FAI Final Project Report

B10202064 傅學惟

1 Model configuration and Trials

1.1 Win rate estimation

During each street, I random assign cards to our opponent and the remaining community cards then compare the result. By sampling severals configurations, the agent obtains an approximation $\hat{p}$ to the win rate, then the agent can perform actions based on the estimated win rate.

1.2 Threshold Agent

The naive idea is, if we may choose to perform "fold", "call", or "raise" based on some threshold. For example, if $\hat{p} < 0.3$, fold, $0.3 < \hat{p} \leq 0.7$, call, $0.3 \leq \hat{p} \leq 0.7$, raise. With some tuning, I find a emperical threshold value:

$$\begin{cases} \text{fold, if } \hat{p} < 0.35 \\ \text{call, if } 0.35 \leq \hat{p} < 0.55 \\ \text{raise, if } 0.55 \leq \hat{p} \end{cases}$$

1.3 Heuristic Raise Amount

I define the heuristic raise formula as follow.

$$\max\left((\hat{p} - 0.5) \times RaiseAmount[max] \times \frac{1}{6}\left(1 + len(CommunityCard)\right), RaiseAmount[min]\right)$$

So that the acceptable raise amount is linearly scaled up with the win rate, and the number of the community cards, since we will be more certain about our action with more community cards.

1.4 Expectation Agent

The previous agent has a problem that it decide wether to call or raise purely based on the winning rate and didn't take the cost into account. For example, if the opponent raise a high amount of stack (e.g. 500), our agent will raise with winning rate only 0.6. Therefore, we

consider another decision rule, driven by the expectation stack after the action. To be specific, suppose the estimated winning rate is $\hat{p}$, compare the following expectation values

$$\begin{aligned}
\mathbb{E}\left[Stack|Fold\right] &= Stack \\
\mathbb{E}\left[Stack|Call\right] &= Stack + \hat{p} \cdot pot - (1-\hat{p}) \cdot CallCost \\
\mathbb{E}\left[Stack|Raise\right] &= Stack + \hat{p} \cdot (pot + RaiseCost) - (1-\hat{p}) \cdot RaiseCost.
\end{aligned}$$

Note that we assume the opponent call for the raise (except for the all in case, which is handled in the code), and we use the same heuristic raise amount formula to calculate the determine the raise amount. We then choose the action with maximum expectation.

1.5 Cautious Agents (Threshold2, Expectation2

I mentioned that a linear scale multiplier is used to make raise amount less when community cards are not fully reveiled. The strategy is used to avoid from the model being overconfident too early. Since this linear scaling is purely heuristic, I consider another strategy: consider the deviation of the wining rate in the river street to all the previous streets. To be specific, I take 1000 simulations of the game, then take the average of the value

$$\mathbb{E}\left[\left|\hat{p}(4) - \hat{p}(i)\right|\right], \forall i \in \{0,1,2,3\}$$

where $\hat{p}(i)$ denote the approximated wining rate at the i-th street, we treat this average deviation as the "information remain unreveiled" at i-th street. Then, I further analyze the probability of deviation outside a interval, i.e. $\Pr[\left|\hat{p}(4) - \hat{p}(i)\right| \geq \epsilon_i]$ to find the 25% confidence interval and 40% confidence interval. Note that if we assume the distribution is symmetric, we can bound the probability of overoptimistic $\Pr[\hat{p}(i) - \hat{p}(4) \geq \epsilon_{i,0.25}]$, $(\Pr[\hat{p}(i) - \hat{p}(4) \geq \epsilon_{i,0.3}])$ within the probability 0.375 and 0.2. The result of avgerage deviation is listed in

Confidence Level	**Preflop**	**Flop**	**Turn**	**River**
25%	0.1162	0.0481	0.0228	0
40%	0.1883	0.0999	0.0476	0

Table 1: Confidence interval parameters ϵ at different levels over four streets.

We then define the safe rate as $\hat{p}_{\epsilon_i} := \hat{p} - \epsilon_i$. In all "cautious" variants (of the threshold agent and the expectation agent), we will use this $\hat{p}_\epsilon$ to replace $\hat{p}$. Also, we use

$$\max\left((\hat{p}_{\epsilon_i} - 0.5) \times RaiseAmount[max] \times (1-\epsilon_i), RaiseAmount[min]\right)$$

to replace the original raise amount formula. All the other part of the code remain the same as the threshold agent and the expectation agent.

1.6 Folding Trick

For a 2p game, as long as we match the below criterion, we may fold safely to achieve win without taking any additional risk.

$$stack - InitialStack > \frac{1}{2} \times RemainRounds \times (bigBlind + smallBlind) + BigBlind,$$

All agents discussed use this strategy to boost the win rate.

1.7 Additional Tries - Hybrid Method and Aggressiveness Tracking

In section 2, we know that expectation agent, even the cautious version, prompt to call, or even raise when the pot is large, even with low win rate (safe rate). However, if opponent make the pot big, they are often very confident about there hand, therefore, pure expectation optimization become very risky, and therefore I use a hybrid setup to adjust the cautious expectation agent. The pseudo code is in below. The

Algorithm 1 declare_action (simplified)

Use $\hat{p}_{\epsilon_{0.40}}$ if opponent is aggressive (cost $> 80\%$ of other action, or all-in); else use $\hat{p}$
if `safe_rate` < 0.4 or (preflop and min_raise $> 10\times$ small blind) **then**
    No raise
else if `safe_rate` < 0.6 **then**
    raise $\leftarrow$ safe_margin $\times$ max_raise $\times$ (1 $-$ uncertainty) (or $5\times$ small blind if preflop)
else
    raise $\leftarrow$ pot / 4
end if
Clamp raise to [min_raise, max_raise]
if opponent all-in and safe_rate > 0.5 **then**
    Consider all-in
end if
Compute expected values; return action with max expectation

2 Discussion and conclusion

2.1 Result and Baseline Analysis

The win rate results are shown in table 2. First, we observe that while the threshold agent and its variants perform well against baseline1 to baseline3, their effectiveness significantly drops against baseline4 to baseline7. This may be because the threshold agent does not account for contextual factors such as pot size or the opponent's actions. As a result, it becomes vulnerable when facing more adaptive or strategic opponents.

In contrast, the expectation-based agent consistently outperforms others, aligning with what probability theory would suggest. However, its conservative variants yield mixed results. Against aggressive opponents like baseline4, the added caution has minimal benefit. But when playing against more passive agents such as baseline5 and baseline6, this conservative behavior can become a liability, leading to reduced performance due to missed opportunities, but in general, the conservative variants are more stable compare to the expectation agent.

2.2 Acting Order Analysis

Another interesting observation is the asymmetry in win rates when switching the play order between our agent and the baselines. Specifically, acting second) tends to be more advantageous against conservative opponents like baseline5 and baseline6, while acting first is often better against aggressive opponents. This may be because when facing conservative agents, the second player gains more information before committing to an action, allowing for better adaptation. On the other hand, against aggressive agents, acting first can help apply pressure and avoid being intimidated by their large bets, reducing the risk of folding prematurely.

Agent/Against	baseline1	baseline2	baseline3	baseline4	baseline5	baseline6	baseline7
Threshold	(1, 1)	(.95, .8)	(.8, .8)	(.1, .15)	(.4, .325)	(.25, .3)	(.1, .25)
Expectation	(.95, 1)	(1, 1)	(1, .95)	(.5, .55)	(.75, .55)	(1, .35)	(.4, .55)
Threshold2	(.95, .95)	(.9, .7)	(.775, .7)	(.15, .2)	(.45, .3)	(.85, .3)	(.15, .05)
Expectation2	(1, 1)	(1, .95)	(1, 1)	(.5, .5)	(.8, .75)	(1, .35)	(.2, .4)
Hybrid	(.90, .95)	(.95, .90)	(1, 1)	(.55, .60)	(.50, .50)	(.30, .45)	(.40, .25)

Table 2: Win rates per agent for positions after (first) and before (second) opponent, across 8 match settings, with 20 samples for each results.

2.3 Conclusion

While these variants are not yet optimal, they suggest that tailoring responses to opponent style can be valuable. Future improvements may involve more sophisticated opponent modeling, including hand reading (with aggressiveness detection being a basic form), action planning via Monte Carlo Tree Search, or the application of learning-based strategies to dynamically adjust decisions.

3 Final Submission

Altough expectation agent is strong, I submit the Hybrid agent, which is not the strongest, but the most stable agent compare to other agent.