Report

Group member: Jinze Wan (101168891), Jiale Han (101149011), Billy Chen (101108885)

Jinze Wan: hard(),soft(), compute_total(), report

Jiale Han: get_shuffled_deck(), deal_card(), print_hand(), demonstration

Billy Chen: black_jack(), test code

Introduction

Blackjack is a classic gambling game known as the most likely event for players to make money. In blackjack, the player (AI) maximizes his financial benefit by taking actions that yield the greatest expected rewards with limited knowledge of the environment.

The rule of blackjack is to take 52 cards as A deck, count J,Q,K as 10, A as 1 or 11, to get A total of points as close to but not more than 21 points. At first the banker hands out two cards of I to himself and to each player, with one of the banker's cards face down, meaning that the player only knows the number of points on one of the dealer's cards. Players can choose to hit, stand, or double if they have not yet asked for an extra card. After player’s action finished, banker can show his all card, decide whether to want card with must be greater than or equal to 17 fixed rules next, once be greater than or equal to 17 points, no matter whether player exceeds banker point, cannot want card. No matter player or banker, once the point exceeds 21, automatically judge lose.

As with any gambling game, the rules favor the banker, so the AI's strategy of action is crucial in order to achieve positive returns.

Methods

In this project, AI is a goal based agent. The agent will take actions based on the current environment, and the agent's actions will also have an impact on the environment, and the AI will take actions with the goal of maximizing financial benefits.

Since the rules favor the house, the AI needs to use the double rule to maximize returns when circumstances are favorable to it. The following table summarizes the actions the AI takes to maximize its own profits without knowing all the information about the bookmaker.

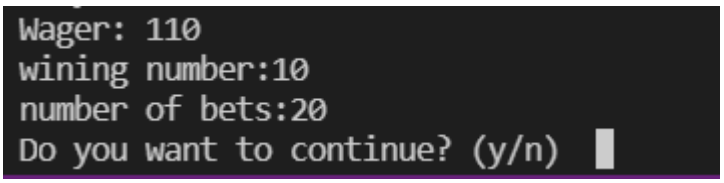
Player hand	Dealer's face-up card									
	2	3	4	5	6	7	8	9	10	A
Hard totals (excluding pairs)										
18–21	S	S	S	S	S	S	S	S	S	S
17	S	S	S	S	S	S	S	S	S	Us
16	S	S	S	S	S	H	H	Uh	Uh	Uh
15	S	S	S	S	S	H	H	H	Uh	Uh
13–14	S	S	S	S	S	H	H	H	H	H
12	H	H	S	S	S	H	H	H	H	H
11	Dh	Dh	Dh	Dh	Dh	Dh	Dh	Dh	Dh	Dh
10	Dh	Dh	Dh	Dh	Dh	Dh	Dh	Dh	H	H
9	H	Dh	Dh	Dh	Dh	H	H	H	H	H
5–8	H	H	H	H	H	H	H	H	H	H
Soft totals										
	2	3	4	5	6	7	8	9	10	A
A,9	S	S	S	S	S	S	S	S	S	S
A,8	S	S	S	S	Ds	S	S	S	S	S
A,7	Ds	Ds	Ds	Ds	Ds	S	S	H	H	H
A,6	H	Dh	Dh	Dh	Dh	H	H	H	H	H
A,4–A,5	H	H	Dh	Dh	Dh	H	H	H	H	H
A,2–A,3	H	H	H	Dh	Dh	H	H	H	H	H

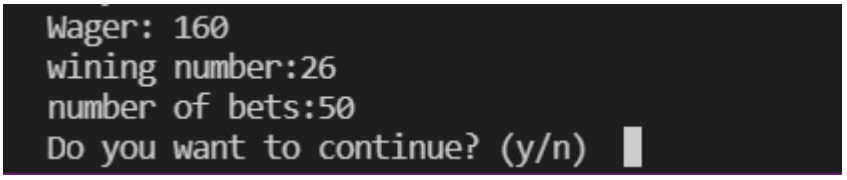
H = Hit
S = stand
D = double
U = surrender

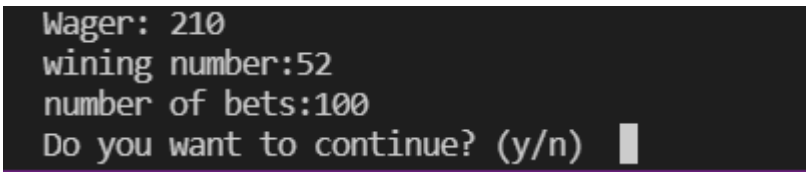
AI will act based on whether there is an A in its current hand, if there is no A in hand, use hard policy, if there is an A in hand, use soft policy. Because the AI does not know the number of cards the banker covers, the AI can only take action based on probability. And given the randomness of the deal, the AI can still lose even if it takes the action with the highest expected reward. However, when using this strategy, the AI can still get close to a 49% win rate, and with the double rule, there is still enough probability of a positive financial return in enough cases.

It can be seen that AI tends to stand when its current hand total is high, unless the dealer's name is high. Because in this case the AI has a high enough probability of getting a higher total than the opponent and can avoid exceeding 21 total points.On the other hand, if the current hand total is low enough, the AI will take the card, because it is highly unlikely that the total will be lower than the banker and over 21. When the total is low enough and at an ideal interval, the AI will also be particularly inclined to double, as there is a high chance of scoring high points without going beyond 21.

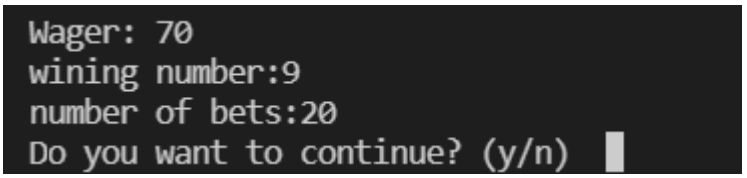
Result

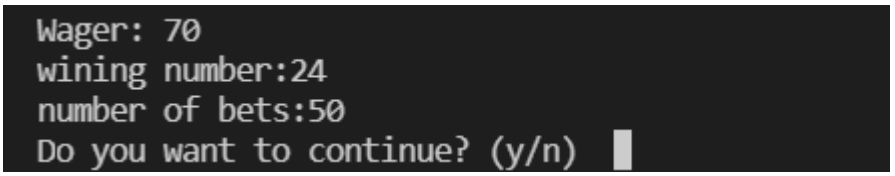


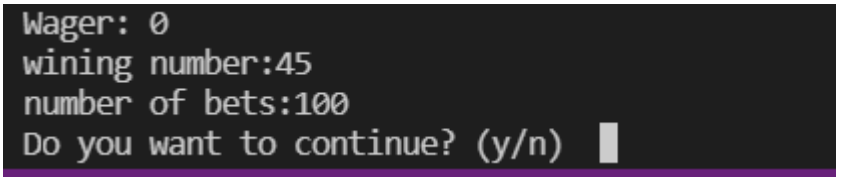




At the beginning, the AI had 100 Wagers. As we can see, when the number of bets is high enough, the AI gets a positive reward, even though it doesn't significantly exceed 50%. This is because the AI takes as many double actions as possible in situations where the win price is high, maximizing its profit.







The above is the result of another test, and we can see that the AI still got a 45% win rate, but ended up with 0 wager. This is because of the uncertainty of gambling games and, most importantly, because ai doesn't know all factors about the

environment. Banker has a face down card. In this case, it is possible that the banker has a very favorable hand, AI is still according to the banker’s face-up card to take double action, the result of the double action of AI will increase their losses.

In other tests, both of these scenarios occurred frequently, with the AI's win rate stabilizing between 45% and 52%, and sometimes exceeding, reflecting the uncertainty of gambling games. Still, that's a better chance of winning than relying on human intuition or the popular blackjack simplification strategy. Both of these situations tend to run out of wager faster on a bad day.

Discussion

In this project, we tried to reconstruct the action of a gambler as close as possible to the situation in life. The rules favor the banker, the deal is uncertain, and the gambler does not know the full extent of the other hand. This is a good case for designing AI.

The result shown successfully recreates a gambler who used his intelligence (and, inevitably, luck) to make money. In situations where the information is not known, gamblers will carefully place bets based on probability, and AI restores this behavior and actively exploits it. (And, if AI is unlucky, it can amplify the loss.) In a large number of tests, the AI's win rate also approached 49%, indicating that the AI design was basically successful.

We also noticed that sometimes the AI lost its bets in a lot of code testing, indicating that the AI strategy we were currently designing was flawed.

However, in this work, we realized that using reinforcement learning to design AI is another good approach. If the AI is designed using this pattern, it can constantly improve its actions based on past situations. This is something our current AI lacks. In addition, if we add a feature that runs AI surrender, the AI can reduce its losses by taking half the bet back if the situation goes against it.

In the card game Agent, we confirm that the model applied in this project can also be tried in Texas Holdem. If we design similar AI again in the future, we might try this game.