這些加入的策略就像是最後一道防線在保護 Agent 能夠在對的時間不要做出錯的舉動。最終這個 Agent 對上 5 個 baseline 20 局能有 95%、90%、100%、60%、50%左右的勝率，總體而言比兩個 Agent 各自的表現還要來的更好。我也請了競技撲克社競賽季軍和我的最終版 Agent 比五戰三勝，戰況十分膠著直到第五局我的 Agent 才勉強拿下勝利。總而言之，德州撲克還是以個運氣成分偏多的遊戲，我們能做的也就只有卻保自己的 Agent 能夠戰勝機率，並獲得最終勝利。

五、參考資料

- https://medium.com/pyladies-taiwan/reinforcement-learning-%E9%80%B2%E9%9A%8E%E7%AF%87-deep-q-learning-26b10935a745
- https://ithelp.ithome.com.tw/articles/10208668
- https://github.com/ishikota/PyPokerEngine
- https://github.com/dickreuter/tf_rl
- https://github.com/willwhite99/PokerAI
- https://github.com/offthewallace/poker-DQN