Optimal Blackjack AI

COMP 3106

Group 83

Jake Jazokas - 101083496

Akinbile Ayodeji - 101063483

Oluwademilade Edward Akapo - 101095403

Statement of contributions

Each member performed an approximately equal contribution towards the project. All members contributed towards the final report and the demonstration. For code implementation, Jake and Akinbile worked on the Basic AI as well as the Client and Server Side Game Logic and Edward worked on implementing the Model-based AI.

Introduction

Problem Motivation

It is difficult to always know the optimal move when playing Blackjack. Combined with the added difficulty of card counting, making the optimal move for any given situation can be a difficult calculation even for the most seasoned player. Our goal is to create an artificial intelligence agent to play blackjack optimally. The agent consistently makes the most optimal moves, giving it a higher probability to win than your average player. This will also result in our agent winning more money than an average Blackjack player.

Problem Statement

As it is difficult to constantly make optimal moves in the game of blackjack, we set out to test multiple implementations of AI to see if we can come up with one that has a higher expected value of returns after a series of games, than your average player.

The blackjack variation we made has 4 active players, consisting of 2 networked, live, humans, and 2 AI implementations. As well as a dealer. These 4 players are competing against each other and the dealer and they only have 2 possible moves for each turn, Hit or Stand. This variant is similar to your standard game but in this version, each player gets one hidden card at the beginning. The game is played in rounds, with the humans acting first, then the AI's and finally the dealer.

Related Work

For our project, we took a lot of inspiration from a similar project done by Joshua Greiser and Tristan Hasseler from Stanford University [2]. We used their idea for state parameterization and a blackjack strategy chart to come up with our model environment.

Methods

Methods from Artificial Intelligence

Simple Reflex Agent

This agent checks the visible hands of all players, then checks some preset if-then rules to determine its next move. Its next move is determined by if another player is likely to have a better hand than the AI because in that case, we need to get as close to 21 to win. If the AI has a relevantly strong hand (18-21), it will stand most of the time, unless it thinks another player has 21.

Model-Based Agent

To simplify the possible combinations of states and action pairs, the blackjack game only allows two possible actions per player, Hit (draw another card) or Stand(do not draw any more cards). Moreover, all players must bet a fixed amount of 25 $. Although casino blackjack typically includes more options such as variable betting, double down, surrender… Etc, reducing the action space to these two options drastically simplifies the model complexity.

Figure 1 below shows the state parameterization for our environment. States 3 - 182 represent the active Ai player states against an opponent. To determine the action for each state the agent observes the state of the game, it will look at its hands and the visible hands of all players and use a series of state-action pairs to confirm its next move.

		Opponent players Upcard									
		A	2	3	4	5	6	7	8	9	10
AI players Hand	4	3	21	39	57	75	93	111	129	147	165
	5	4	22	40	58	76	94	112	130	148	166
	6	5	23	41	59	77	95	113	131	149	167
	7	6	24	42	60	78	96	114	132	150	168
	8	7	25	43	61	79	97	115	133	151	169
	9	8	26	44	62	80	98	116	134	152	170
	10	9	27	45	63	81	99	117	135	153	171
	11	10	28	46	64	82	100	118	136	154	172
	12	11	29	47	65	83	101	119	137	155	173
	13	12	30	48	66	84	102	120	138	156	174
	14	13	31	49	67	85	103	121	139	157	175
	15	14	32	50	68	86	104	122	140	158	176
	16	15	33	51	69	87	105	123	141	159	177
	17	16	34	52	70	88	106	124	142	160	178
	18	17	35	53	71	89	107	125	143	161	179
	19	18	36	54	72	90	108	126	144	162	180
	20	19	37	55	73	91	109	127	145	163	181
	21	20	38	56	74	92	110	128	146	164	182

Fig.1 State Parameterization

To come up with the optimal moves for each state in figure 1 we used a reference strategy chart shown in Figure 2 below. We adjusted the strategy to fit our specific variation of blackjack, which has all players facing each other and the Dealer.

Reference Strategy

HARD TOTALS

	DEALER UPCARD									
	2	3	4	5	6	7	8	9	10	A
17	S	S	S	S	S	S	S	S	S	S
16	S	S	S	S	S	H	H	H	H	H
15	S	S	S	S	S	H	H	H	H	H
14	S	S	S	S	S	H	H	H	H	H
13	S	S	S	S	S	H	H	H	H	H
12	H	H	S	S	S	H	H	H	H	H
11	D	D	D	D	D	D	D	D	D	D
10	D	D	D	D	D	D	D	D	H	H
9	H	D	D	D	D	H	H	H	H	H
8	H	H	H	H	H	H	H	H	H	H

SOFT TOTALS

	DEALER UPCARD									
	2	3	4	5	6	7	8	9	10	A
A9	S	S	S	S	S	S	S	S	S	S
A8	S	S	S	S	Ds	S	S	S	S	S
A7	Ds	Ds	Ds	Ds	Ds	S	S	H	H	H
A6	H	D	D	D	D	H	H	H	H	H
A5	H	H	D	D	D	H	H	H	H	H
A4	H	H	D	D	D	H	H	H	H	H
A3	H	H	H	D	D	H	H	H	H	H
A2	H	H	H	D	D	H	H	H	H	H

Key	
H	Hit
S	Stand
D	Double if allowed, otherwise hit
Ds	Double if allowed, otherwise stand

Fig.2 Reference Strategy[2]

Thus the result gives us Figure 3 below, which has the optimal move for most of the possible environment states encountered by the AI. Given that we have 4 opponent players, we implemented the decision-making algorithm of the AI to use this model to judge the best move to play by judging every player's hands independently this will give it a sequence of Hits and Stands after it has looked at all opponents and dealers hands, then it will pick the action by looking at the most frequently occurring action in the sequence (If it's a tie between hit and stand it will hit).

AI players Hand	Opponent players Upcard									
	A	2	3	4	5	6	7	8	9	10
4	H	H	H	H	H	H	H	H	H	H
5	H	H	H	H	H	H	H	H	H	H
6	H	H	H	H	H	H	H	H	H	H
7	H	H	H	H	H	H	H	H	H	H
8	H	H	H	H	H	H	H	H	H	H
9	H	H	H	H	H	H	H	H	H	H
10	H	H	H	H	H	H	H	H	H	H
11	H	H	H	H	H	H	H	H	H	H
12	H	H	S	S	H	S	H	H	H	H
13	H	H	S	S	S	S	H	H	H	H
14	H	S	S	S	S	S	H	H	H	H
15	H	S	S	S	S	S	H	H	H	H
16	H	S	S	S	S	S	H	H	H	H
17	S	S	S	S	S	S	S	S	S	S
18	S	S	S	S	S	S	S	S	S	S
19	S	S	S	S	S	S	S	S	S	S
20	S	S	S	S	S	S	S	S	S	S
21	S	S	S	S	S	S	S	S	S	S

Fig.3 Model used for AI

On the other hand, we have cases for which the agent can't use the environment model to accurately judge its next move(such as when a player has a total score of Upcards greater than 11), thus we introduced a probability function and a variable called diff = (21 - Ai's score). The function has the AI only hit when the probability of it getting a card lower than or equal to the diff from the deck, given the cards the AI can see that have already been taken from the deck is greater than 0.50.

Validation Experiments

In order to validate that our implementations are performing more optimally than an average player, we plan to use ourselves for validation against the results from the AI's. We can do this by comparing the total winnings of the AI versus ourselves over a series of 25 and 50 games. We can also have the players make random moves to test that the AI has a better than random chance of winning.

Table 1 Win rate after 20,000 games for each policy

Policy	State Mapping 1 (agent's hand)	State Mapping 2 (agent's hand + dealer's upcard)
Random Policy	28%	28%
Value Iteration	41.2%	42.4%
Sarsa	41.9%	42.5%
Q-Learning	41.4%	42.5%

Fig.4 The AI win rate from [2]

Based on the expected win rate found in the paper [2], we can assume that our AI implementation will have similar results. Thus, we can base part of our validation of our AI on how close the results are to the ones in Fig.4.

Results

Qualitative Results

50 Games vs Humans

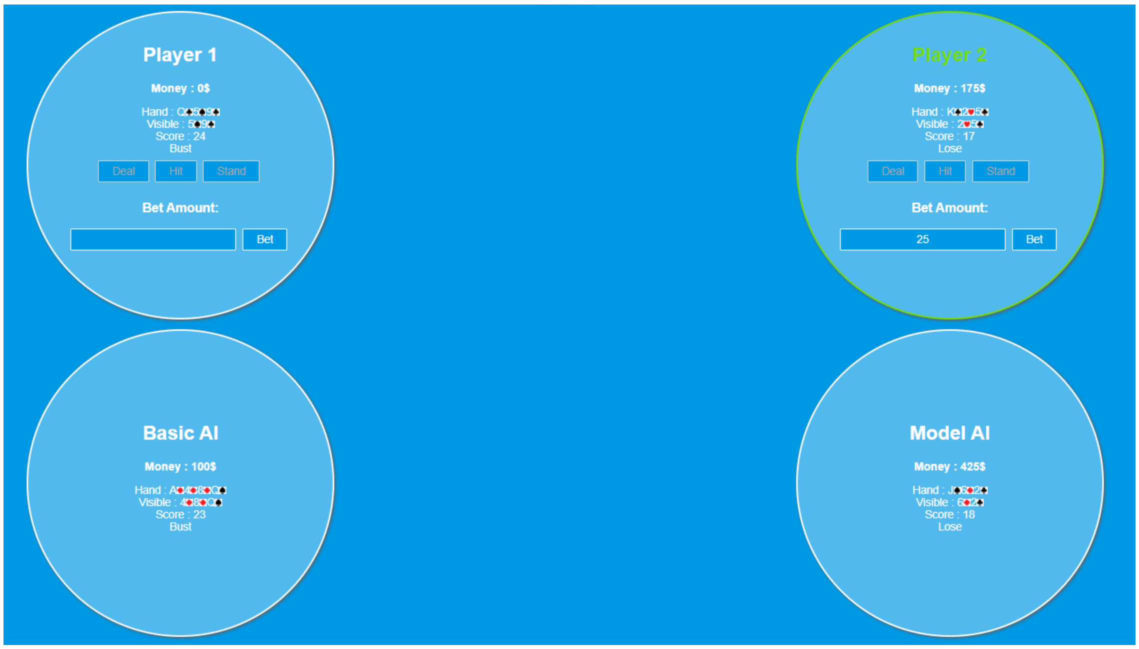

Fig.5 The results from playing 50 rounds with 2 human players

	Wins / Losses	Win % (50 Rounds)	+/- Money
Player 1	10 Wins / 40 Losses	20%	-1000
Player 2	17 Wins / 33 Losses	34%	-825
Player Average		*27%*	*-912.5*
Basic State-Based AI	14 Wins / 36 Losses	28%	-900
Model-Based AI	23 Wins / 27 Losses	46%	-575

25 Games vs Different Strategies

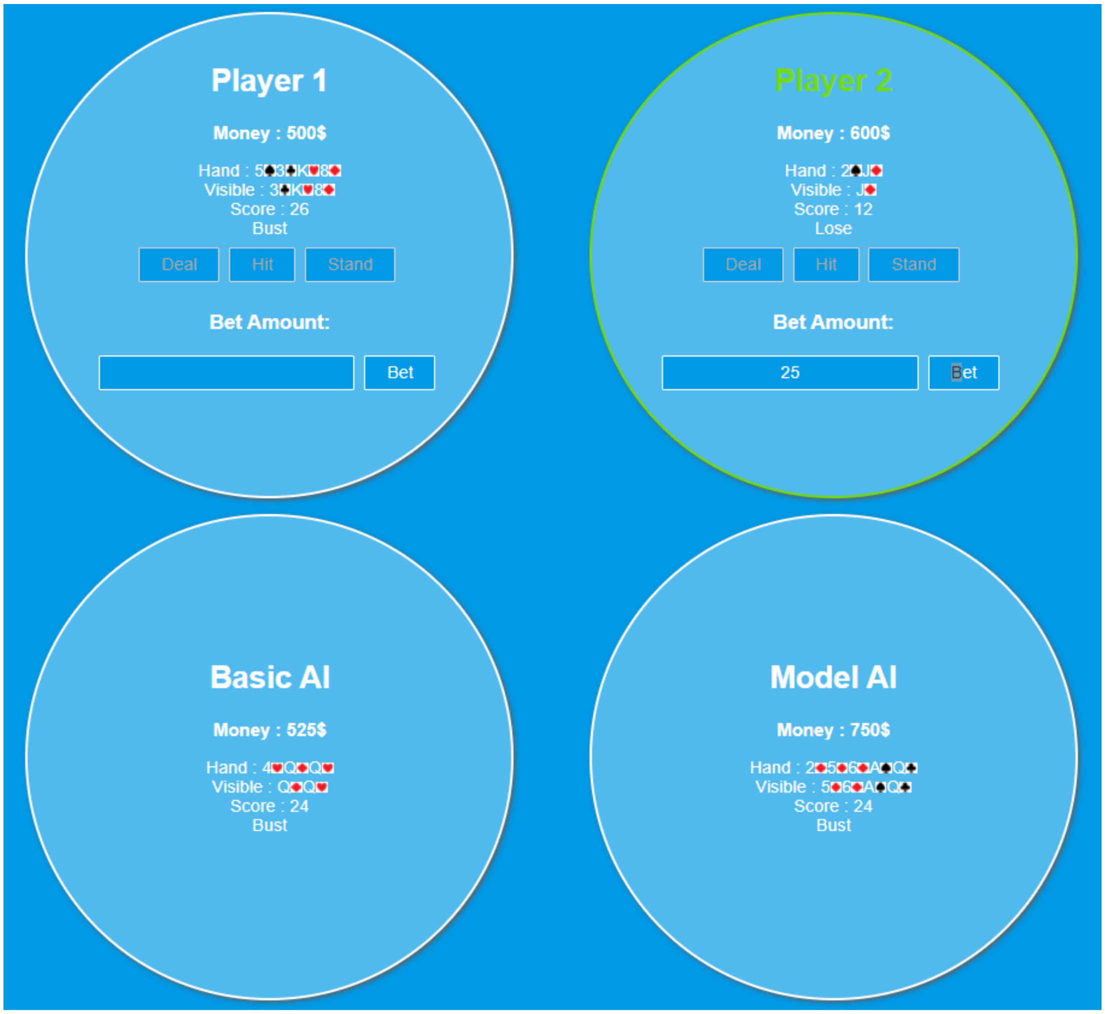

Fig.6 The results from playing 25 rounds both players on a fixed action

P1 Hits (Unless 21), P2 Stands	**Wins / Losses**	**Win % (25 Rounds)**	**+/- Money**
Player 1	5 Wins / 20 Losses	20%	-500
Player 2	9 Wins / 16 Losses	36%	-400
Player Average		*28%*	*-450*
Basic State-Based AI	6 Wins / 19 Losses	24%	-475
Model-Based AI	15 Wins / 10 Losses	60%	-250

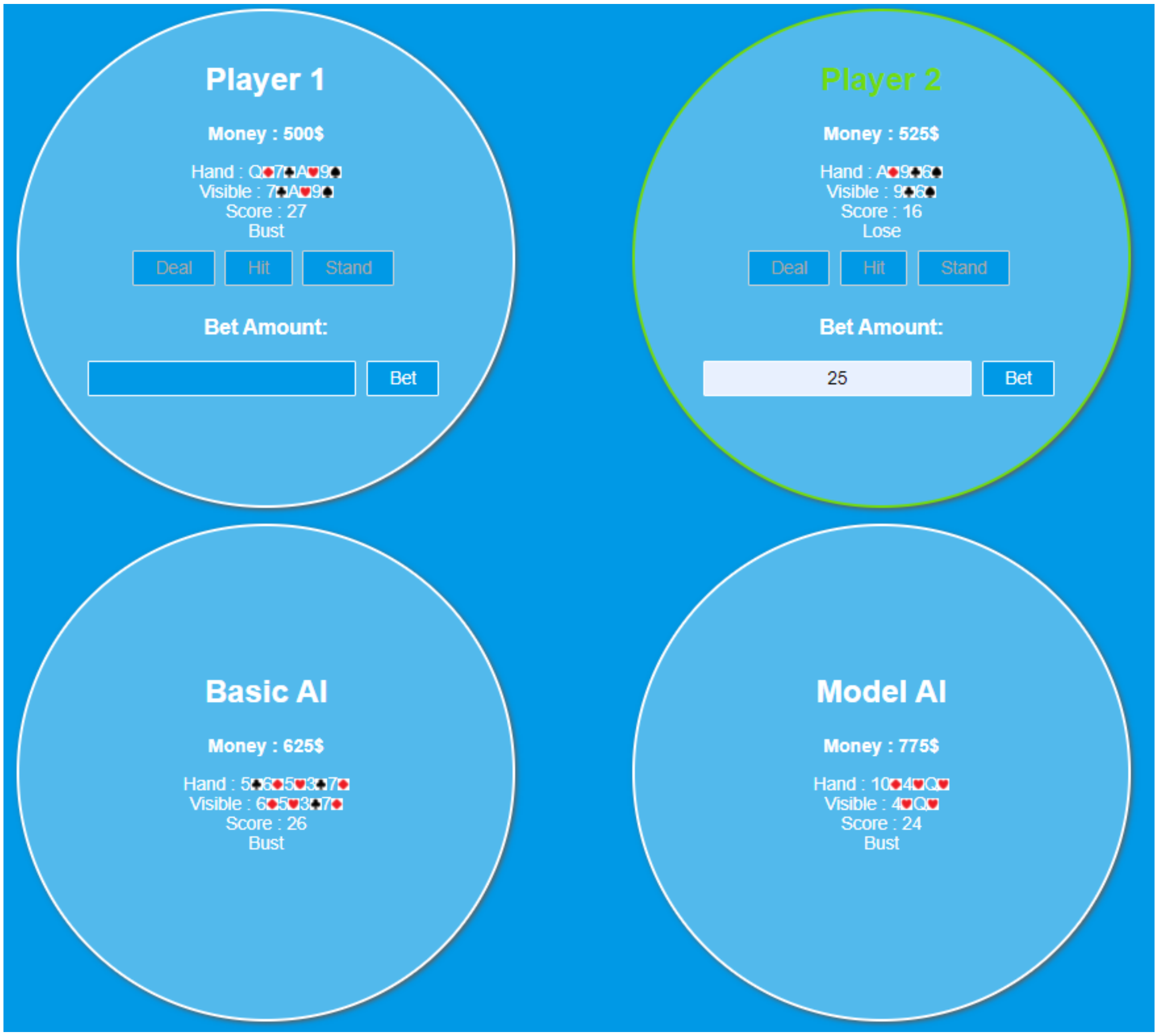

Fig.7 The results from playing 25 rounds both players using a random action

Random Action (Player)	**Wins / Losses**	**Win % (25 Rounds)**	**+/- Money**
Player 1	5 Wins / 20 Losses	20%	-500
Player 2	6 Wins / 19 Losses	24%	-475
Player Average		*22%*	*-487.5*
Basic State-Based AI	10 Wins / 15 Losses	40%	-375
Model-Based AI	16 Wins / 9 Losses	64%	-225

Average Win Percentage Across All Tests

Player 1	20%
Player 2	31.33%
Player Average	*25.67%*
Basic State-Based AI	30.67%
Model-Based AI	56.67%

Outcomes Achieved

Therefore as we can see from the test results above, our model-based AI implementation performed far better than both random chance, and your average player. The basic AI also performed slightly better than an average player. These results also show that our actions as average players result in relatively the same odds of winning as random chance.

After 100 rounds of blackjack, it is obvious that our initial goal has been achieved and the AI implementations perform better than an average player. In the long-term results of each scenario above, the Model-based AI has more money than any other player or AI. As we can also see in the results above our model-based AI implementation performed better than human players across all tests and therefore showed reliability.

Compared to the results achieved in the research paper [2] we also calculated the random chance of winning to also be around ~25% and our AI had an expected win rate of around ~56%. This is slightly higher than the ~42% from the research paper that our AI was based on. However, this is likely due to the difference in sample size between our implementation and the papers.

Discussion

Implications of the Project Implementation

This shows that with knowledge of which cards are in play, you can make enough informed decisions to keep wins higher than losses which is the important part of any betting card game such as blackjack. This process is called card counting which can be really hard for an average person to do for every game which is why our model agent has more overall wins than other players in the environment.

Limitations of the Project Implementation

Since it's a model-based AI and the environment is a stochastic, partially observable environment, the Agent can only make decisions based on the state it can observe therefore it will not always win. The decision-making of the agent is the probability of the next state given the current state of the environment since the environment is static. This ends up with the agent having losses but there is no way a game of probability cannot have losses.

Directions for Future Work

The direction we are trying to move is towards q-learning, this is the get the proper long-term reward for each action at each state. This does not happen in our current implementation of the model which is just a probabilistic action implementation.

References

[1] Jones, C., “Blackjack Basic Strategy Chart,” , 2020. URL https://www.blackjackapprenticeship.com/blackjack-strategy-charts/.

[2] Geiser, J. and Hasseler, T. “Beating Blackjack - A Reinforcement Learning Approach” URLhttps://web.stanford.edu/class/aa228/reports/2020/final117.pdf