FAI Final Project Report

姓名: 鐘文駿

學號: B11902167

Methods

Method 1

這個方法不管在哪個階段，都會有以下通則:

1. 如果這把 fold 會導致對手得到 pot 後，後面對面選擇全部 fold 就會贏，那就要避免 fold。按照手牌選擇要 call 還是 all in。
2. 如果在某個 street 選擇 all in，後面都要選擇 check (call 0)

- **Preflop:** 根據下方圖表決定策略

	A	K	Q	J	T	9	8	7	6	5	4	3	2
A	85%	68%	67%	66%	66%	64%	63%	63%	62%	62%	61%	60%	59%
K	66%	83%	64%	64%	63%	61%	60%	59%	58%	58%	57%	56%	55%
Q	65%	62%	80%	61%	61%	59%	58%	56%	55%	55%	54%	53%	52%
J	65%	62%	59%	78%	59%	57%	56%	54%	53%	52%	51%	50%	50%
T	64%	61%	59%	57%	75%	56%	54%	53%	51%	49%	49%	48%	47%
9	62%	59%	57%	55%	53%	72%	53%	51%	50%	48%	46%	46%	45%
8	61%	58%	55%	53%	52%	50%	69%	50%	49%	47%	45%	43%	43%
7	60%	57%	54%	52%	50%	48%	47%	67%	48%	46%	45%	43%	41%
6	59%	56%	53%	50%	48%	47%	46%	45%	64%	46%	44%	42%	40%
5	60%	55%	52%	49%	47%	45%	44%	43%	43%	61%	44%	43%	41%
4	59%	54%	51%	48%	46%	43%	42%	41%	41%	41%	58%	42%	40%
3	58%	54%	50%	48%	45%	43%	40%	39%	39%	39%	38%	55%	39%
2	57%	53%	49%	47%	44%	42%	40%	37%	37%	37%	36%	35%	51%

我的策略會分成以下三種情況 (不管位置):

- 如果表格中的勝率大於等於 65%，會直接 raise 5 個大盲
- 如果勝率界在 55% ~ 65% 之間，會選擇 call
- 如果勝率在 55% 以下，會選擇 fold

例外: 如果接下來的 round 全部 fold 都能贏，就選擇fold

- **Postflop (Flop, River and Turn):** 使用蒙地卡羅取樣本算目前手牌的勝率，根據勝率決定這個 street 的決策。取樣本的方法如下:

  1. 從有 52 張牌的牌庫中，先把自己的 hole cards 和 community cards 取出來
  2. 從剩下的牌中，取出 2 張當作對手的模擬 hole cards，以及剩下未被翻開的樟樹的 community cards (0 ~ 2張)。
  3. 用 eval_hand() 函式去檢查自己和對方的手牌牌型，並比較哪方模擬出來的結果會獲勝
  4. 重複步驟 1 ~ 3，一次決定模擬 500 次並算出自己的勝率。

  算出模擬的勝率後，利用以下規則去決定自己在 postflop 的決策，分成以下三種:

  1. 如果勝率大於等於 60%，選擇 all in
  2. 如果勝率界在 60% 和 50% 之間，選擇 raise 5 個大盲
  3. 如果勝率在 50% 以下，要注意這裡如果隨便 fold 可能會失去免費看後面的牌的機會，所以要去檢查我是否是先手或對方是否也是 check，這時選擇 check (call 0)

Method 2

使用 Supervised Learning 訓練出一個策略模型，模仿 baseline7 的決策邏輯。流程如下：

- **蒐集訓練資料:** 由於無法查看及修改 baseline7 的 code，所以我用原本 method 1 的方法去觀察及記錄以下資訊:

  1. 由於無法直接得到對手的手牌細節，因此只能記錄對方手牌的 strength 和 rank (使用 `receive_round_result_message()`)
  2. 當前的 community cards
  3. 該局的行為
  4. 對方目前的 stack 和 pot 的大小

- **轉換成 Supervised Training Data**: 把蒐集的資料轉換成可以使用的形式:
  - **Label:** Baseline 7 採取的行為 (call, raise or fold)
  - **Features**
    1. hole cards 的 strength 和 rank
    2. community cards
    3. street

    4. baseline 7 與對方的 stack
    5. pot
    6. 當前 street 的下注順序
- **Model Training**: 使用 MLP，訓練一個 Multi-class classifier。輸入一筆遊戲狀態特徵後，輸出最可能的動作
- **額外策略**: 因為原本的 baseline7 可能也會做出反直覺的舉動，因此額外加上了一些 rule 來當作最後決策:
    - 如果接下來的 round 全部 fold 都能贏，就選擇fold
    - 如果這把 fold 會導致對手得到 pot 後，後面對面選擇全部 fold 就會贏，那就要避免 fold。按照手牌選擇要 call 還是 all in。
    - 避免在籌碼比對面多 200 以上時選擇 all in

Comparasion

以下是兩種不同的 method，對每個不同的 baseline 各打 100 場的勝率。可以得出整體表現 method 1 比 method 2 表現得更好，尤其在比較困難的 baseline 更為明顯。

Method 1

- baseline 0: 78%
- baseline 1: 78%
- baseline 2: 81%
- baseline 3: 75%
- baseline 4: 59%
- baseline 5: 44%
- baseline 6: 46%
- baseline 7: 52%

Method 2

- baseline 0: 56%
- baseline 1: 61%
- baseline 2: 40%
- baseline 3: 49%
- baseline 4: 36%
- baseline 5: 21%

- baseline 6: 11%
- baseline 7: 20%

Discussion and Conclusion

整體來說 method 1 的表現比 method 2 好很多，以下是我認為的原因:

- Method 1 是參照 GTO 的策略去實作，雖然簡單又暴力，但直接使用最高機率的打法式非常有效。這裡**不使用最大期望值**打法的原因是: 我認為總場數非常少，包含一局 20 round 和總共的 5 局，1 round 輸掉少數的 stack 就容易輸掉一整局。所以每局都採取最大機率打法，只要某一 round 贏了一些 stack，就可以讓我領先很多，而不是去賭低機率但期望值高的 round。
- Method 1 比較好去調整實作細節，可以先自己去觀察與各個 baseline 的對局後再去想有甚麼可以改進的地方 (比如沒注意就很容易不小心錯過免費的 check)。在這方面 method 2 就很容易有問題，因為我是拿 baseline 7 的資料去訓練，我也無法很細部得知其實做細節，因此很難去做調整，只能做一些防呆機制 (例如檢查免費 check)。
- Machine Learning 的方法還需要大量的資料，但很顯然我們沒有辦法生出幾百萬筆資料去 train model，而導致很嚴重的 underfitting。但用同一個 baseline 去 train 又很容易導致 overfitting，而導致無法與其他 baseline 抗衡。

所以就結果來說，rule-based 的 AI 在這裡表現的不差，所以我最後選擇用 method 1。